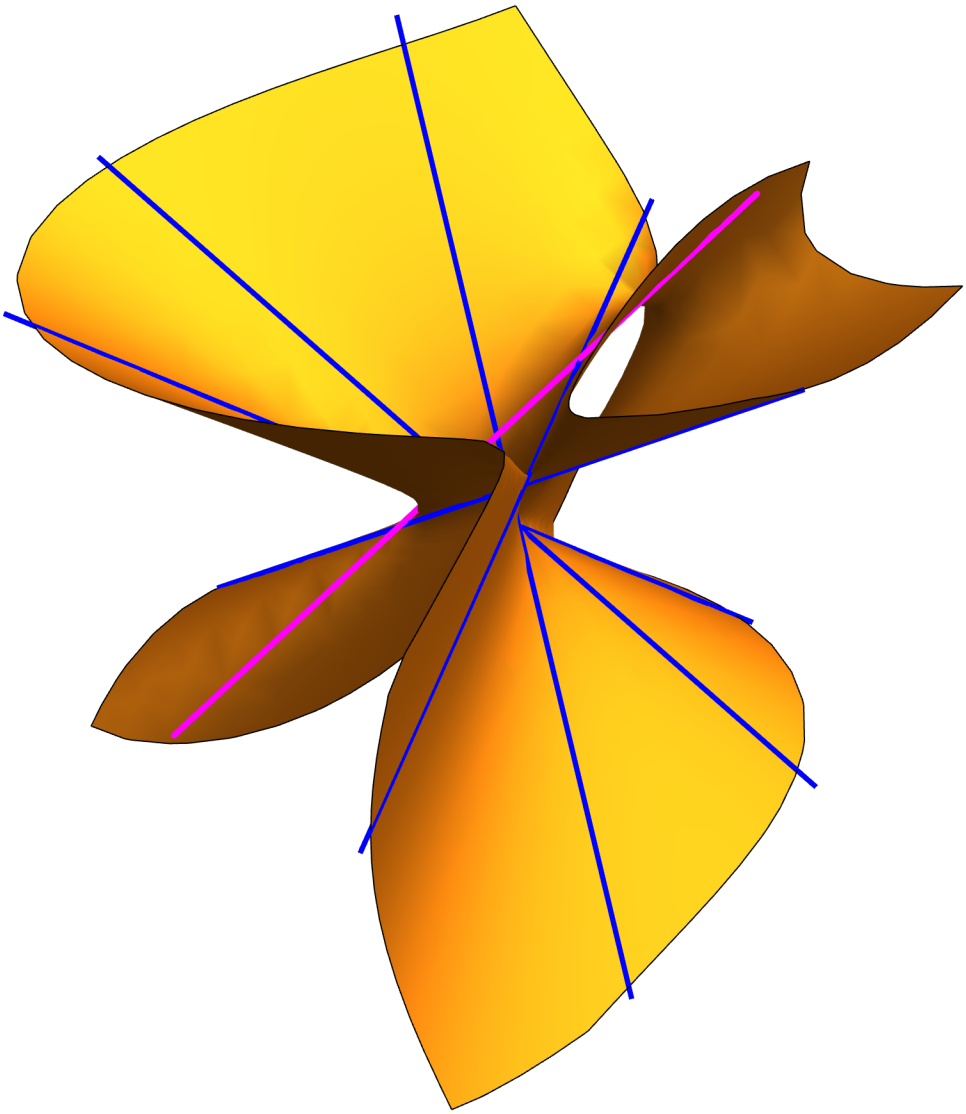
December 2021

# Surface Book
# Chapter 1

Barry H Dayton
barryhdayton.space



# 0. Introduction

Surfaces are much more complicated than curves. For example Riemann defined genus of a curve in the 1850's. But there was no genus of a 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 Chapter 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 will be a new, inclusive `GlobalFunctionsS.nb` notebook for this book. Global functions specifically for surfaces may end in NS (naive surface) or RS (rational surface).

---

## Table of Contents

**1.** Chapter One

**1.1.** Naive Surfaces

**1.1.1.** Definition and Examples 4

**1.1.2.** Regular and Smooth surfaces. 7

**1.2.** Rational Parametric Surfaces 10

**1.3.** Implicit Equations for rational parametric surfaces

**1.3.1.** Theoretical Method 17

**1.3.2.** Direct Method 19

**1.4.** The Torus Story

**1.4.1.** Preliminaries 24

**1.4.2.** The Torus 25

**1.5.** Curves in Surfaces

**1.5.1.** Curves in Parametric Surfaces 39

**1.5.2.** Curves in Implicit Surfaces 40

**1.5.3.** Parametric Curves in Implicit surfaces 42

**1.5.4.** Some new code for old functions 43

**1.5.5.** Ovals and Pseudo-lines 44

**1.6.** Rational Points and Rational Surfaces 46

**1.7.** Quadric Surfaces 52

**1.8.** Trigonometric Parameterization

**1.8.1.** Quadric Surfaces 57

**1.8.2.** Other parametric surfaces given by trigonometry 62

**1.8.3.** The Klein Bottle 66

**1.9.** Lines on a cubic Surface 68

**1.9.1.** The Double 6 configuration 69

**1.9.2.** The Theory 69

**1.9.3.** The Problems 70

**1.9.4.** The Construction 77

**1.9.5.** Additional Lines on the Cubic 79

**1.9.6.** The implicit equation 81

**1.9.7.** Finding lines on a smooth implicit cubic 83

**1.9.8.** Lines on the Clebsch Diagonal Cubic. 89

## Other material not included

There are 3 appendices to Section 1.9 giving most of the calculations. These are available separately on my website in notebook form only. Also on my website is the GlobalFunctionsS notebook with the global functions I use here along with many of the functions from my previous books, some of which are updated, and an alphabetical index to these functions with syntax.



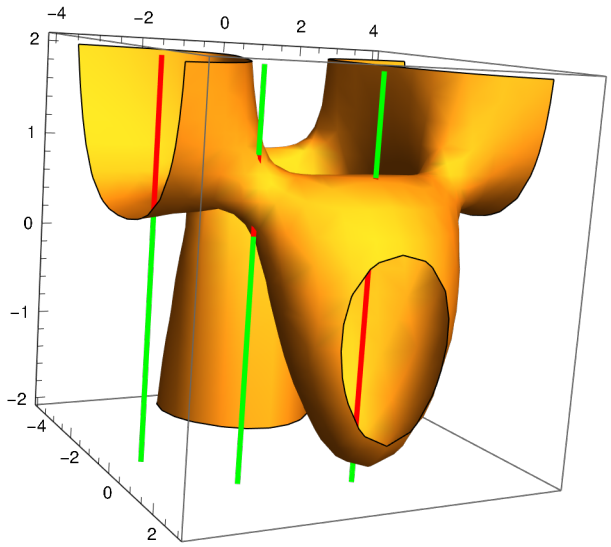
# 1.1 Introduction to Naive Surfaces

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.

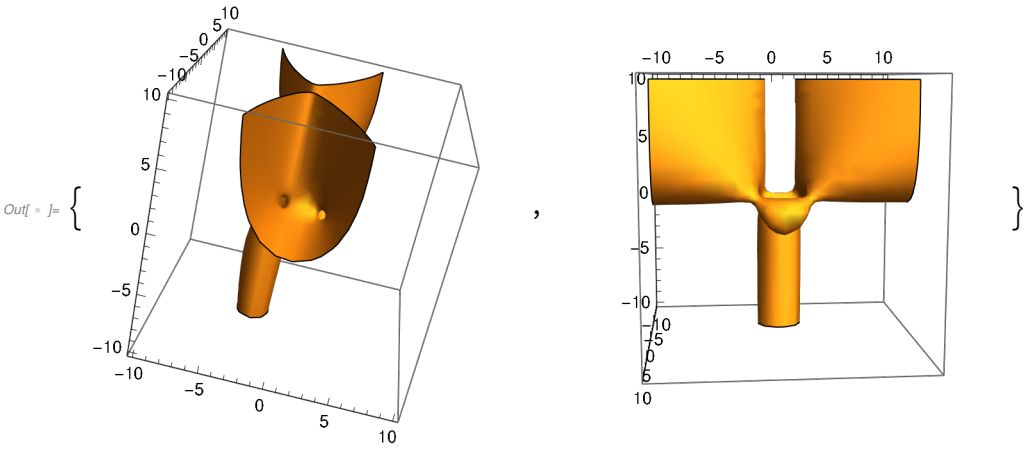
Note this plot changes as we change the bounding box or orientation. We can see more or less of the

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}
```

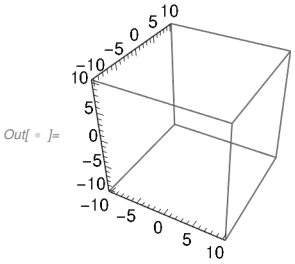
Out[ ]=



Some things can go wrong . The equation $x\text{^}2 + y\text{^}2 + z\text{^}2 = 0$ 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.

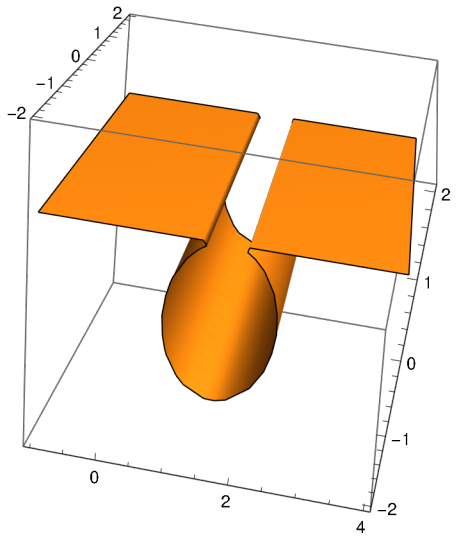
In[ ]:=
```
ts2 = N[Expand[(-1 + z) × (48 - 80 x + 25 x^2 + 16 z^2)]]
```

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]
```

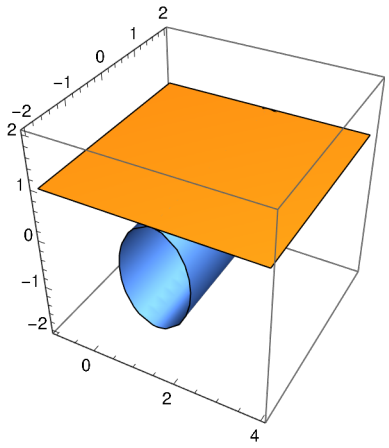
Out[ ]=




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]
```

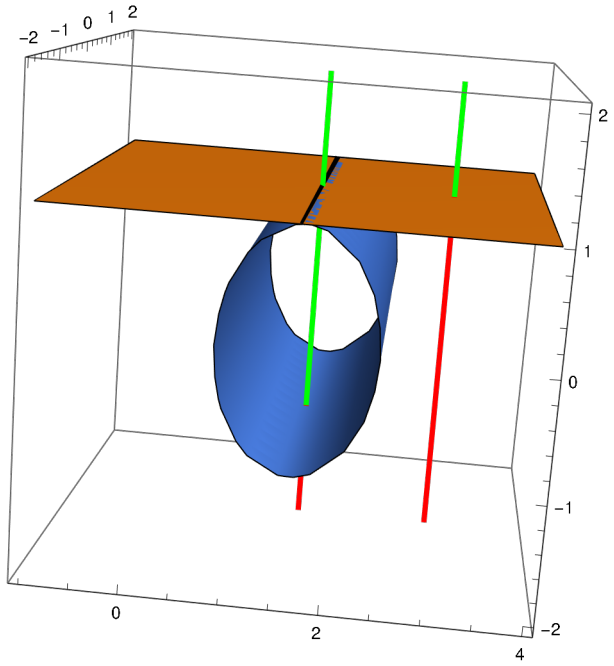
» Square Free

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!

## 1.2 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)
```

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

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

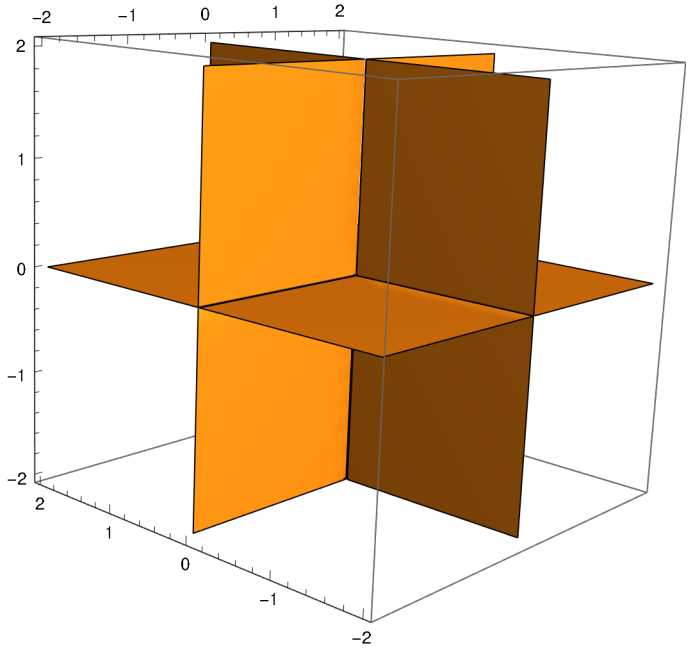
***Jordan - Brouwer*** *Let f be a non-degenerate square free polynomial giving a smooth surface. Then the surface f is two sided, moreover for p in the surface there is a neighborhood of p which is topologically an open plane disk.*

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.

# 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

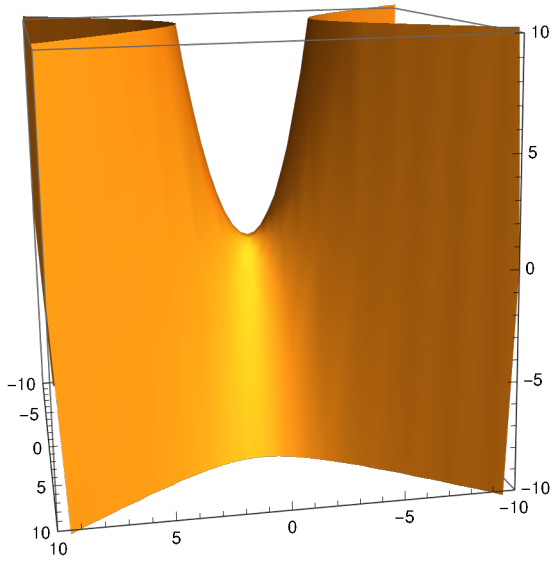
In[ ]:=
```
F1 = {s, t, s^2 - t^2}
```

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]
```

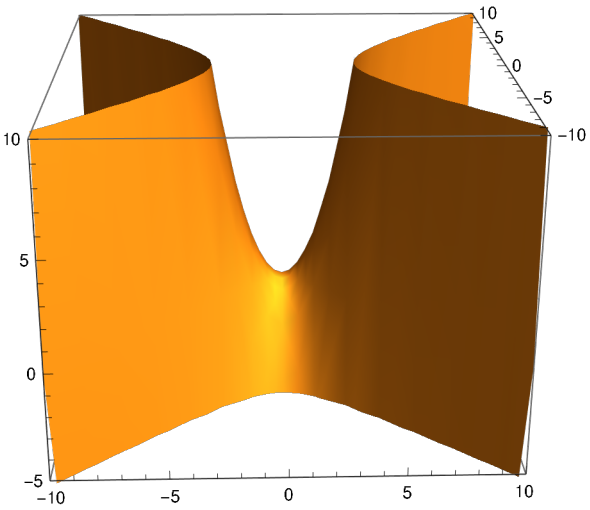
Out[ ]=




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.

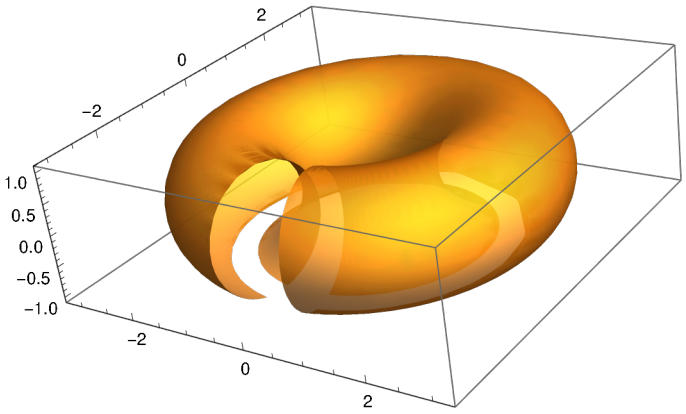
Here is a non-trivial example of a rational parametric surface, the *torus*. Note in this case the definition does not give a common denominator but it is easily seen that a common denominator would be $(1+s^2)\times(1+t^2)$.

In[ ]:=
```
Ts = {4 s (1 + t + t^2)/((1 + s^2) × (1 + t^2)), -(2 × (-1 + s^2 - t + s^2 t - t^2 + s^2 t^2))/((1 + s^2) × (1 + t^2)), (1 - t^2)/(1 + t^2)};
```

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

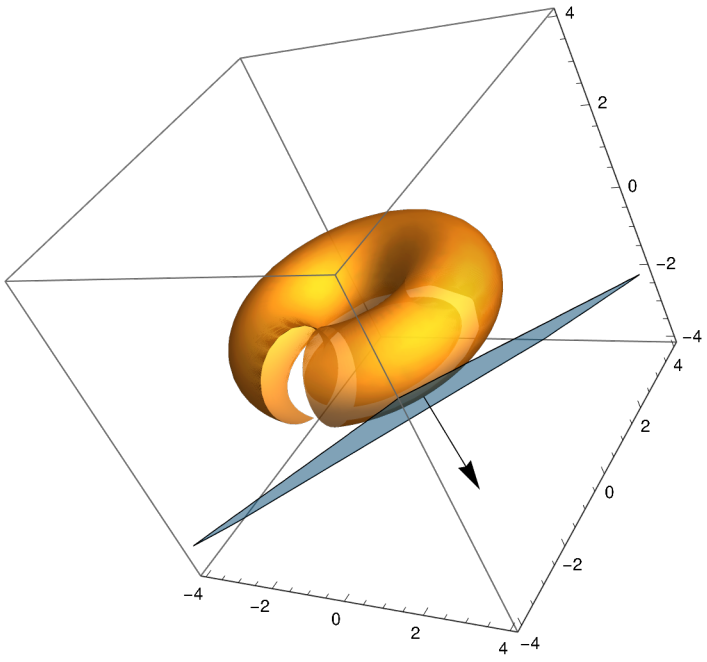
or, better

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[ ]= $-0.4576 + 0.09984\ x - 0.07488\ y - 0.1664\ z$

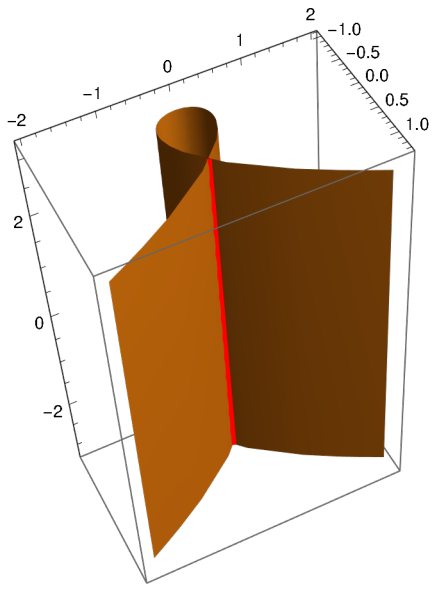
As with naive surfaces a rationally parameterized surface $F(s, t)$ is regular at {s0, t0} if there is a tangent plane at $F$(s0, t0). But as with naive algebraic surfaces regularity at {s0,t0} does not imply smoothness at F(s0,t0). But the situation is very different. For naive surfaces it is a local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

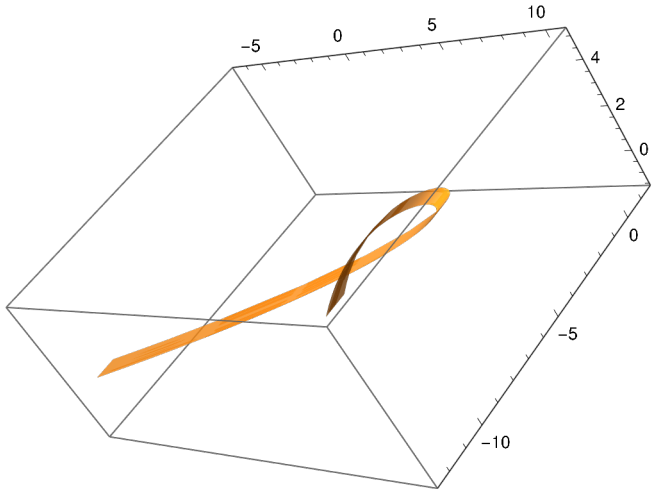
are non - zero, so all of these points are regular in the the parameters. The problem is that different parameter values give the same points. While harder to deal with the problem is no worse than with ts2 so we have nothing to do.

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

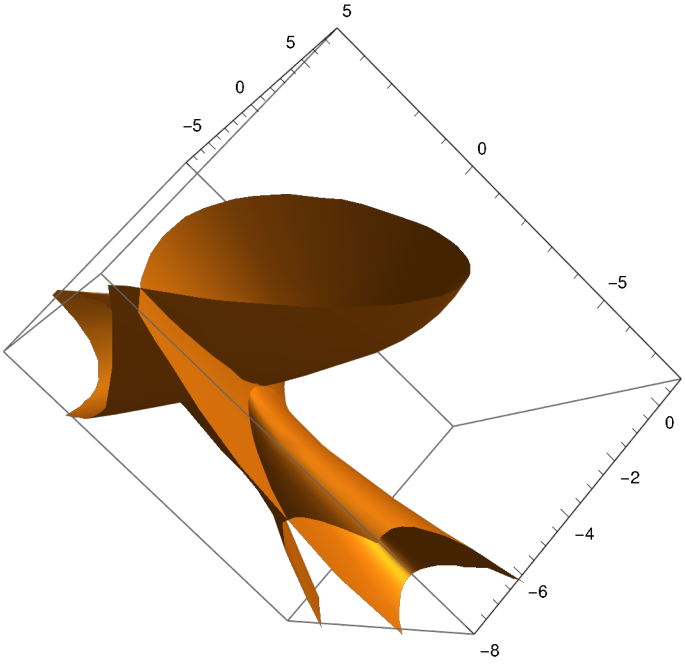
of a point of the parametric surface just working locally, mainly that rational parametric surfaces, even the ribbon, are subsets of naive algebraic surfaces.

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 $X[1]$, $X[2]$...., $X[n]$. 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 = {(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)};
```

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[ ]=
$$\{Y[1] \to s,\ Y[2] \to t,\ Y[3] \to s\, t,\ Y[4] \to s^2,\ Y[5] \to s^2\, t\}$$

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

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⟧
```

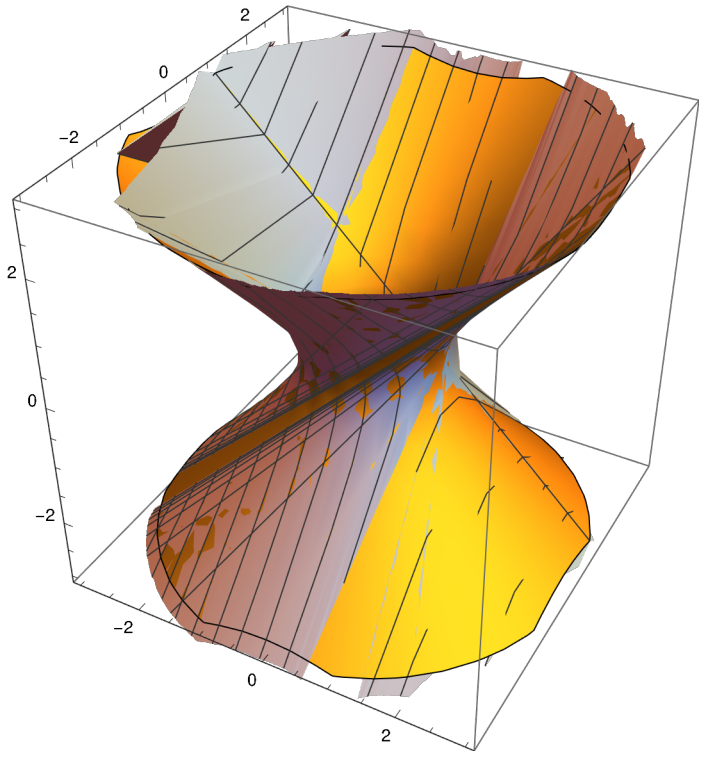
Out[ ]=
$$1. - 1.\, x^2 - 1.\, y^2 + 1.\, z^2$$

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.

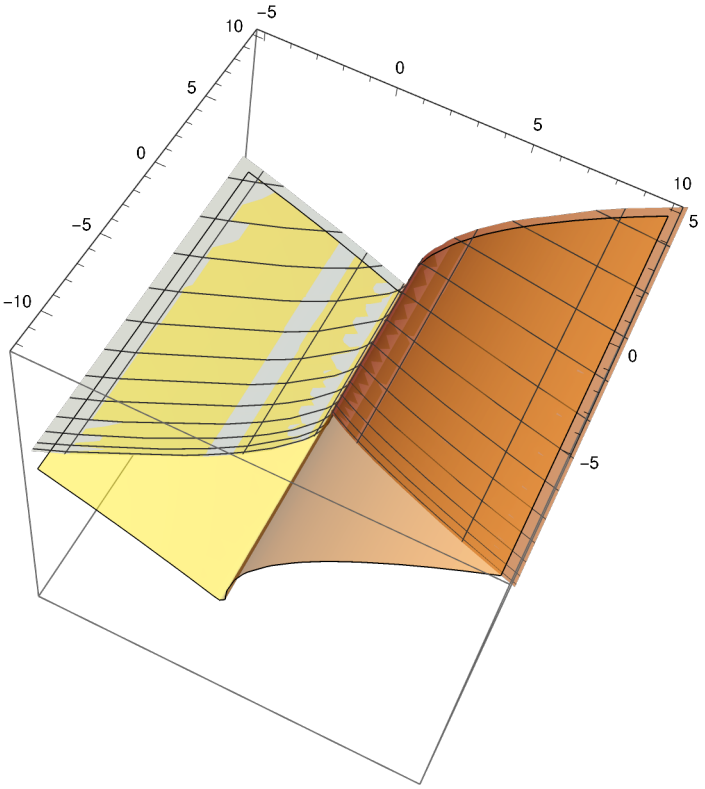
```
par2affRS[F_, d_, md_, T_, X_] :=
 Module[{n, TB, ar, cr, SA, AK, mon, ncr, nak, NSA, medNSA, FA, SAA},
  n = Length[X];
  If[Length[F] ≠ n, Echo["Dimension mismatch F,X"]; Abort[]];
  TB = Expand[Table[mon /. Thread[X → F], {mon, mExpsMD[md, X]}]];
  cr = <|CoefficientRules [#, T]|> & /@ TB;
  ncr = 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]
```

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

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

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

In[ ]:= 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 = {(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)};
```

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

```
TransformationFunction[A][{t^d, t^(d-1), …, t}]
```

or the equivalent

```
fltMD[{t^d, t^(d-1), …, 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. $AB$ is defined and $AB = 0$
2. If $vB = 0$ then $v$ is in the row space of $A$
3. If $vB = 0$ then $v$ is in the row space of $A$

That is, $A$, $B$ are maximal satisfying $AB = 0$. 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.

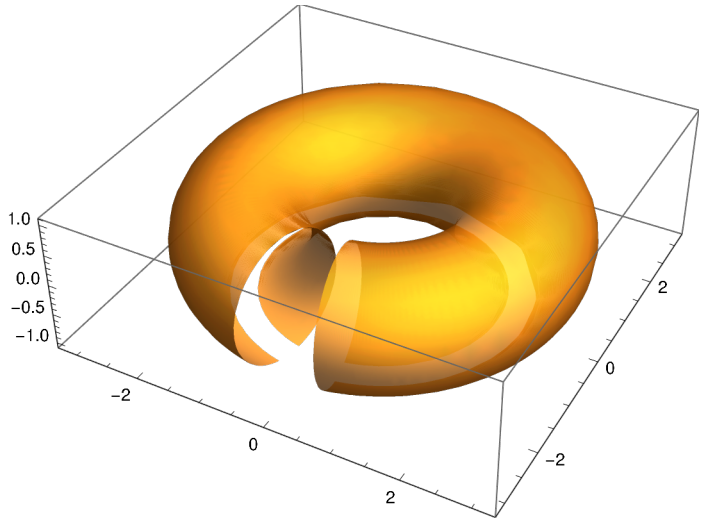
## 1.4 .2 The Torus

Here is our rationally parametrized surface .

In[ ]:=
```
T = {4 s (1 + t + t^2) / ((1 + s^2) × (1 + t^2)), -(2 × (-1 + s^2 - t + s^2 t - t^2 + s^2 t^2)) / ((1 + s^2) × (1 + t^2)), ((1 - t^2) × (1 + s ^ 2)) / ((1 + t^2) × (1 + s ^ 2))};
```

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

Since $1=\frac{1+t\text{^}2}{1+t\text{^}2}$ 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}}
```

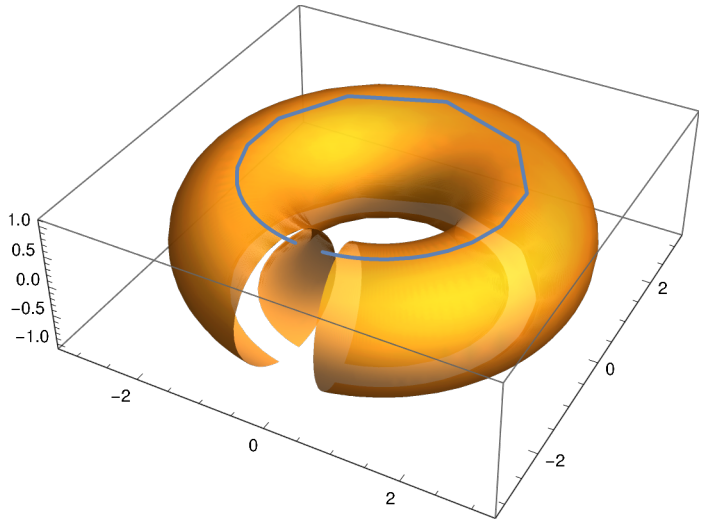
Checking

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]
```

» Initial Hilbert Function {1, 3, 5}

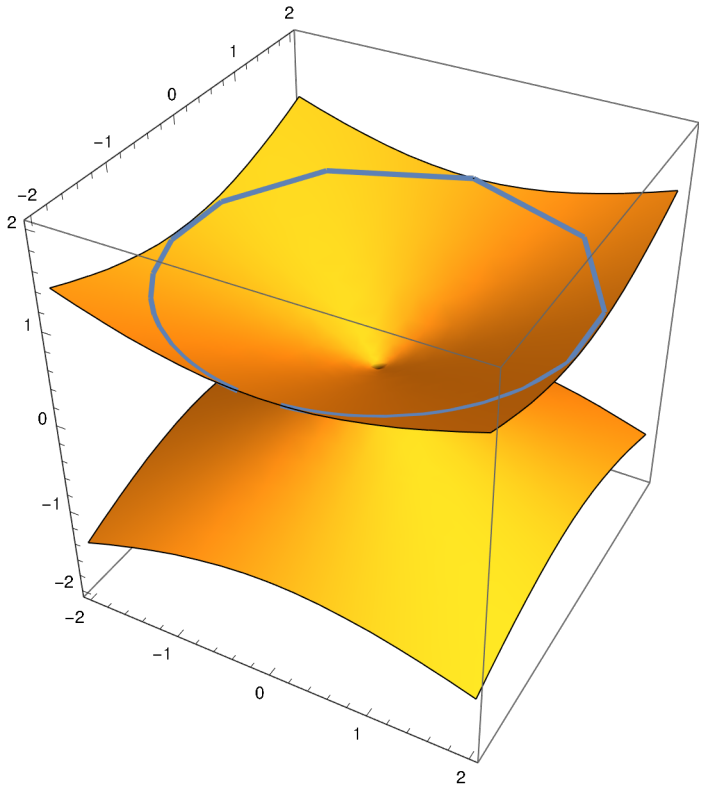
» Final Hilbert Function {1, 3, 5}

Out[ ]= $\{1. - 1.\, z,\ -0.25\, x^2 - 0.25\, y^2 + 1.\, z^2\}$

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

A transformation matrix is

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

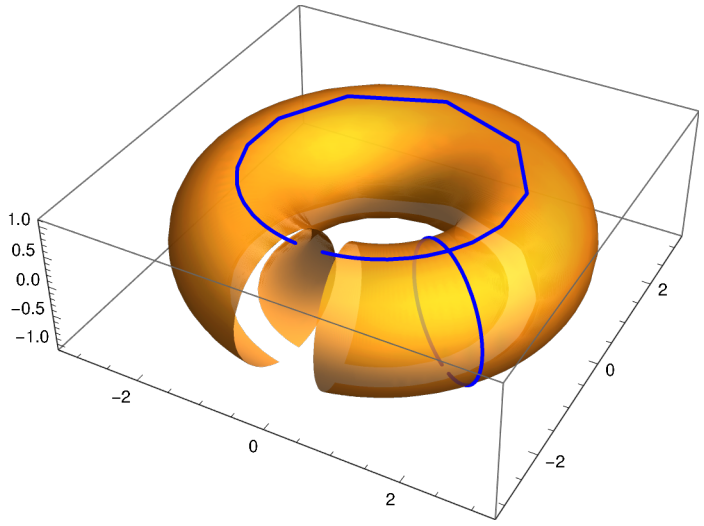
Checking

In[ ]:=
```
fltMD[{t^2, t}, As2]
```

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

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]
```

» Initial Hilbert Function {1, 3, 5}

» Final Hilbert Function {1, 3, 5}

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

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]
```

» Initial Hilbert Function {1, 3, 5, 4, 4}

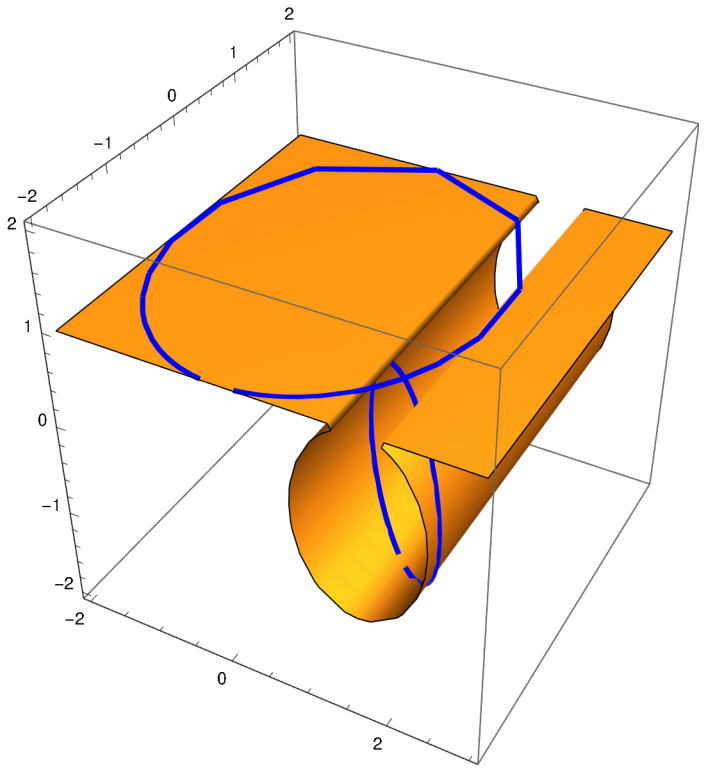
» Final Hilbert Function {1, 3, 5, 4, 4}

Out[ ]= $\{-0.75\,x - 1.\,y + 0.75\,x\,z + 1.\,y\,z,\ 4.8 - 5.\,x - 1.6\,x^2 + 1.\,x^3 - 1.6\,y^2 + 1.\,x\,y^2 + 1.6\,z^2 + 1.\,x\,z^2,$
$-3.\,x + 0.75\,x^3 - 4.\,y + 1.\,x^2\,y + 0.75\,x\,y^2 + 1.\,y^3,$
$-3. + 5.\,x - 1.5625\,x^2 + 3.\,z - 5.\,x\,z + 1.5625\,x^2\,z - 1.\,z^2 + 1.\,z^3\}$

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

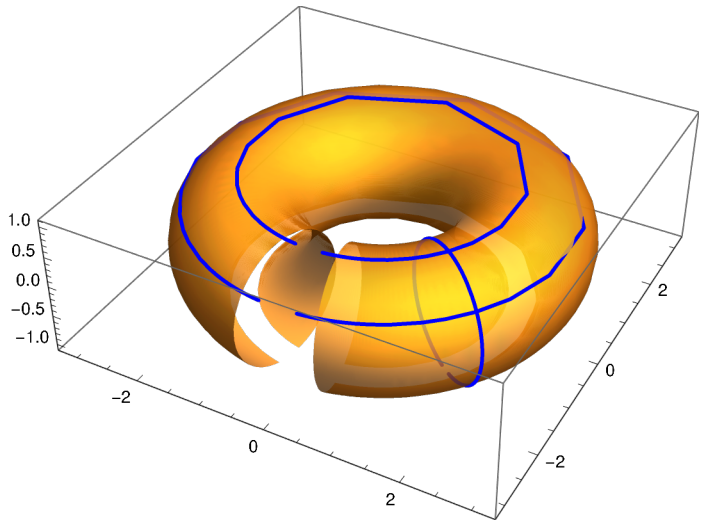
In[ ]:=
```
Atp5 = {{0, 5.6, 0}, {-2.8, 0, 2.8}, {.6, 0, .6}, {1, 0, 1}};
```

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]
```

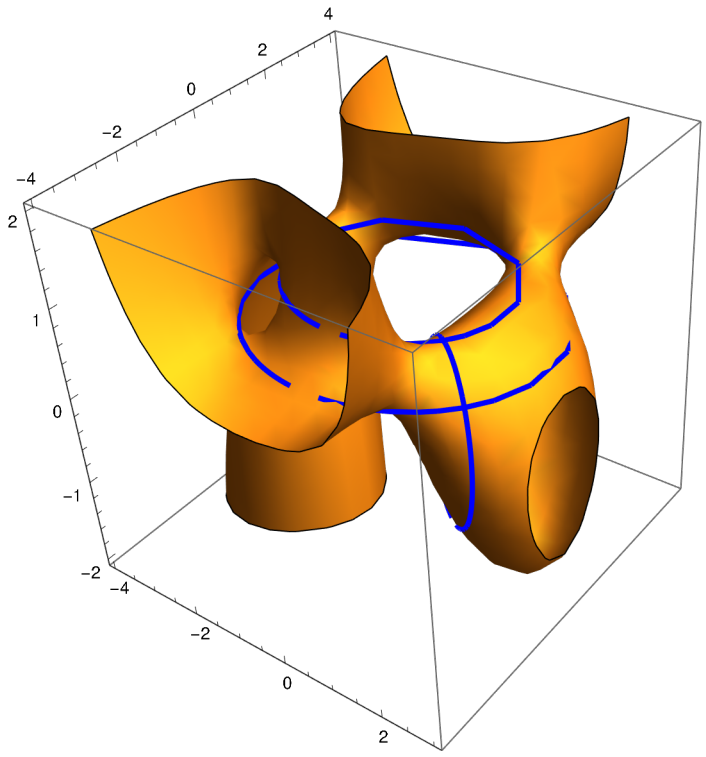
» Initial Hilbert Function {1, 3, 6, 7, 6}

» Final Hilbert Function {1, 3, 6, 7, 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}};
```

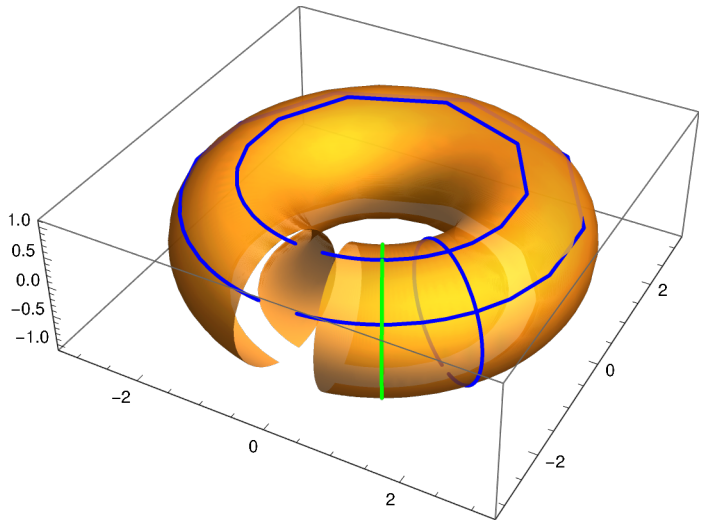
Checking :

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[ ]=
$$\{-6. + 4.33333\, x - 0.333333\, x^3 - 5.05556\, y + 0.388889\, x^2\, y - 0.333333\, x\, y^2 + 0.388889\, y^3 + 3.\, z - 2.66667\, x\, z + 1.\, x^2\, z + 3.11111\, y\, z + 1.\, y^2\, z - 2.\, z^2 - 0.333333\, x\, z^2 + 0.388889\, y\, z^2 + 1.\, z^3,$$
$$-6.4 + 2.03175\, x - 2.92619\, x^2 - 0.203175\, x^3 + 0.154762\, x^4 - 2.37037\, y - 13.\, x\, y + 0.237037\, x^2\, y + 1.\, x^3\, y - 0.914286\, y^2 - 0.203175\, x\, y^2 + 0.154762\, x^2\, y^2 + 0.237037\, y^3 + 1.\, x\, y^3 + 4.28571\, x^2\, z + 8.\, x\, y\, z + 3.04762\, y^2\, z - 2.13333\, z^2 - 1.21905\, x\, z^2 + 0.154762\, x^2\, z^2 + 1.42222\, y\, z^2 + 1.\, x\, y\, z^2,$$
$$-19.125\, x^2 + 1.40625\, x^4 - 35.7\, x\, y + 2.625\, x^3\, y - 13.6\, y^2 + 2.40625\, x^2\, y^2 + 2.625\, x\, y^3 + 1.\, y^4 + 13.5\, x^2\, z + 25.2\, x\, y\, z + 9.6\, y^2\, z,$$
$$16.8 - 5.33333\, x + 8.525\, x^2 + 0.533333\, x^3 - 0.40625\, x^4 + 6.22222\, y + 35.7\, x\, y - 0.622222\, x^2\, y - 2.625\, x^3\, y + 3.\, y^2 + 0.533333\, x\, y^2 - 0.40625\, x^2\, y^2 - 0.622222\, y^3 - 2.625\, x\, y^3 - 13.5\, x^2\, z - 25.2\, x\, y\, z - 9.6\, y^2\, z + 5.6\, z^2 + 3.2\, x\, z^2 + 1.\, x^2\, z^2 - 3.73333\, y\, z^2 + 1.\, y^2\, z^2\}$$

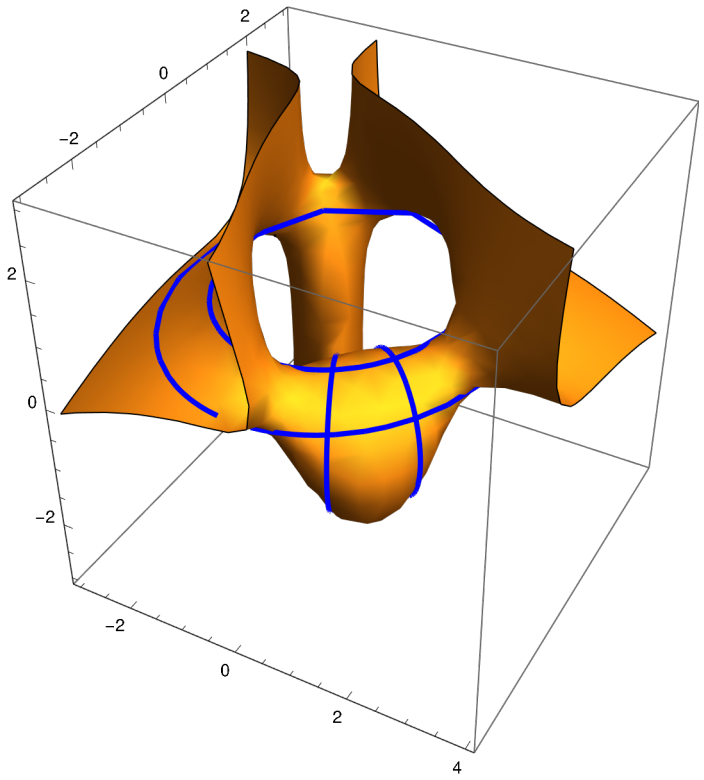
In[ ]:=
```
Length[basis4]
```

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.

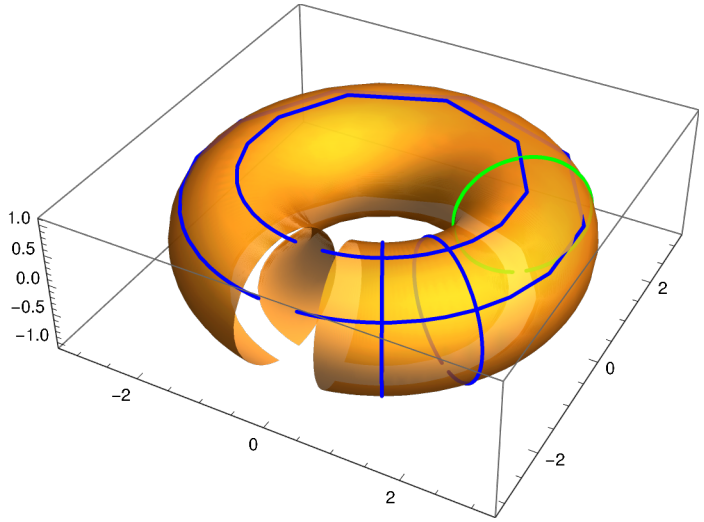
Now we add another vertical circle .

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]
```

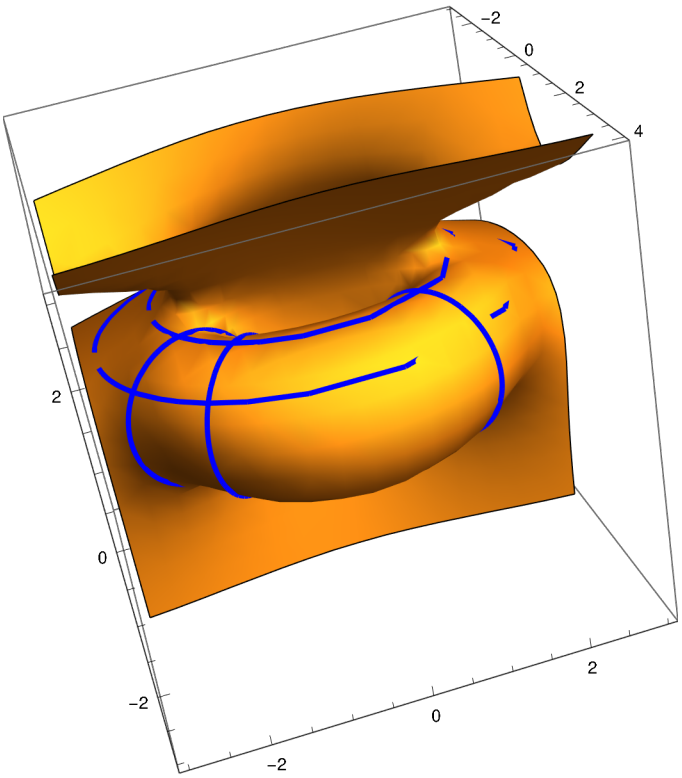
» Initial Hilbert Function {1, 3, 6, 10, 12}

» 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.

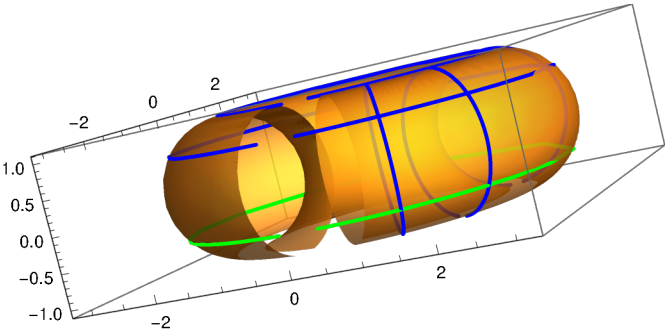
One more horizontal circle .

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[ ]=




```
In[ ]:= 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[ ]= $9 - 10\,x^2 + x^4 - 10\,y^2 + 2\,x^2\,y^2 + y^4 + 6\,z^2 + 2\,x^2\,z^2 + 2\,y^2\,z^2 + z^4$

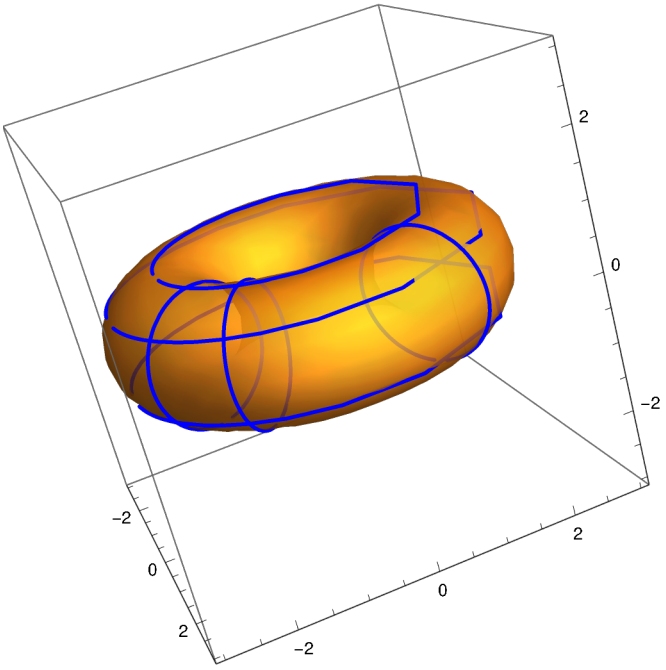
we actually get an integer coefficient surface.

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 containing 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, however there are two cases.

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

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

```
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}

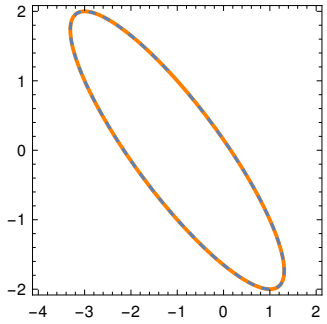
» Final Hilbert Function {1, 3, 5}

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]]}]
```

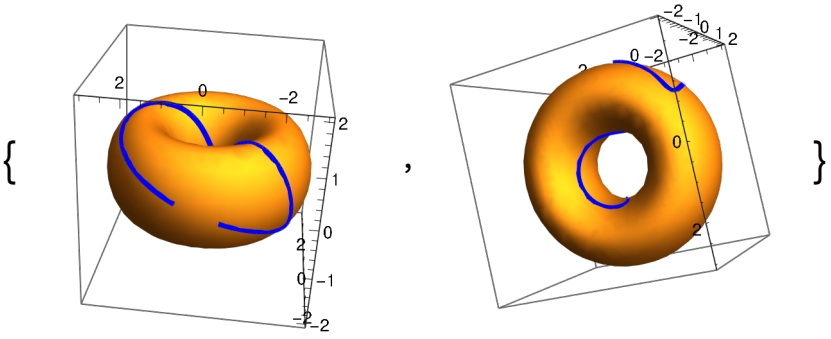
Out[ ]=

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

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

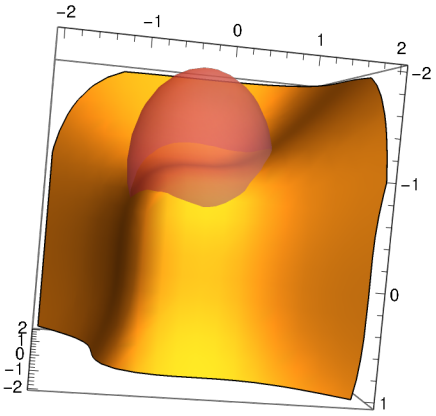
In[ ]:=
```
fermat = x^3 + y^3 + z^3 + 1;
```

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}]
```

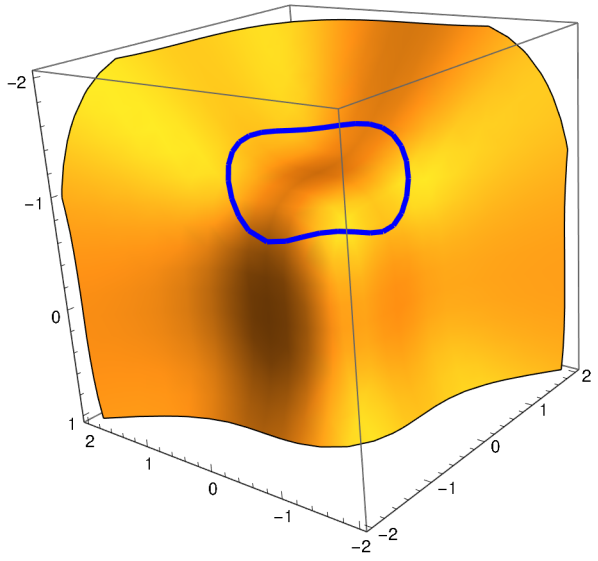
Out[ ]=
```
{{-1.24155, 0., 0.970389},
 {-1.23247, 0.195949, 0.952659}, {-1.21237, 0.384334, 0.898436},
 {-1.19336, 0.557564, 0.8073}, {-1.18662, 0.707692, 0.681428},
 {-1.19629, 0.827291, 0.526364}, {-1.21646, 0.911499, 0.349736},
 {-1.23537, 0.958754, 0.159351}, {-1.24116, 0.969773, -0.0371887},
 {-1.22443, 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

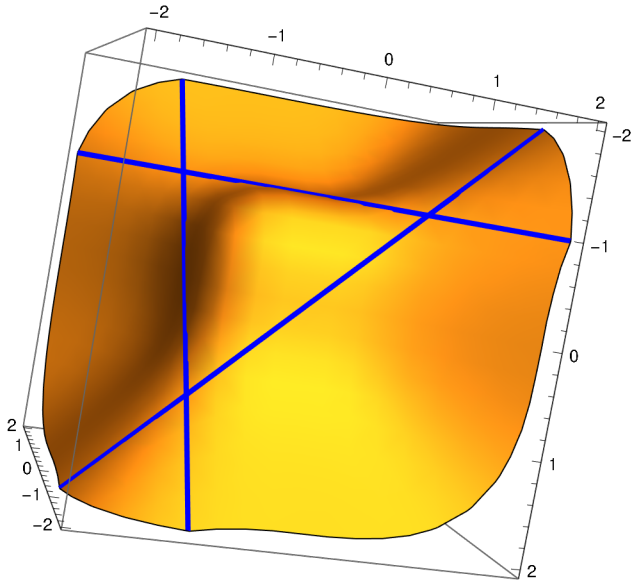
A third possibility is to use a parametric curve with the implicit surface. However this requires some cleverness as there is no general method for doing this. For example one may observe that the that the Fermat surface above contains the parametric lines $\{t, -t, -1\}$, $\{t, -1, t\}$ and $\{-1, t,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

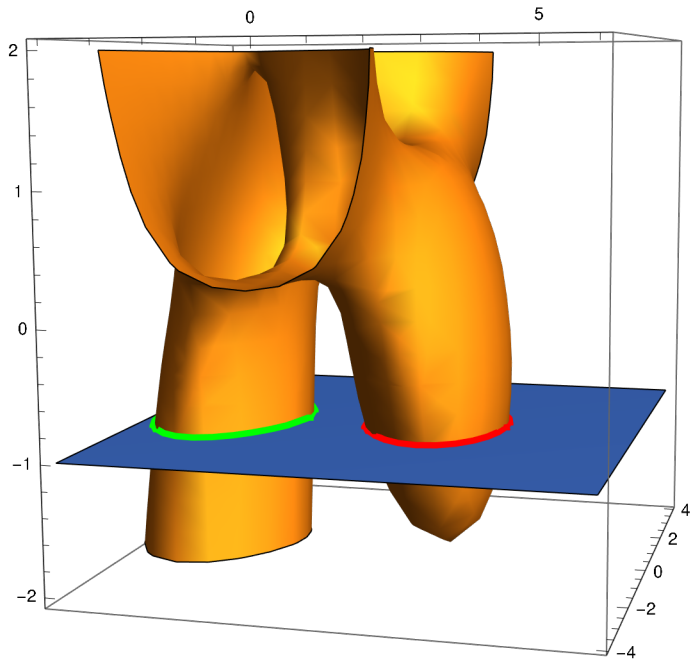
## 1.5.5 Ovals and Pseudo-Lines

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 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;
```

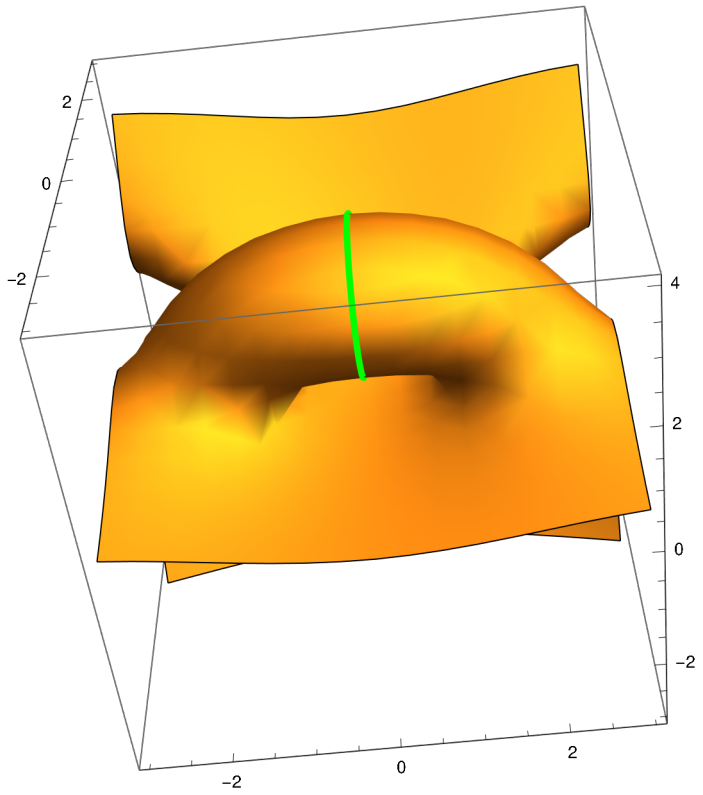
We intersect this with the plane $z = -1$ and get two ovals as shown in the plot in red and green. We suppress the work.




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.

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




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.

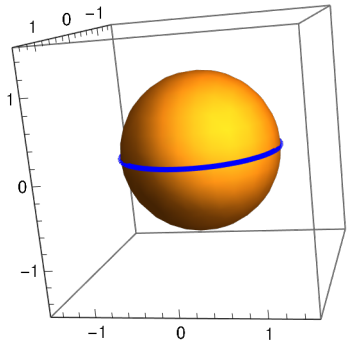
We also note from the torus example that the 3 horizontal curves each meet the three vertical curves in exactly one point. This is in stark difference where any algebraic curve meets an oval in an even number of points by multiplicity. That property of an oval was a crucial step in our proof of Harnak's 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;
equator = {2 t / (1 + t^2), (1 - t^2) / (1 + t^2), 0};
```

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[ ]:= Tor = {4 s (1 + t + t^2) / ((1 + s^2) (1 + t^2)), -(2 (-1 + s^2 - t + s^2 t - t^2 + s^2 t^2) / ((1 + s^2) (1 + t^2))), (1 - t^2) (1 + s^2) / ((1 + t^2) (1 + s^2))};
```

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

```
In[ ]:= p = Tor /. {s → 21 / 4, t → 13 / 7}
Out[ ]= {51 912 / 49 813, -131 325 / 49 813, -60 / 109}
```

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 $\epsilon$ 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[ ]= 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 polynomial 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[w^3 + x^3 + y^3 + z^3 == 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}}
```

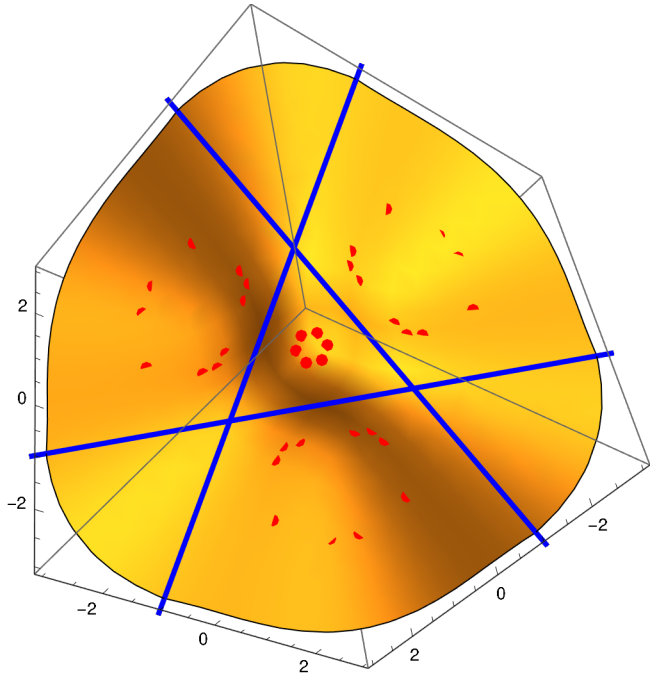
Proceeding this way I found 6 instances which after permuting

```
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.

# 1.7 Quadric Surfaces

The study of quadric surfaces is properly a topic in projective geometry, which will be covered in a later chapter in this book. However some of the ideas recently discussed here are involved and this is a important topic in elementary mathematics so in this section we will begin the discussion.

The standard coverage of this is uneven and misleading. For example the term *hyperboloid of two sheets* is nonsense as all non-degenerate quadric surfaces are rationally parameterized surfaces and hence *of one sheet.* I will suggest some non-standard terminology but suggest that it be widely adopted.

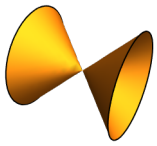
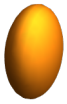
Quadric surfaces are defined from the affine point of view by an equation

$$a_1 x^2 + a_2 x y + a_3 y^2 + a_4 x z + a_5 y z + a_6 z^2 + a_7 x + a_8 y + a_9 z + a_{10} = 0$$

where the coefficients $a_i$ are machine numbers with at least one of $a_1$, $a_2$, …, $a_6$ not zero.
For example a random quadric might be

```
In[ ]:= 4.492182872989918` + 1.5027217857511275` x -
        3.2932471474961034` x^2 - 4.861394482747162` y + 3.21859207861387` x y -
        5.401643964553532` y^2 + 5.226019667264691` z - 0.8091107243142233` x z +
        3.7145392742572234` y z + 5.269463158972744` z^2 == 0
Out[ ]= 4.49218 + 1.50272 x - 3.29325 x^2 - 4.86139 y + 3.21859 x y -
        5.40164 y^2 + 5.22602 z - 0.809111 x z + 3.71454 y z + 5.26946 z^2 == 0
```

Out[ ]=

| Projective Quadric Surfaces | | | | |
|---|---|---|---|---|
| Type | Degenerate | Cone | Ellipsoid | Hyperboloid |
| Possible Picture |  |  |  |  |
| example | $xz=0$ | $z^2=x^2+y^2$ | $x^2+\frac{y^2}{2}+\frac{z^2}{4}=1$ | $x^2+y^2-z^2=1$ |
| singularity ? | line | point | none | none |
| ruled? | two parts | single | none | double |
| essential ovals? | no | no | no | yes |
| Affine Variants | parallel planes | cylinder | parabolic hyperbolic | elliptic parabolic |

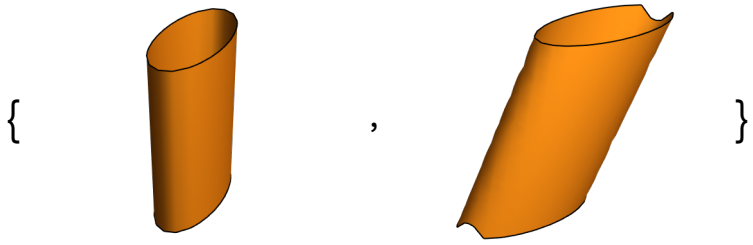
Note that some quadratic polynomials in three variables may not give surfaces . Since we are working in the real domain some may have solution sets that are empty or finite. Some could give lines such as the polynomial $(y - 2x)^2 + (z + 3x)^2 = 0$ which gives the parametric line $\{t, 2t, -3t\}$. Or the polynomial will not be square free such as $(x - 2y + 3z + 1)^2$. I am not including these in this discussion. They can easily be excluded by the fact that they have no regular points.

I will make some comments on the types . Again we are working in the real domain, if we allow complex numbers then ellipsoids are hyperboloids with complex rulings.

The degenerate quadrics are reducible, that is they may be factored, as such they are necessarily singular. In affine space they could be the composite of two parallel planes, but then they meet in an infinite line in projective space.

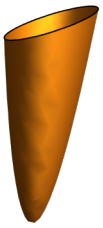
A cylinder is a quadric that is equivalent to a plane quadric where one of the variables x, y, z is absent. For example the equation on the left is $x\text{^}2 + y\text{^}2 - 1$ where that on the right is a rotation applied to this first equation giving

$-1. + 0.69313\ x^2 - 0.77063\ x\ y + 0.50167\ y^2 - 0.00975\ x\ z - 0.24645\ y\ z + 0.0552\ z^2$.

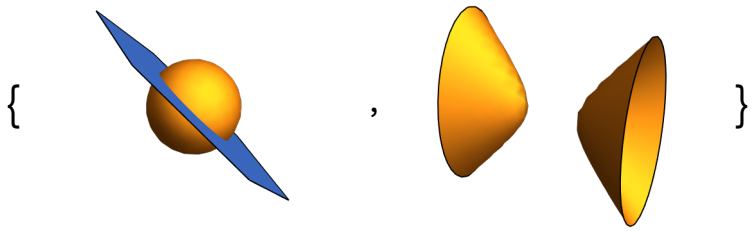


In the left we have a ruled surface of vertical lines, each of one has infinite point {0, 0, 1, 0}. Since all these lines go through this one point it is a cone in projective space. Rotating it still gives a cone. Thus in projective space a cylinder is just a cone with the vertex in the infinite plane.

If we perform a FLT transform on the ellipsoid above which sends one point to an infinite point we get a *parabolic ellipsoid* ( called a paraboloid in the literature).



On the other hand if we cut the ellipsoid with a plane which goes to infinity we get

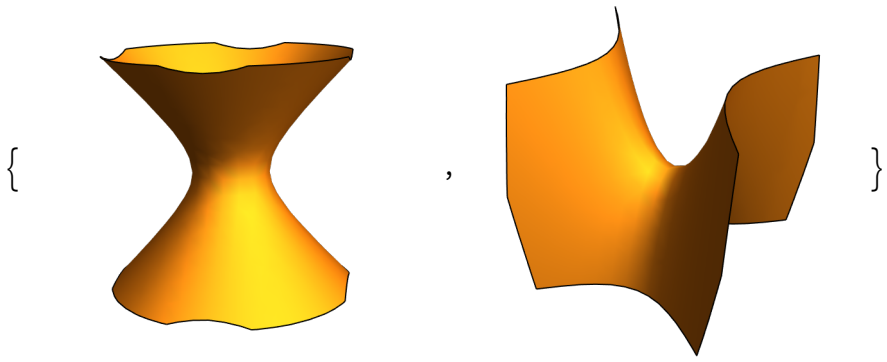


which wrongly was called a hyperboloid of 2 sheets but I call it a *hyperbolic ellipsoid.* Since every hyperbolic ellipsoid and every parabolic ellipsoid are FLT images of the ellipsoid then the properties of no non-null-homotopic (essential) ovals and two sided-ness are preserved for all of these.

In the affine plane there are two hyperboloids. In addition to the one pictured above, and below left there is the *elliptic hyperboloid* otherwise known as just the *hyperboloid. T*he below right is the *parabolic hyperboloid* , otherwise known as the hyperbolic paraboloid.

In[ ]:=
```
{ellhyp = ContourPlot3D[1 - x^2 - y^2 + z^2 == 0, {x, -3, 3}, {y, -3, 3},
    {z, -3, 3}, Mesh → None, Axes → None, Boxed → False, ImageSize → Small],
 ContourPlot3D[z == x^2 - y^2, {x, -3, 3}, {y, -3, 3}, {z, -3, 3},
  Mesh → None, Axes → None, Boxed → False, ImageSize → Small]}
```

Out[ ]=



The two hyperboloids do share 3 important properties

1) These are doubly ruled surfaces .

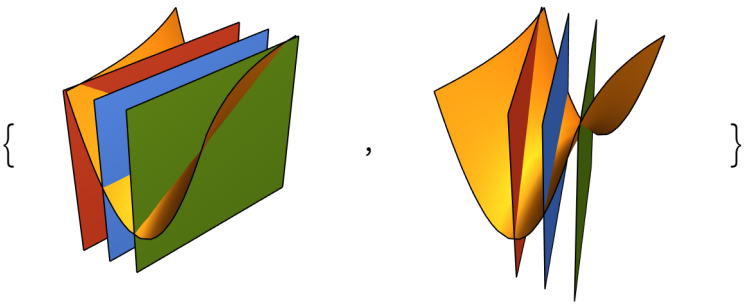
2. The tangent plane at every point cuts the hyperboloid in two lines, one from each ruling.

3. The hyperboloid is determined by any 3 skew lines, that is any three skew lines in 3-space are part of one ruling of a hyperboloid.

The difference is this: in the parabolic hyperboloid all the lines in one ruling are all parallel to one plane, this is not true of the elliptic paraboloid. For example consider our parabolic hyperboloid

In[ ]:=
```
{ContourPlot3D [{h1 == 0, x - y == 0, x - y == 1, x - y == -1}, {x, -3, 3},
  {y, -1.2, 1.2}, {z, -3, 3}, Mesh → None, Axes → None, Boxed → False],
 ContourPlot3D [{h1 == 0, x + y == 0, x + y == 1, x + y == -1}, {x, -3, 3},
  {y, -1.2, 1.2}, {z, -3, 3}, Mesh → None, Axes → None, Boxed → False]}
```

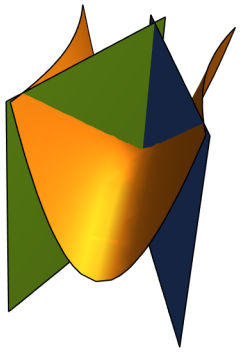
Out[ ]=



Both the families of planes $x + y = a$, and $x - y = b$ as $a$, $b$ run through the real numbers cut this surface in lines which must be skew to each other but each plane of the form $x + y = a$, intersects each plane of the form $x - y = b$ in a line which meets the surface $h_1$ in one point.

In[ ]:=
```
ContourPlot3D [{h1 == 0, x + y == 1, x - y == -1}, {x, -3, 3},
 {y, -1.2, 1.2}, {z, -3, 3}, Mesh → None, Axes → None, Boxed → False]
```

Out[ ]=



Thus the skew lines 3) above will all be parallel to one particular plane if and only if the surface they generate is a parabolic hyperboloid. This fact was observed in the book by Hilbert and Cohn-Vossen, who also observed that the elliptic hyperboloids contain an ellipse which is essential although they did not state this fact in those words.

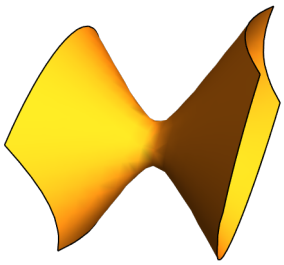
Importantly one notes that the parabolic hyperboloid does not occur in projective space as there is no such thing as parallel planes, any two planes meet. Moreover if we transform the parabolic hyper - boloid by an appropriate projective FLT it becomes just a hyperboloid:

In[ ]:=
```
B = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 0, 1}, {1, 1, 1, -1}};
h2 = FLT3D[{z - x^2 + y^2}, B, {x, y, z}][[1]]
```

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

In[ ]:=
```
ContourPlot3D[h2 == 0, {x, -3, 3}, {y, -3, 3}, {z, -3, 3},
 Mesh → None, Axes → None, Boxed → False, ImageSize → Small]
```

Out[ ]=

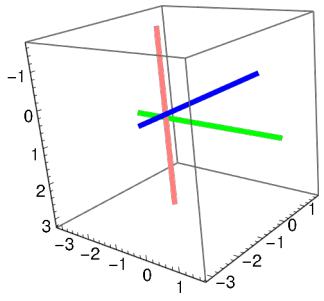


Here is a seemingly impossible set of skew lines to appear in an elliptic hyperboloid.

```
L1f = {t, 0, 0};
L2f = {0, t, 1};
L3f = {-1, -1, t};
```

In[ ]:=
```
ParametricPlot3D[{L1f, L2f, L3f}, {t, -3, 3}, PlotStyle → {Blue, Green, Pink}]
```

Out[ ]=




The equations are

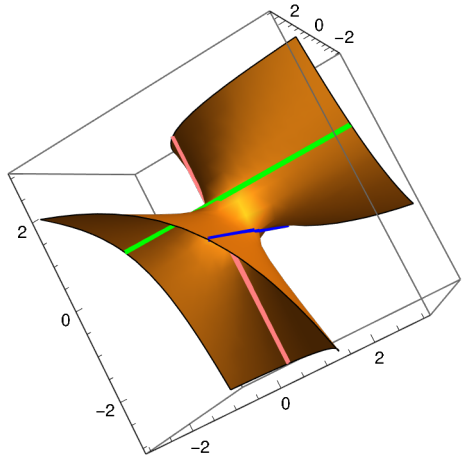
In[ ]:=
```
L1eq = {y, z};
L2eq = {x, z - 1};
L3eq = {x + 1, y + 1};
```

In[ ]:=
```
L1syl = sylvesterMD[L1eq, 2, {x, y, z}];
L2syl = sylvesterMD[L2eq, 2, {x, y, z}];
L3syl = sylvesterMD[L3eq, 2, {x, y, z}];
hp2 = First[
  Chop[vectorSpaceIntersection3[L1syl, L2syl, L3syl, dTol], dTol].mExpsMD[2, {x, y, z}]]
```

Out[ ]=
```
-0.5 y - 0.5 x y - 0.5 x z + 0.5 y z
```

In[ ]:=
```
Show[ContourPlot3D [hp2 == 0, {x, -3, 3}, {y, -3, 3}, {z, -3, 3}, Mesh → None],
 ParametricPlot3D [{L1f, L2f, L3f}, {t, -3, 3}, PlotStyle → {Blue, Green, Pink}]]
```

Out[ ]=



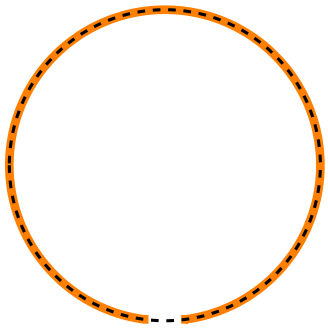

While all quadrics have a rational parameterization not all are quadratic in the parameters, for example our example of the elliptic hyperboloid in section 1.3 which has cubic terms. Conversely not every quadratic parameterization is a quadric surface. We will see some more examples in the next section.

# 1.8 Trigonometric 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[ ]:=
```
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[ ]=



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.*

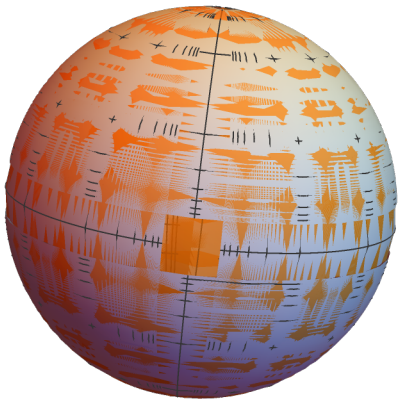
## 1.8.1 quadric surfaces

The most famous surface with such parameterization is the Sphere .

In[ ]:=
```
trigSphere = {Sin[u] Cos[v], Sin[u] Sin[v], Cos[u]};
rationalSphere = {(1 - t^2) × 2 s / ((1 + t^2) × (1 + s^2)), (1 - t^2) × (1 - s^2) / ((1 + t^2) × (1 + s^2)), 2 t (1 + s^2) / ((1 + t^2) × (1 + s^2))};
```

In[ ]:=
```
Show[ParametricPlot3D [trigSphere , {u, -Pi , Pi}, {v, -Pi , Pi},
  PlotRange → 1.2 , PlotStyle → Directive[Orange , Opacity[.7]], Mesh → None],
 ParametricPlot3D [rationalSphere , {t, -15 , 15}, {s, -15 , 15},
  PlotStyle → LightGray , MaxRecursion → 4], Boxed → False , Axes → None]
```

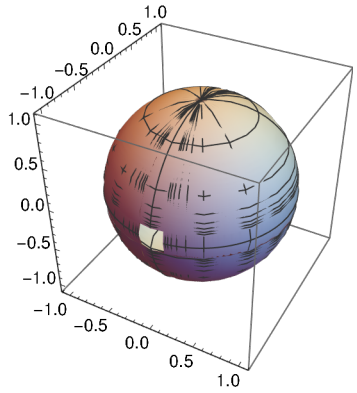
Out[ ]=



Note the parametric plot with these rational parameters misses a small square near {0,1,0}.

```
ParametricPlot3D [rationalSphere , {t, -15 , 15}, {s, -15 , 15},
 PlotStyle → LightGray , MaxRecursion → 4, ImageSize → Small]
```

Out[ ]=




We can derive parametric equations for the ellipsoid using the well known trigonometric equation, for example

```
ellipsoid1 = {4 Sin[u] Cos[v], 2 Sin[u] Sin[v], Cos[u]};
```

Or we may take our rationalSphere and apply the FLT with matrix

```
In[ ]:= PellipMat = {{4, 0, 0, 1}, {0, 2, 0, 2}, {0, 0, 1, 3}, {0, 0, 0, 1}};
        PellipMat // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 4 & 0 & 0 & 1 \\ 0 & 2 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

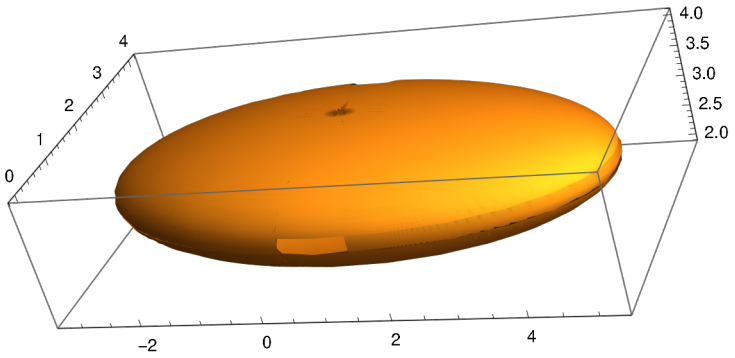
```
In[ ]:= pellip = Together[fltMD[rationalSphere , PellipMat]]
```

Out[ ]=

$$\left\{\frac{1+8\,s+s^2+t^2-8\,s\,t^2+s^2\,t^2}{(1+s^2)\times(1+t^2)},\ \frac{4\times(1+s^2\,t^2)}{(1+s^2)\times(1+t^2)},\ \frac{3+2\,t+3\,t^2}{1+t^2}\right\}$$

```
In[ ]:= ParametricPlot3D [pellip , {t, -15, 15}, {s, -15, 15}, MaxRecursion → 4, Mesh → None]
```

Out[ ]=




This also translates the center to {1,2,3}.

We can add a rotation to this.

```
In[ ]:= RPellipMat =
          TransformationMatrix [N[RotationTransform [2 Pi / 3, {10, 1, 0}, {0, 0, 1}]]].PellipMat
```

```
Out[ ]= {{3.94059 , 0.29703 , 0.0861727 , 1.45452}, {0.594059 , -0.970297 , -0.861727 , -2.54524},
         {-0.344691 , 1.72345 , -0.5 , 1.63728}, {0. , 0. , 0. , 1.}}
```

```
In[ ]:= rpellip = Together[fltMD[rationalSphere , RPellipMat]]
```

Out[ ]=

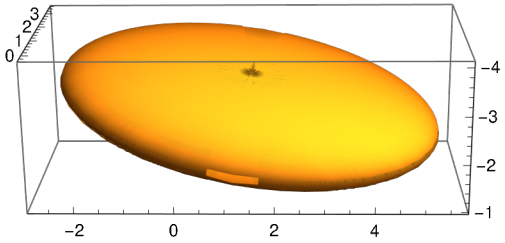
$$\left\{\frac{1}{(1.+s^2)\times(1.+t^2)}\,1.75155\times(1.+4.49954\,s+0.660839\,s^2+0.0983958\,t+0.0983958\,s^2\,t+0.660839\,t^2-4.49954\,s\,t^2+1.\,s^2\,t^2),\right.$$

$$-\frac{1}{(1.+s^2)\times(1.+t^2)}\,3.51553\times(1.-0.337963\,s+0.447995\,s^2+0.49024\,t+0.49024\,s^2\,t+0.447995\,t^2+0.337963\,s\,t^2+1.\,s^2\,t^2),$$

$$\left.\frac{1}{(1.+s^2)\times(1.+t^2)}\,3.36074\times(1.-0.205128\,s-0.025641\,s^2-0.297554\,t-0.297554\,s^2\,t-0.025641\,t^2+0.205128\,s\,t^2+1.\,s^2\,t^2)\right\}$$

In[ ]:=
```
ParametricPlot3D [rpellip , {t, -15, 15}, {s, -15, 15}, MaxRecursion → 4, Mesh → None]
```

Out[ ]=

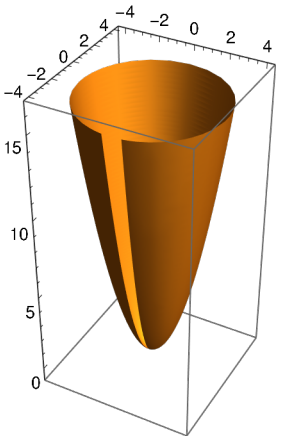



One can carefully have one parameter trigonometric while the other is a variable, for example the parabolic ellipsoid

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

In[ ]:=
```
ParametricPlot3D [parabell , {t, 0, 4}, {s, -15, 15}, MaxRecursion → 4, Mesh → None]
```

Out[ ]=




Unfortunately you may not use the same trick here for the hyperbolic ellipsoid that one uses in calcu - lus, that is

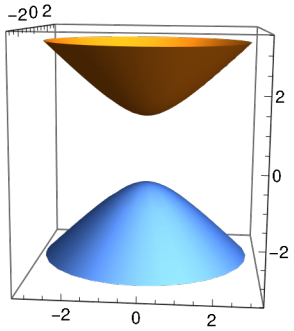
In[ ]:=
```
chyp = {u Cos[v], u Sin[v], Sqrt[u ^ 2 + 1]}
chym = {u Cos[v], u Sin[v], - Sqrt[u ^ 2 + 1]}
```

Out[ ]=
$$\left\{u\ \text{Cos}[v],\ u\ \text{Sin}[v],\ \sqrt{1+u^2}\right\}$$

Out[ ]=
$$\left\{u\ \text{Cos}[v],\ u\ \text{Sin}[v],\ -\sqrt{1+u^2}\right\}$$

In[ ]:=
```
ParametricPlot3D [{chyp , chym}, {u, 0, 3}, {v, -Pi, Pi}, Mesh → None]
```

Out[ ]=




because each gives only “one sheet” and is not a rational function of u.

However we saw in the last section transforming the ellipsoid by a projective FLT did give a hyperbolic ellipsoid.

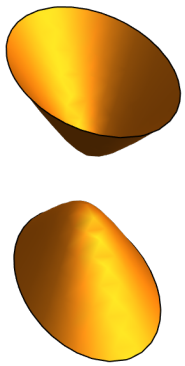
In[ ]:=
```
A = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 0, 1}, {1, 1, 1, 0}};
```

In[ ]:=
```
he = FLT3D[{x ^ 2 + y ^ 2 + z ^ 2 - 1}, A, {x, y, z}][[1]]
```

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

In[ ]:=
```
ContourPlot3D [he == 0, {x, -2, 2}, {y, -2, 2}, {z, -2, 2},
  Mesh → None , Axes → None , Boxed → False , MaxRecursion → 3]
```

Out[ ]=

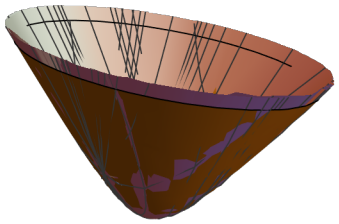


We can use this idea to get a rational parameterization of the hyperbolic ellipse.

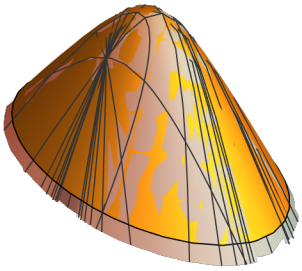
In[ ]:=
```
phe = Simplify[fltMD[rationalSphere , A]]
```

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

In[ ]:=
```
Show[ContourPlot3D [he == 0, {x, -2, 2}, {y, -2, 2}, {z, -2, 2},
  Mesh → None, Axes → None, Boxed → False, MaxRecursion → 3],
 ParametricPlot3D [phe, {s, -15, 15}, {t, -15, 15}, PlotStyle → LightGray ]]
```



Out[ ]=



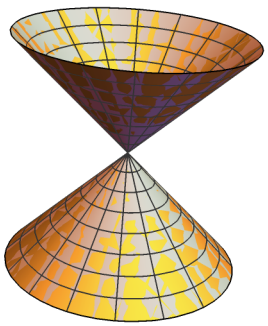
Note that this one parameterized function gives both “sheets”, another reason why this should not be called the hyperboloid of two sheets.

Finally we note that we can parameterize the cone by

```
pcone = {s Cos[t], s Sin[t], s};
```

In[182]:=
```
Show[ContourPlot3D [x ^ 2 + y ^ 2 == z ^ 2, {x, -3, 3},
  {y, -3, 3}, {z, -3, 3}, Mesh → None, ContourStyle → Opacity[.8]],
 ParametricPlot3D [pcone, {t, -Pi, Pi}, {s, -3, 3}, PlotStyle → LightGray],
 Axes → None, Boxed → False, ImageSize → Small]
```

Out[182]=



## 1.8.2 Other parametric surfaces via trigonometry

**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]};
```

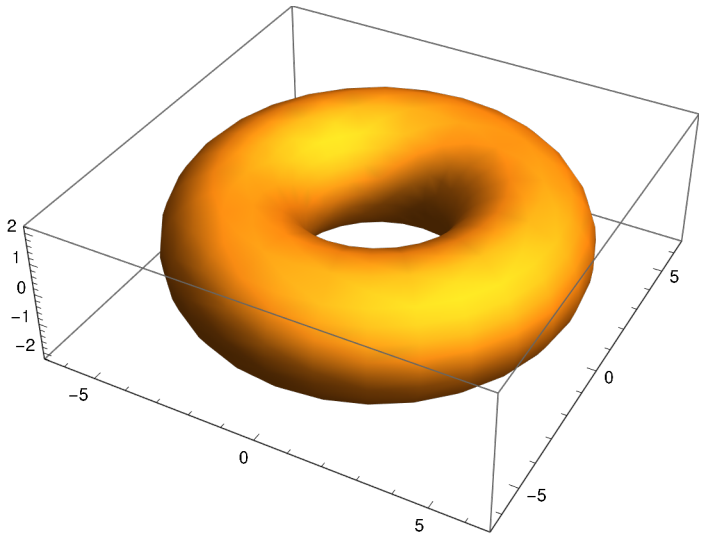
For large radius 4 and small radius 2

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[ ]=

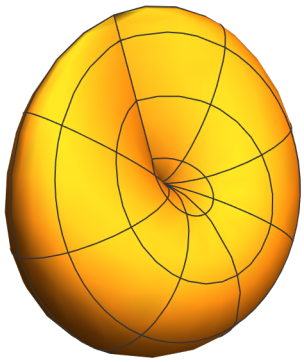



**The Crosscap**

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[ ]=



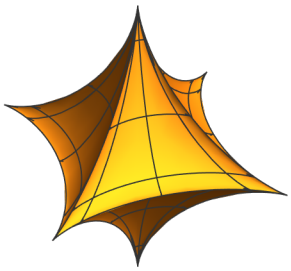
This is algebraic since elementary trig identities, eg. $\mathrm{Sin}[2\,u] = 2\,\mathrm{Sin}[u]\,\mathrm{Cos}[u]$, allow one to write these parameters in terms of the proxies for sin3 and cosine. Also the square of the proxies are again rational

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

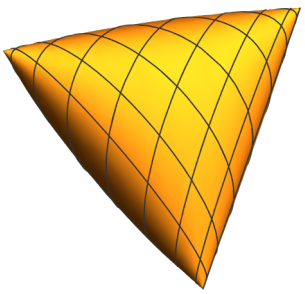
$$x^{2/3} + y^{2/3} + z^{2/3} = 1$$

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[ ]=



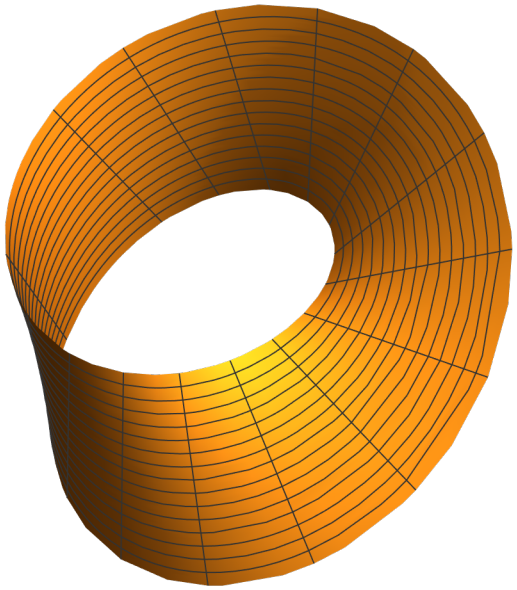
An example of a surface which is not algebraic is the Möbius Strip.

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

Out[ ]=
$$\left\{\left(1 + t \cos\left[\frac{u}{2}\right]\right) \cos[u], \left(1 + t \cos\left[\frac{u}{2}\right]\right) \sin[u], t \sin\left[\frac{u}{2}\right]\right\}$$

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 polynomially 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.

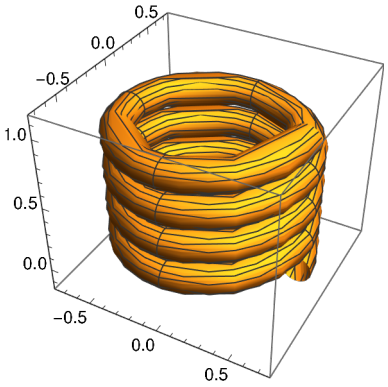
The example given has

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

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

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

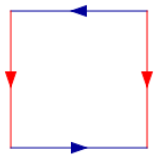
Out[ ]=




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.

## 1.8.3 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.

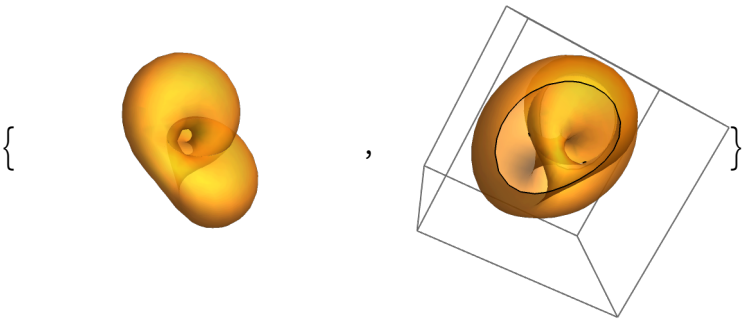


We take our exposition from *Wolfram Mathworld.* An implicit equation of a projection into 3-space is

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

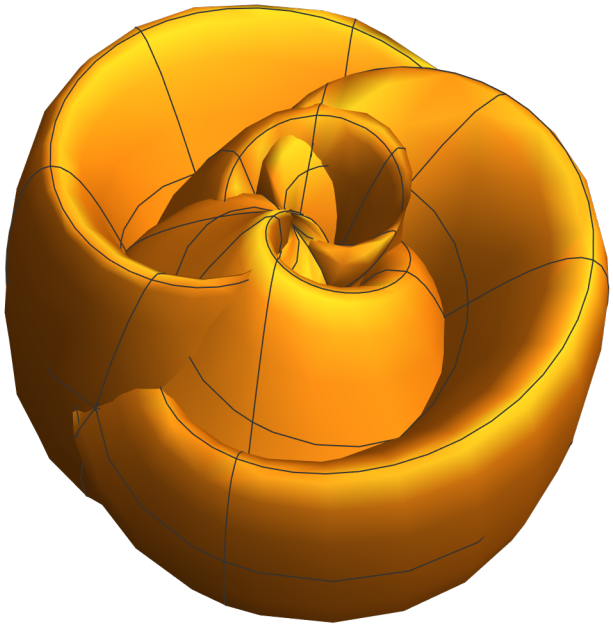
projecting causes self intersections. Here is an interesting trigonometric parameterization of an interpretation of this 4 dimensional surface.

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.

# 1.9 Lines on a Cubic Surface

In 1849 Arthur Cayley and George Salmon showed that every smooth cubic contains exactly 27 lines. Elsewhere I have written extensively about this topic, notably my article [*Ideals of Numeric Realizations of Configurations of Lines],* A variation of this article together with some additional information is available on my website. In this section and its notebook appendices I am giving a new take on this material.

In general, even if the cubic surface is a real surface, many of these lines may be complex, in fact the number of real lines can only be 3, 7, 15 or 27. For example the Fermat Surface of Section 1.5 and 1.6 contains, as we saw, 3 real lines and hence 24 complex lines. These lines are easy to write down by inspection using the pattern established for the three real lines. Let $\alpha,\beta$ be the two cube roots of -1 other than -1 itself, that is $\alpha$=.5-Sqrt[3]/2 $i$, $\beta$ = .5+Sqrt[3]/2$i$.

```
In[ ]:= α = .5 - Sqrt[3] / 2 I
        β = .5 + Sqrt[3] / 2 I
Out[ ]= 0.5 - 0.866025 i
Out[ ]= 0.5 + 0.866025 i
```

```
In[ ]:= α ^ 3
Out[ ]= -1. - 1.11022 × 10^-16 i
```

The three real lines are

```
In[ ]:= lf1 = {t, -t, -1};
        lf2 = {t, -1, -t};
        lf3 = {-1, t, -t};
```

By replacing the - 1' s, including the coefficient of $-t$, by $\alpha$, and or $\beta$ we can easily construct the remain - ing 24 lines, a few more will be listed below

```
In[ ]:= lf4 = {t, α t, -1};
        lf5 = {β, t, -t};
        lf6 = {α, t, β t};
```

Note, for example

```
In[ ]:= (x ^ 3 + y ^ 3 + z ^ 3 + 1) /. Thread[{x, y, z} → lf6]
Out[ ]= (2.22045 × 10^-16 - 1.11022 × 10^-16 i) + (2.22045 × 10^-16 + 1.11022 × 10^-16 i) t^3
```

The reader can write down the rest if they choose to. I will note that in my **GlobalFunctions.nb** that there is a function called pLineIntersectionMD which finds the intersection of two parametric lines in any dimensional space. This will be discussed with code in section 1.9.3. It does specifically work for all lines including pairs of lines with possible infinite or complex intersections. The empty set is returned if the lines are skew.

```
In[ ]:= pLineIntersectionMD [lf1, lf6, t, {x, y, z}, dTol]
Out[ ]= {0.5 - 0.866025 i, -0.5 + 0.866025 i, -1. + 1.17961 × 10^-16 i}
```

## 1.9.1 The double Six configuration

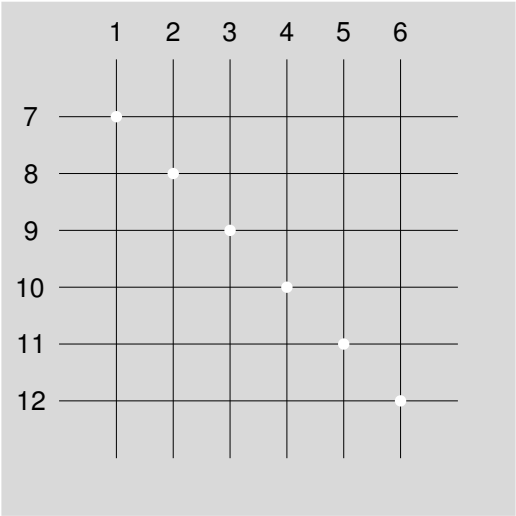
In H . S . M . Coxeter' s review of Volume II of Ludwig Schläfli's collected works he says that one paper

> . . is modestly entitled "An attempt to determine the 27 lines upon a surface of the third order, and to divide such surfaces into species in reference to the reality of the lines upon the surface ." The existence of 27 such lines had already been discovered by Cayley and Salmon, but this paper of 1856 gives the first complete description of this configuration . .

The key to Schläfli's analysis is his discovery of 12 line sub-configurations of the 27 lines, this configura - tion called a *double 6* . From these one may extract the remaining 15 lines easily.

A double 6 configuration consists of two sets of 6 mutually skew lines such that a line in the first set intersects 5 lines of the second set, we number the lines in each set so that the $k^{th}$ line in the first set is skew from the $k^{th}$ line of the second set but intersects all the other lines of the second set. We can draw this where a blank area indicates no intersection.

Out[ ]=




In a double 6 there are 15 double 2 configurations, two lines from each skew set which do not intersect the other set, for example L1, L2 ,L7, L8 is a double 2. For each double 2 there is a unique line which intersects all 4 lines. Since a line which meets a cubic surface in 4 points, counting multiplicities is in the cubic surface the cubic that contains the double 6 also contains these 15 lines.

## 1.9.2 The theory

[Hilbert and Cohn-Vossen] show in their book how to construct a double 6 configuration in $\mathbb{R}^3$ making 6 somewhat arbitrary, or if you prefer random, choices. I gave an example of this in my *Configuration* paper mentioned above. Given a double 6 there is an explicit construction of 15 additional lines which meet the double 4 in 4 points. The theorem is that for any particular double 6 there is a unique smooth

cubic surface containing this double 6. It then must also contain the other 15 lines which meet the double 4 in 4 points for a total of 27 lines.

Conversely every smooth cubic contains 27 lines and within these 27 lines there are double 6 configura - tions determining all of these lines.

I will construct a double 6 using the Hilbert Cohn-Vossen method in appendix A. Here is their method which I will modify slightly.

In[ ]:=

| line | construction |
|---|---|
| 1 | random line |
| 8 | random line meeting line 1 |
| 9 | random line meeting line 1 |
| 10 | random line meeting line 1 |
| 11 | random line meeting line 1 |
| 6 | other line meeting 8,9,10,11 |
| 12 | random line meeting line 1 |
| 5 | other line meeting 8,9,10,12 |
| 4 | other line meeting 8,9,11,12 |
| 3 | other line meeting 8,10,11,12 |
| 2 | other line meeting 9,10,11,12 |
| 7 | other line meeting 2,3,4,5 |

In the next subsection we discuss some of the problems that must be solved with the tools to solve them. The major work will be in the notebook appendices.

## 1.9.3 The Problems

The appendices depend on being able to solve certain problems, particularly problem E below which is needed to find lines 6, 5,4,3 2 and 7. I describe here, through examples, how to use a combination of built-in functions and my global functions to do this.

**A . Find the two tangent lines through a point on a hyperboloid.** Let the hyperboloid and nice integer point be

In[ ]:=
```
h1Eq = -y - x y - x z + y z;
q1 = {-1, -1, 2};
h1Eq /. Thread[{x, y, z} → q1]
```

Out[ ]= 0

We first find the tangent plane at this point .

In[ ]:=
```
tP = tangentPlaneNS [h1Eq, q1, {x, y, z}]
```

Out[ ]= $-1 - x + 2 \times (1 + y)$

The two lines are the intersections of the tangent plane with the hyperboloid . In this nice exact case it is easy

```
In[ ]:= Solve[h1Eq == 0 && tP == 0, {x, y, z}]
```

Solve : Equations may not give solutions for all "solve " variables .

```
Out[ ]= {{x → 1 + 2 y, z → -2 y}, {x → -1, y → -1}}
```

We can now just write down either the implicit equations or parametric formula for these lines.

```
In[ ]:= l1eq = {1 + 2 y - x, -2 y - z};
        l1p = {1 + 2 t, t, -2 t};
```

```
In[ ]:= l2eq = {x + 1, y + 1};
        l2p = {-1, -1, t};
```

Note for line 1, line 2 is similar, we can verify these formulas

```
In[ ]:= l1eq /. Thread[{x, y, z} → l1p]
        Simplify[h1Eq /. Thread[{x, y, z} → l1p]]
```

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

```
Out[ ]= 0
```

Unfortunately if these are given numerically **Solve** may not work. Consider a different point.

```
In[ ]:= q2 = {-0.5820528096134947` , -0.41794719038650535` , -1.0644355432484727` };
        h1Eq /. Thread[{x, y, z} → q2]
```

```
Out[ ]= -3.37508 × 10^-14
```

```
In[ ]:= tP2 = Expand[tangentPlaneNS [h1Eq , q2, {x, y, z}]]
```

```
Out[ ]= 0.417947 + 1.48238 x - 1.48238 y + 0.164106 z
```

The first solution from **Solve** is

```
In[ ]:= Solve[h1Eq == 0 && tP2 == 0, {x, y, z}][[1]]
```

Solve : Solve was unable to solve the system with inexact coefficients . The answer was obtained by solving a corresponding exact system and numericizing the result .

Solve : Equations may not give solutions for all "solve " variables .

```
Out[ ]= {x → 1.88744 × 10^-23 × (-7.4689 × 10^21 + 5.59143 × 10^22 y -
            6332.47 Sqrt[1.39113 × 10^36 + 6.65696 × 10^36 y + 7.96388 × 10^36 y^2]),
          z → 1.36396 × 10^-21 × (-9.33613 × 10^20 - 3.6658 × 10^20 y +
            791.559 Sqrt[1.39113 × 10^36 + 6.65696 × 10^36 y + 7.96388 × 10^36 y^2])}
```

This solution is not satisfactory . The technique is to find two points other than q2 in the intersection and, by the theory, we can then find the lines from q2 to these points.

```
In[ ]:= sol2 = NSolveValues [{h1Eq , tP2}, {x, y, z}]
```

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

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

```
Out[ ]= {{-0.444331 , -0.226149 , -0.575961}, {-0.792106 , -0.568778 , -0.529467}}
```

The first line is

```
In[ ]:= l1eq = lineMD[q2, sol2[[1]], {x, y, z}]
Out[ ]= {-0.142566 - 0.505657 x - 0.733816 y + 0.430697 z,
  0.221125 + 0.784292 x - 0.57961 y + 0.00645667 z}
```

Now we can find the first line using `Solve`

```
In[ ]:= sol2b = Solve[l1eq == 0, {x, y, z}]
```

Solve : Equations may not give solutions for all "solve " variables .

```
Out[ ]= {{y → 0.392647 + 1.39265 x, z → 1. + 3.54682 x}}
```

The solution is given using the parameter x, replacing this by t we have

```
In[ ]:= l1p = {x, y, z} /. sol2b[[1]] /. {x → t}
Out[ ]= {t, 0.392647 + 1.39265 t, 1. + 3.54682 t}
```

Checking

```
In[ ]:= Simplify[l1eq /. Thread[{x, y, z} → l1p]]
        Simplify[h1Eq /. Thread[{x, y, z} → l1p]]
Out[ ]= {0., 2.77556 × 10^-17}
Out[ ]= 5.64271 × 10^-13 + 2.04947 × 10^-12 t + 1.75637 × 10^-12 t^2
```

which is good to approximately our default tolerance .

**B . Going from parametric equation of line to implicit equations** . In principle one can use the general implicitization method as in Section 1.4 but with lines it is easiest to find two points and use the Global Function `lineMD`. This is automated by Global Function `pl2eqMD` which handles `parametric lines in` $\mathbb{R}^n$ for any $n$.

It doesn't need to be automated, for example consider

```
In[ ]:= line1 = {t, 0.39264678170294964` + 1.3926467817030561` t,
          1.000000000001437` + 3.5468182768858614` t}
Out[ ]= {t, 0.392647 + 1.39265 t, 1. + 3.54682 t}
```

We calculate

```
In[ ]:= p = line1 /. {t → 0}
        q = line1 /. {t → 4}
Out[ ]= {0, 0.392647 , 1.}
Out[ ]= {4, 5.96323 , 15.1873}

In[ ]:= line1Eq = lineMD[p, q, {x, y, z}]
Out[ ]= {-0.20001 - 0.709398 x - 0.536696 y + 0.410741 z,
         -0.170932 - 0.606265 x + 0.765762 y - 0.129742 z}
```

But sometimes to get more accuracy or if the 2 points are rational we would like an equation system with rational coefficients . But lineMD returns floating point numbers as do the methods in section 1.3 . A simple routine specifically for lines in $\mathbb{R}^3$ is

```
In[ ]:= ratLine3D[p_, q_] := Module[{form, formp, formq, sol},
          form = {x - a y + b, x - c z + d};
          formp = form /. Thread[{x, y, z} → p];
          formq = form /. Thread[{x, y, z} → q];
          sol = Solve[formp == 0 && formq == 0][[1]];
          form /. sol]
```

Note that it is assumed that the variables are x, y, z and that x is a parameter, meaning the two points p, q have distinct first component. If not rename the variables, run then name them back again. It is somewhat surprising that the equation solved appears to be underdetermined, but Solve apparently needs the extra variable. Anyway we only need one solution so if the Solve returns several we are only using the first. Here is an example:

```
In[ ]:= p = {-14/15, -17/15, 0};
        q = {1/13, -11/13, 11/13};

In[ ]:= l = ratLine3D[p, q]
Out[ ]= {-171/56 + x - 197 y/56, 14/15 + x - 197 z/165}
```

Test: Note that r1 p + r2 q will be in the line through p,q for any r1+r2=1

```
In[ ]:= r = 3 / 7 p + 4 / 7 q
Out[ ]= {-162/455, -63/65, 44/91}

In[ ]:= l /. Thread[{x, y, z} → r]
Out[ ]= {0, 0}
```

**C . Find intersection point or determine parallel or skew given two parametric lines .** The reader is reminded that we are actually working in projective 3 space but seeing only affine space. Two lines are

parallel if they have a common infinite point. Skew means they do not intersect or are parallel. Fortunately we have a very good Global Function to tell the difference. I have mentioned it before but here is the code based directly on the SVD.

```
In[ ]:= nullspace[M_, tol_] :=
         Take[SingularValueDecomposition [N[M]][[3]], All, -(Dimensions[M][[2]] - matrixrank[M, tol])]

        pLineIntersectionMD [L1_, L2_, t_, X_, tol_] :=
         Module[{n, cr1, cr2, p1, p2, v1, v2, eq1, eq2, S, r, ans},
          n = Length[X];
          If[Length[L1] ≠ n, Echo["Line 1 error"]; Abort[]];
          If[Length[L2] ≠ n, Echo["Line 2 error"]; Abort[]];
          p1 = Chop[L1 /. {t → 0}];
          v1 = Append[Chop[(L1 - p1) /. {t → 1}], 0];
          eq1 = lineMD[p1, v1, X];
          p2 = Chop[L2 /. {t → 0}];
          v2 = Append[Chop[(L2 - p2) /. {t → 1}], 0];
          eq2 = lineMD[p2, v2, X];
          S = sylvesterMD [Join[eq1, eq2], 1, X];
          r = matrixrank[S, tol];
          If[r < n, Return[{0}]];
          If[r > n, Return[{}]];
          ans = Flatten[nullspace[S, tol]];
          If[Abs[ans[[1]] < tol, RotateLeft[Chop[ans, tol], 1], Take[ans / ans[[1]], -n]]
         ]
```

To confirm intersection we should use a tight tolerance, but to confirm skewness we should use a loose one. Here are two random parallel lines

```
In[ ]:= rline1 = {-1.284743961295125` + 1.7850221750544781` t,
          -1.8513906749735787` + 0.32363757592140274` t,
          -1.7705832745415062` - 0.49925464276626474` t}
Out[ ]= {-1.28474 + 1.78502 t, -1.85139 + 0.323638 t, -1.77058 - 0.499255 t}

In[ ]:= rline2 = {-3.8470503573307893` + 1.3999119717968946` t,
          -3.2811667316024042` + 0.253814279389482` t,
          1.5989379697539752` - 0.39154278369810475` t}
Out[ ]= {-3.84705 + 1.39991 t, -3.28117 + 0.253814 t, 1.59894 - 0.391543 t}

In[ ]:= pLineIntersectionMD [rline1, rline2, t, {x, y, z}, dTol]
Out[ ]= {-0.948688, -0.172004, 0.26534, 0}
```

Note that the function returns a list of length 4 with the last component 0, this means infinite point. Now let

In[ ]:=

In[ ]:=
```
rline3 = {-1.1577650571599911` + 1.609386049766386` t,
   -1.66840669830856` + 0.2899071921064588` t,
   -1.5955859749592347` - 0.4488387468161354` t}
```

Out[ ]= {-1.15777 + 1.60939 t, -1.66841 + 0.289907 t, -1.59559 - 0.448839 t}

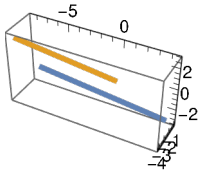
In[ ]:=
```
pLineIntersectionMD [rline1 , rline3 , t, {x, y, z}, dTol]
```

Out[ ]= {}

Consider

In[ ]:=
```
ParametricPlot3D [{rline1 , rline2}, {t, -3, 3}, ImageSize → Tiny]
```

Out[ ]=




It perhaps looks like these are skew but note

In[ ]:=
```
pLineIntersectionMD [rline1 , rline3 , t, {x, y, z}, .003]
```

Out[ ]= {0.948813 , 0.171104 , -0.265476 , 0}

So these lines are parallel meeting in an infinite point. For our later work parallel lines are NOT skew.

A nice property of this function is that if one only wants to know whether 2 lines meet one can use

```
Length[pLineIntersectionMD [line1 , line2 , t, {x, y, z}, tol]]
```

If the result is 0 the lines are skew, if 1 the lines are equal, 3 means an affine intersection and 4 means an infinite intersection, i.e. parallel. We will use this heavily in later subsections.

**D . Finding hyperboloid generated by 3 skew lines .** We have done this in Section 7 but so this Section can stand alone we repeat with 3 parametric lines.

In[ ]:=
```
rline4 = RandomReal [{-3, 3}, {3, 2}].{1, t}
rline5 = RandomReal [{-3, 3}, {3, 2}].{1, t}
```

Out[ ]= {1.64127 + 1.98068 t, -2.48105 - 0.466556 t, 0.416791 + 1.84621 t}

Out[ ]= {0.52162 - 1.46426 t, 0.208229 - 1.25196 t, -2.3118 + 0.578546 t}

We will find the hyperloid generated by lines rl1, rl4, rl5. First we check skewness

In[ ]:=
```
{pLineIntersectionMD [rl1 , rl4 , t, {x, y, z}, .001],
  pLineIntersectionMD [rl1 , rl5 , t, {x, y, z}, .001],
  pLineIntersectionMD [rl4 , rl5 , t, {x, y, z}, .001]}
```

» Line 1 error

Out[ ]= $Aborted

Next we find implicit equations

```
In[ ]:= rl1eq = pl2eqMD[rline1, t, {x, y, z}]
Out[ ]= {0.124503 + 0.301341 x - 0.72363 y + 0.608319 z,
  0.927707 - 0.00411915 x + 0.319782 y + 0.192568 z}

In[ ]:= rl4eq = pl2eqMD[rline4, t, {x, y, z}]
Out[ ]= {0.00276285 - 0.682579 x - 0.341924 y + 0.645884 z,
  0.919497 - 0.0454673 x + 0.364185 y + 0.140812 z}

In[ ]:= rl5eq = pl2eqMD[rline5, t, {x, y, z}]
Out[ ]= {0.381565 + 0.633352 x - 0.62443 y + 0.251713 z,
  0.815507 - 0.218984 x + 0.413509 y + 0.340594 z}
```

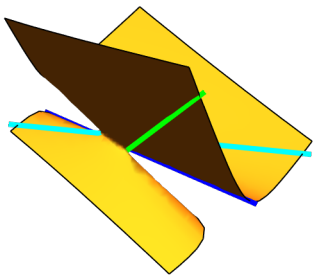
Then we find Sylvester matrices, $m = 2$ is sufficient for this, although if we actually want equation of the configuration of these three lines we should use at least $m = 4$. Just finding the hyperboloid loses the information about what lines we used which may be important later.

```
In[ ]:= syl1 = sylvesterMD[rl1eq, 2, {x, y, z}];
        syl4 = sylvesterMD[rl4eq, 2, {x, y, z}];
        syl5 = sylvesterMD[rl5eq, 2, {x, y, z}];
        hp2 =
          First[Chop[vectorSpaceIntersection3 [syl1, syl4, syl5, dTol], dTol].mExpsMD[2, {x, y, z}]]
Out[ ]= 0.794171 + 0.204124 x - 0.00394934 x^2 + 0.27198 y + 0.0884639 x y -
  0.0239499 y^2 + 0.469243 z + 0.0247282 x z + 0.150685 y z + 0.0416093 z^2
```

To look at this hyperboloid and the lines

```
In[ ]:= Show[ContourPlot3D [hp2 == 0, {x, -5, 5}, {y, -5, 5}, {z, -5, 5}, Mesh → None],
          ParametricPlot3D [{rline1, rline4, rline5}, {t, -5, 5}, PlotStyle → {Blue, Green, Cyan}],
          Axes → False, Boxed → False, ImageSize → Small]
```

Out[ ]=

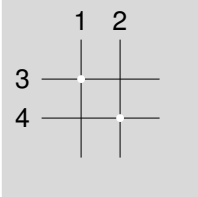


**E. Finding two lines intersecting 4 skew lines the last intersecting the hyperboloid generated by the first 3 in two points.** Actually Hilbert stated this more generally, but if a line not in, or tangent to, a hyperboloid intersects a hyperboloid in one point then since the equation of the hyperboloid has degree 2 there are exactly 2, possibly infinite, points of intersection of the line and the hyperboloid.

Using the above methods one simply notes that these two lines are the lines in the opposite ruling of the first 3 lines at the points of intersection. In the construction of the double 6 one of the lines is already known so one merely needs to construct the two tangent lines at the other intersection point, one will be skew to the first 3 lines and the other will intersect the first 3 lines so one test using **pLineIntersectionMD** is sufficient. So it is really not necessary to give an example.

A *double 2* is a configuration of 4 lines with the following diagram :

Out[ ]=




**F. Given a double 2 find a line which meets all 4 lines.** Note that intersecting lines 1,4 define a plane as do intersecting lines 2,3. In projective 3-space space any two distinct planes meet in a unique line. Rather than go through the procedure of problem D, we can assume we know the intersection points of 1,4 and 2,3 and one more point on each line. Then the equations of the planes come from **linearSetMD**, each plane with a single equation. The intersecting line is the line with these 2 equations. As in **A**. if one needs parametric equations one can use Solve.

I give an example below in 1.9.5.

## 1.9.4 The double 6 construction

I modify the Hilbert Cohn-Vossen method by starting out with the hyperboloid given both parametrically and later by an implicit quadric equation in Section 1.3. This way I can find lots of rational points and lines in the construction. Lines L1, L8, L9 and L10 come from this hyperboloid. Further lines L5, L6 will then also be in this paraboloid and L11 and L12 meet the hyperboloid in rational points so will themselves be rational.

Recall the hyperboloid and its equation are given by

In[ ]:=
```
hyp1 = {(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)};
hypEq = 1 - x^2 - y^2 + z^2;
```

Here are the lines given by their parametric form.

In[ ]:=
```
L1 = {t, 5/3 + (4 t)/3, 4/3 + (5 t)/3};
```

In[ ]:=
```
L2 = {t, 1.10873690400994` - 0.4642368931192767` t,
  -0.4642368931190869` + 1.669047069329676` t}
```

Out[ ]=
```
{t, 1.10874 - 0.464237 t, -0.464237 + 1.66905 t}
```

In[ ]:=
```
L3 = {t, 1.125206152628268` - 0.5076671846982648` t,
   -0.3081725820785607` + 1.5241953578676644` t};
```

In[ ]:=
```
L4 = {t, 0.9721721581433124` - 0.4260900032234079` t,
  -0.11264557902259985` + 1.3711014253079723` t}
```

Out[ ]= {t, 0.972172 - 0.42609 t, -0.112646 + 1.3711 t}

In[ ]:=
```
L5 = {t, 29/20 - 21 t/20, -21/20 + 29 t/20};
```

In[ ]:=
```
L6 = {t, 1, t};
```

In[ ]:=
```
L7 = {t, 1.661032057842025` - 0.9952722110334632` t,
  -0.40924299170135053` + 1.6161700818709201` t}
```

Out[ ]= {t, 1.66103 - 0.995272 t, -0.409243 + 1.61617 t}

In[ ]:=
```
L8 = {t, 13/5 + 12 t/5, -12/5 - 13 t/5};
```

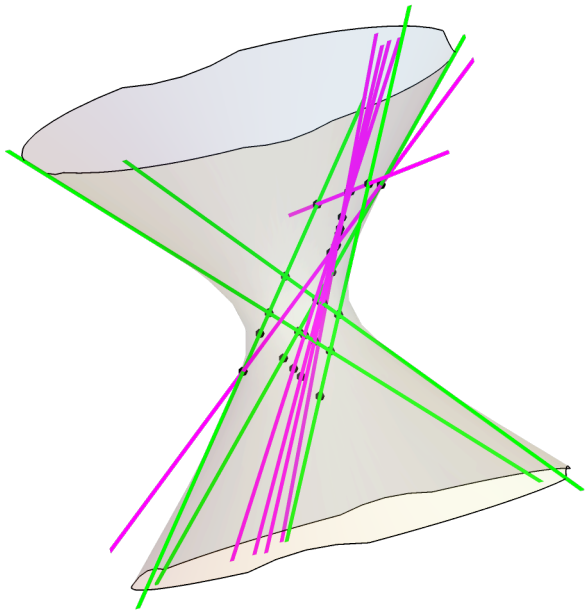
In[ ]:=
```
L9 = {t, 17/15 + 8 t/15, -8/15 - 17 t/15};
```

In[ ]:=
```
L10 = {t, 17/15 - 8 t/15, 8/15 - 17 t/15};
```

In[ ]:=
```
L11 = {t, 5/13 + 4 t/13, 4/13 + 11 t/13};
```

In[ ]:=
```
L12 = {t, 599/180 - 179 t/90, 409/180 - 19 t/90};
```

Here is the plot with the intersection points



The work is contained in Appendix 1.9A which will be available only in Mathematica Notebook form. This appendix is designed to be evaluated rather than read. In the Mathematica pull down **Evaluation** menu choose Evaluate Notebook It will do the computation and check the answers above. The reader may wish to skim the code in these notebooks but unless you want to make your own double 6 it makes poor reading material.

## 1.9.5 Additional Lines

As mentioned above there are 15 additional lines that will intersect this double 6 in 4 points, hence will in any naive cubic surface containing these lines. The construction is outlined in Problem F, here is an example. The reader who wants all 15 must work them out themselves, they are not included in the Appendix A.

We consider the line from the double 2 consisting of L1, L2, L7 and L8. First we find the planes containing L7, L2 and L1, L8.

Note these lines have the following implicit equations

```
In[ ]:= L1eq = ratLine3D[L1 /. {t → 0}, L1 /. {t → 4}]
```

$$\text{Out[ ]= } \left\{\frac{5}{4}+x-\frac{3\,y}{4},\ \frac{4}{5}+x-\frac{3\,z}{5}\right\}$$

```
In[ ]:= L2eq = ratLine3D[L2 /. {t → 0}, L2 /. {t → 4}]
Out[ ]= {-2.3883 + x + 2.15407 y, -0.278145 + x - 0.599144 z}

In[ ]:= L7eq = ratLine3D[L7 /. {t → 0}, L7 /. {t → 4}]
Out[ ]= {-1.66892 + x + 1.00475 y, -0.253218 + x - 0.618747 z}
```

```
In[ ]:= NSolve[Join[L2eq, L7eq]]
Out[ ]= {{x → 1.04003, y → 0.625914, z → 1.27163}}

In[ ]:= L8eq = ratLine3D[L8 /. {t → 0}, L8 /. {t → 4}]
Out[ ]= {13/12 + x - 5 y/12, 12/13 + x + 5 z/13}

In[ ]:= syl7 = sylvesterMD[L7eq, 1, {x, y, z}];
        syl2 = sylvesterMD[L2eq, 1, {x, y, z}];

In[ ]:= int72 = vectorSpaceIntersection[syl7, syl2, dTol];
        plane72 = int72[[1]].mExpsMD[1, {x, y, z}]
Out[ ]= 0.277687 - 0.828887 x - 0.0481178 y + 0.483239 z
```

Likewise

```
In[ ]:= syl1 = sylvesterMD[L1eq, 1, {x, y, z}];
        syl8 = sylvesterMD[L8eq, 1, {x, y, z}];
        int18 = vectorSpaceIntersection[syl1, syl8, dTol];
        plane81 = int18[[1]].mExpsMD[1, {x, y, z}]
Out[ ]= -0.701646 - 0.613941 x + 0.350823 y + 0.0877058 z

In[ ]:= Therefore

In[ ]:= L13 = First[SolveValues[plane72 == 0 && plane81 == 0, {x, y, z}] /. {x → t}]
        SolveValues : Equations may not give solutions for all "solve" variables.
Out[ ]= {t, 2.09159 + 1.28909 t, -0.366371 + 1.84363 t}
```

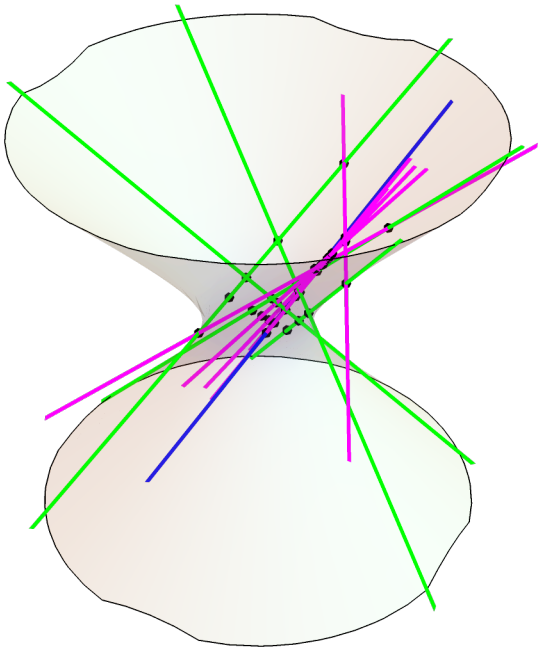
Checking :

```
In[ ]:= p131 = pLineIntersectionMD[L13, L1, t, {x, y, z}, dTol]
Out[ ]= {9.60473, 14.473, 17.3412}

In[ ]:= p132 = pLineIntersectionMD[L13, L2, t, {x, y, z}, dTol]
Out[ ]= {-0.560566, 1.36897, -1.39985}

In[ ]:= p137 = pLineIntersectionMD[L13, L7, t, {x, y, z}, dTol]
Out[ ]= {-0.188482, 1.84862, -0.713861}

In[ ]:= p138 = pLineIntersectionMD[L13, L8, t, {x, y, z}, dTol]
Out[ ]= {-0.45765, 1.50164, -1.21011}
```

In Appendix A we calculate a lot of points called **D6Points** which will not listed here

```
In[ ]:= Show[ContourPlot3D [hypEq == 0, {x, -4, 4}, {y, -4, 4}, {z, -4, 4},
          ContourStyle → Directive[LightGreen , Opacity[.2]], Mesh → None],
        ParametricPlot3D [{L1, L9, L10, L5, L6}, {t, -20, 20}, PlotStyle → Green , PlotRange → 5],
        ParametricPlot3D [{L2, L3, L4, L7, L11, L12}, {t, -5, 5}, PlotStyle → Magenta],
        Graphics3D [{Black , PointSize[.015], Point[D6Points], Point[{p131 , p132 , p137 , p138}]}],
        ParametricPlot3D [L13, {t, -10, 10}, PlotStyle → Blue], Boxed → False , Axes → False]
```

Out[ ]=



The blue line is L13, one intersection point is outside of the plot range. Again the green lines lie in the original hyperboloid, shown as background, but the magenta and blue lines do not.

## 1.9.6 The Implicit Cubic

We can proceed as in Section 4, the torus, to find the equation of a cubic containing the double 6 obtained in subsection 4. It is important to note that we are aiming to find the equations of a reducible curve which is a union of the lines. We know from the *Space Curve Book* that these are generally not naive curves and will have more than two equations. For this reason we go one at a time and use a higher degree in the calculation. From past experience we can surmise that degree 5 will be sufficient, initially even degree 4 may work. But in each step we are adding to the curve so we want to avoid, say, using the equation of the hyperboloid alone containing many of the lines because this hyperboloid also has many points that will not be in the final cubic. We may at some point see the equation of the hyperboloid but with additional equations removing these unwanted points.

We will see in our calculation a new idea, at least to me, that we do not need to use all the lines in the double 6. Since we saw that half the lines in the double 6 were determined by the earlier lines the other

lines already exist in any cubic equation in the system. In fact when we have made all the choices allowed we see that there is a unique cubic which continues through the rest of the construct if we choose to continue. Once we have a unique cubic at this point we are actually done. This will happen once we have lines L1, L8, L9, L10, L11 and L12. Although Hilbert's construction puts L6 before choosing L12 I will show that adding L6 was unnecessary to get the cubic equation since it was already in the cubics at the L11 step.

So actually we have a new, to me, theorem.

*Given a line in 3 space and 5 skew lines intersecting that line, the intersections necessarily are distinct and of multiplicity 1 due to the skewness, there is a unique cubic containing these lines as well as the 21 other lines constructed from these as in subsections 4 and 5.*

Again the actual calculations are in a notebook Appendix, Appendix B. This one is more readable than Appendix A but still should be evaluated as a notebook to see that it works. The result is that the cubic

The cubic is

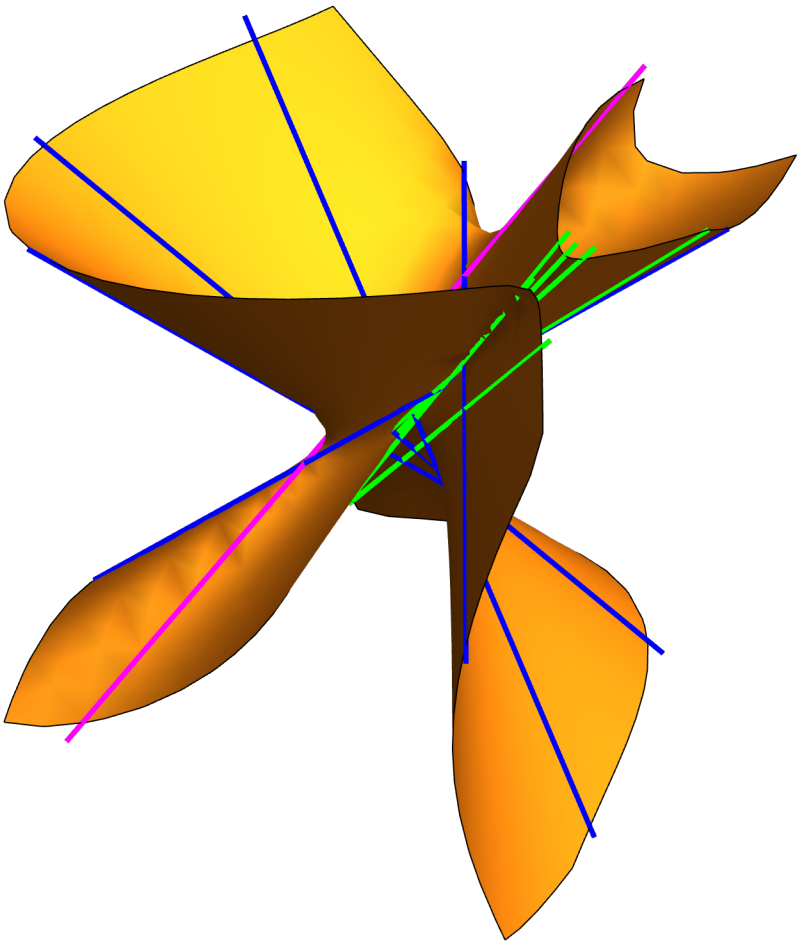
```
In[ ]:= f = -1.9593043005607316` - 3.0142672735884486` x +
   0.7465860452458656` x^2 + 2.1480399483637935` x^3 + 5.299476866625` y +
   2.263740743254375` x y - 1.014539031180315` x^2 y - 4.2498131666479475` y^2 +
   0.6950879815195592` x y^2 + 0.9096406005836525` y^3 + 2.217338134382248` z +
   0.46198790678698215` x z - 1.2587132278003588` x^2 z - 1.8202323527388888` y z -
   0.48046742305836376` x y z - 0.34166723282911526` y^2 z + 0.12195121951197414` z^2 -
   1.8893267205635906` x z^2 + 0.16448128269455703` y z^2 + 1.` z^3
```

```
Out[ ]= -1.9593 - 3.01427 x + 0.746586 x^2 + 2.14804 x^3 + 5.29948 y + 2.26374 x y - 1.01454 x^2 y -
  4.24981 y^2 + 0.695088 x y^2 + 0.909641 y^3 + 2.21734 z + 0.461988 x z - 1.25871 x^2 z -
  1.82023 y z - 0.480467 x y z - 0.341667 y^2 z + 0.121951 z^2 - 1.88933 x z^2 + 0.164481 y z^2 + 1. z^3
```

A plot is

In[ ]:=
```
Labeled[Show[
  ContourPlot3D[f == 0, {x, -5, 5}, {y, -5, 5}, {z, -5, 5}, Mesh → None, MaxRecursion → 5],
  ParametricPlot3D[L1, {t, -5, 5}, PlotStyle → Directive[Thick, Magenta]],
  ParametricPlot3D[{L8, L9, L10, L11, L12}, {t, -5, 5}, PlotStyle → Blue],
  ParametricPlot3D[{L7, L6, L5, L4, L3, L2}, {t, -5, 5}, PlotStyle → Green],
  Boxed → False, Axes → False, ImageSize → Large], "Double 6 on cubic f"]
```

Out[ ]=



Double 6 on cubic f

## 1.9.7 Finding lines on a given smooth cubic, first example.

In this subsection I go the opposite direction . I start with a smooth cubic surface and try to find the 27 lines. Based on the previous work one might think of looking for one line and then looking for 5 skew lines intersecting this line. From there I can find the other 21 lines using the previous techniques.

It actually turns out that it is easier to try to find all 27 lines at once. The trick is that for a parametric

line with parametric function F to lie on the surface $f = 0$ we simply need

```
f /. Thread[{x, y, z} → F] == 0
```

Letting $F$ be a generic curve it is easy to set up the equation which NSolve can solve. Given previous examples most lines do not have a constant first component. So we find these lines first

```
In[ ]:= F1 = {t, a1 + b1 t, a2 + b2 t}
Out[ ]= {t, a1 + b1 t, a2 + b2 t}
```

We first try an easy equation .

```
In[ ]:= cubic1 = 16 * x^3 + 16 * y^3 - 31 * z^3 + 24 * x^2 * z -
          48 * x^2 * y - 48 * x * y^2 + 24 * y^2 * z - 93.5307 * z^2 - 72 * z;
```

Our main equation is

```
In[ ]:= mainEq = Collect[Expand[cubic1 /. Thread[{x, y, z} → F1]], t]
Out[ ]= 16 a1^3 - 72 a2 + 24 a1^2 a2 - 93.5307 a2^2 - 31 a2^3 +
        (-48 a1^2 + 48 a1^2 b1 + 48 a1 a2 b1 - 72 b2 + 24 a1^2 b2 - 187.061 a2 b2 - 93 a2^2 b2) t +
        (-48 a1 + 24 a2 - 96 a1 b1 + 48 a1 b1^2 + 24 a2 b1^2 + 48 a1 b1 b2 - 93.5307 b2^2 - 93 a2 b2^2) t^2 +
        (16 - 48 b1 - 48 b1^2 + 16 b1^3 + 24 b2 + 24 b1^2 b2 - 31 b2^3) t^3
```

We want this to be essentially zero for all t. So the coefficients of $t^k$ must be zero. Let

```
In[ ]:= Clear[a1, a2, b1, b2]
```

Now just solve this non-linear system of 4 equations in 4 unknowns

```
In[ ]:= cf0 = 16 a1^3 - 72 a2 + 24 a1^2 a2 - 93.5307` a2^2 - 31 a2^3;
        cf1 = -48 a1^2 + 48 a1^2 b1 + 48 a1 a2 b1 - 72 b2 + 24 a1^2 b2 - 187.0614` a2 b2 - 93 a2^2 b2;
        cf2 = -48 a1 + 24 a2 - 96 a1 b1 + 48 a1 b1^2 + 24 a2 b1^2 + 48 a1 b1 b2 - 93.5307` b2^2 - 93 a2 b2^2;
        cf3 = 16 - 48 b1 - 48 b1^2 + 16 b1^3 + 24 b2 + 24 b1^2 b2 - 31 b2^3;

In[ ]:= {time, solcubic1} = Timing[NSolve[{cf0, cf1, cf2, cf3}]];

In[ ]:= time
Out[ ]= 0.391518

In[ ]:= Length[solcubic1]
Out[ ]= 27
```

This takes a long time for a computer, but not much for a human. We now display the lines

```
In[ ]:= Do[Print["line[", i, "]=", line[i] = F1 /. solcubic1[[i]], {i, 27}]
```

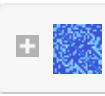
```
line[1]={t, -3.73243 + 13.9293 t, -5.46452 + 14.9294 t}
line[2]={t, 3.22448 + 4.08729 t, -3.00967 - 3.5649 t}
line[3]={t, 2.73814 + 3.4304 t, 1.87092 + 3.02721 t}
line[4]={t, -0.476643 - 1.47664 t, -1.90652 - 1.90653 t}
line[5]={t, 1.1547 - 1. t, -2.3094}
line[6]={t, 1.44663 + 6.17467 t, -2.47977 - 4.18706 t}
line[7]={t, 0.298434 - 0.815559 t, -1.39762 - 0.863769 t}
line[8]={t, 0. + 3.73205 t, 0.}
line[9]={t, 0.297094 + 0.485438 t, -1.62331 - 1.18835 t}
line[10]={t, 1.06079 - 0.957224 t, 0.265302 - 1.17278 t}
line[11]={t, 0.577351 - 1. t, -1.1547}
line[12]={t, 0.651252 + 2.63242 t, -1.11635 + 1.88495 t}
line[13]={t, 3.1547 + 3.73205 t, -2.3094}
line[14]={t, -0.234285 + 0.161952 t, -1.4988 - 0.678101 t}
line[15]={t, 0.365925 - 1.22615 t, -1.71369 + 1.05911 t}
line[16]={t, -0.845298 + 0.267949 t, -2.3094}
line[17]={t, 1.10819 - 1.04469 t, -1.03437 + 1.22519 t}
line[18]={t, -0.612013 + 2.05999 t, -0.896026 - 2.448 t}
line[19]={t, -0.798198 + 0.291512 t, -0.545395 + 0.882467 t}
line[20]={t, 0. - 1. t, 0.}
line[21]={t, -0.42265 + 0.267949 t, -1.1547}
line[22]={t, 0.267956 + 0.0717912 t, -1.4641 + 1.0718 t}
line[23]={t, 1.57735 + 3.73205 t, -1.1547}
line[24]={t, -0.247397 + 0.379879 t, -1.58268 + 0.716051 t}
line[25]={t, -0.788904 + 0.244661 t, -0.197304 - 0.872192 t}
line[26]={t, 0. + 0.267949 t, 0.}
line[27]={t, -0.322788 - 0.677211 t, -1.29112 + 1.29112 t}
```

We can now check with an incidence matrix using pLineIntersectionMD . We make this a little compli - cated for later use . Note an entry 0 means the lines are skew, 1 means they are the same, 3 means they intersect in the affine plane and 4 is an infinite intersection, that is the lines are parallel in affine 3 space.

In[ ]:=
```
lineList = Range[27]
```

Out[ ]=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27}
```

```
In[ ]:= incidence =
  SparseArray[Flatten[Table[{i, j} → Length[pLineIntersectionMD [line[lineList⟦i⟧],
        line[lineList⟦j⟧], t, {x, y, z}, .003]], {i, 27}, {j, 27}], 1]]
```

```
Out[ ]= SparseArray[  Specified elements : 297
                     Dimensions : {27, 27} ]
```

```
In[ ]:= M = Join[Partition[Prepend[lineList , 0], 1], Prepend[incidence , lineList], 2];
Grid[M,
  Background → {None , None , {{{1, 1}, {1, 28}} → LightGray , {{1, 28}, {1, 1}} → LightGray }}]
```

Out[ ]=

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 3 | 3 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 3 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 3 | 0 | 3 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 0 |
| 3 | 3 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 3 | 3 | 0 | 1 | 0 | 3 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 3 | 0 | 0 | 3 |
| 5 | 0 | 0 | 0 | 0 | 1 | 3 | 0 | 0 | 3 | 3 | 4 | 0 | 3 | 3 | 0 | 3 | 3 | 3 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 3 | 0 | 3 | 3 | 1 | 0 | 3 | 0 | 0 | 0 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 3 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 3 | 0 | 3 | 0 | 3 | 3 | 3 | 0 | 3 | 0 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 0 |
| 8 | 3 | 0 | 0 | 0 | 0 | 3 | 0 | 1 | 0 | 3 | 0 | 3 | 4 | 0 | 0 | 0 | 0 | 3 | 0 | 3 | 0 | 0 | 4 | 0 | 3 | 3 | 0 |
| 9 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 0 | 1 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 3 | 3 | 0 | 3 | 3 | 0 |
| 10 | 0 | 0 | 3 | 3 | 3 | 0 | 3 | 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 3 | 3 | 0 | 3 | 3 | 0 | 0 |
| 11 | 3 | 3 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 3 | 3 | 3 | 3 | 3 | 0 | 0 |
| 12 | 0 | 0 | 3 | 0 | 0 | 3 | 3 | 3 | 3 | 0 | 3 | 1 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 3 |
| 13 | 0 | 3 | 3 | 0 | 3 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 4 | 3 | 0 | 0 | 3 |
| 14 | 3 | 0 | 3 | 0 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 |
| 15 | 3 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 3 | 0 | 0 | 0 | 3 | 0 | 1 | 0 | 3 | 0 | 0 | 3 | 3 | 0 | 0 | 3 | 3 | 0 | 0 |
| 16 | 3 | 0 | 0 | 3 | 3 | 0 | 3 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 1 | 0 | 0 | 3 | 0 | 4 | 0 | 0 | 0 | 3 | 4 | 0 |
| 17 | 3 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 3 | 0 | 3 | 0 | 0 | 3 | 0 | 1 | 0 | 3 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 3 |
| 18 | 3 | 3 | 0 | 0 | 3 | 0 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 3 |
| 19 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 1 | 3 | 0 | 3 | 3 | 3 | 3 | 0 | 0 |
| 20 | 0 | 0 | 3 | 3 | 4 | 0 | 3 | 3 | 0 | 0 | 4 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 3 | 3 |
| 21 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 3 | 4 | 0 | 3 | 0 | 0 | 1 | 0 | 3 | 0 | 0 | 4 | 3 |
| 22 | 3 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 3 | 3 | 3 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 3 |
| 23 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 4 | 3 | 0 | 3 | 0 | 4 | 3 | 0 | 0 | 3 | 0 | 3 | 0 | 3 | 0 | 1 | 0 | 0 | 0 | 0 |
| 24 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 0 |
| 25 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| 26 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 3 | 0 | 4 | 3 | 0 | 0 | 3 | 4 | 3 | 0 | 3 | 0 | 1 | 0 |
| 27 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 3 | 3 | 0 | 3 | 3 | 3 | 0 | 0 | 3 | 0 | 1 |

Notice the 1's lie all on the diagonal, so all these lines are distinct. Thus we have all 27 lines.

In the notebook Appendix C we re-arrange the lines to find a double 6. Remember that this is one example, not the only one.

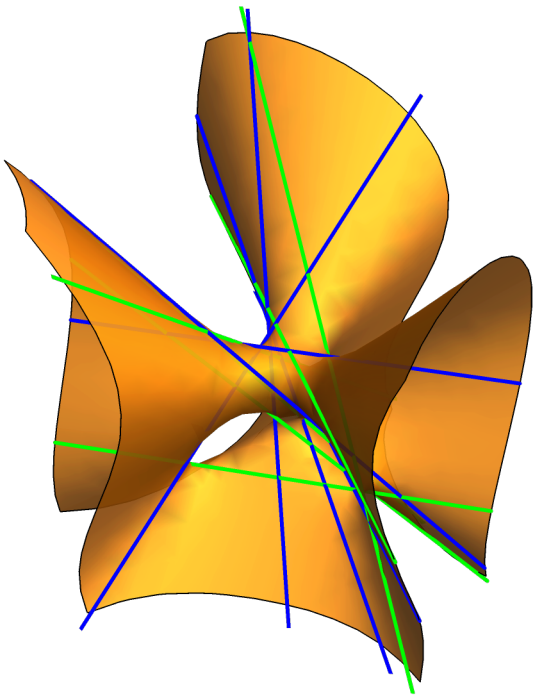
| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5 | 3 | 4 | 7 | 25 | 26 | 2 | 16 | 14 | 9 | 20 | 10 |
| 5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 4 | 3 |
| 3 | 0 | 1 | 0 | 0 | 0 | 0 | 3 | 0 | 3 | 3 | 3 | 3 |
| 4 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 3 | 0 | 3 | 3 | 3 |
| 7 | 0 | 0 | 0 | 1 | 0 | 0 | 3 | 3 | 3 | 0 | 3 | 3 |
| 25 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 3 | 3 | 3 | 0 | 3 |
| 26 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 4 | 3 | 3 | 3 | 0 |
| 2 | 0 | 3 | 3 | 3 | 3 | 3 | 1 | 0 | 0 | 0 | 0 | 0 |
| 16 | 3 | 0 | 3 | 3 | 3 | 4 | 0 | 1 | 0 | 0 | 0 | 0 |
| 14 | 3 | 3 | 0 | 3 | 3 | 3 | 0 | 0 | 1 | 0 | 0 | 0 |
| 9 | 3 | 3 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 1 | 0 | 0 |
| 20 | 4 | 3 | 3 | 3 | 0 | 3 | 0 | 0 | 0 | 0 | 1 | 0 |
| 10 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

In[ ]:=

The pink squares show the two sets of lines are each mutually skew, the cyan squares show the correct incidences among these lines. Note that two of these intersections are infinite. We can plot this

In[ ]:=
```
Show[ContourPlot3D [cubic1 == 0, {x, -4, 4}, {y, -4, 4},
  {z, -4, 4}, ContourStyle → Opacity[.9], Mesh → None], ParametricPlot3D [
  {line[2], line[16], line[14], line[9], line[20], line[10]}, {t, -4, 4}, PlotStyle → Green],
 ParametricPlot3D [{line[5], line[3], line[4], line[7], line[25], line[26]},
  {t, -4, 4}, PlotStyle → Blue], Axes → False , Boxed → False]
```

Out[ ]=



If we expand the picture above we get

Out[ ]=

| | 0 | 5 | 3 | 4 | 7 | 25 | 26 | 2 | 16 | 14 | 9 | 20 | 10 | 1 | 6 | 8 | 11 | 12 | 13 | 15 | 17 | 18 | 19 | 21 | 22 | 23 | 24 | 27 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 4 | 3 | 0 | 3 | 0 | 4 | 0 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 1 | 0 | 0 | 0 | 0 | 3 | 0 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 3 | 0 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 |
| | 7 | 0 | 0 | 0 | 1 | 0 | 0 | 3 | 3 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 3 | 0 | 3 | 0 | 3 | 0 | 0 | 3 | 3 | 0 | 0 |
| | 25 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 3 | 3 | 3 | 0 | 3 | 0 | 0 | 3 | 3 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 3 |
| | 26 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 4 | 3 | 3 | 3 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 4 | 3 | 0 | 3 | 0 |
| | 2 | 0 | 3 | 3 | 3 | 3 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 3 | 0 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 16 | 3 | 0 | 3 | 3 | 3 | 4 | 0 | 1 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 3 | 4 | 0 | 0 | 0 | 0 |
| | 14 | 3 | 3 | 0 | 3 | 3 | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 |
| | 9 | 3 | 3 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 3 | 0 | 3 | 0 | 0 | 3 | 3 | 0 | 0 |
| | 20 | 4 | 3 | 3 | 3 | 0 | 3 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 4 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 3 |
| | 10 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 3 | 0 | 3 | 0 |

we see each of the remaining lines intersect the double 6 in exactly 4 points. Most of these intersections involve only two lines intersection. Rarely we may have 3 lines intersecting if the intersection of the planes containing the double 2 goes through the intersection of two of the lines of the double 2. In the literature these are called an *Eckardt points.* These are easy to identify from the incidence matrix regarding the incidence matrix as an **Association**.

In[ ]:=
```
otherAssoc = <| Table[{i, j} → pLineIntersectionMD [line[i], line[j], t, {x, y, z}, .003],
     {i, 26}, {j, i + 1, 27}]|> ;
V = Select[Values[otherAssoc], Length[#] > 2 &];
st = Select[Tally[V], #[[2]] > 1 &]
```

Out[ ]=
```
{{{0., 0., 0.}, 3}}
```

So the only Eckardt point is the origin . Finding the lines

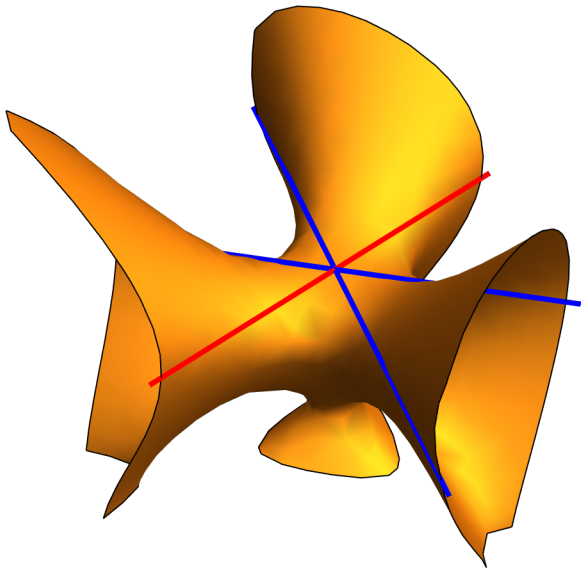
In[ ]:=
```
KeySelect[otherAssoc , otherAssoc[#] == {0, 0, 0} &]
```

Out[ ]=
```
<|{8, 20} → {0., 0., 0.}, {8, 26} → {0., 0., 0.}, {20, 26} → {0., 0., 0.}|>
```

So the single Eckardt is the intersection of lines 20 and 26 of the double 2 and 8 outside the double 2.

In[ ]:=
```
Show[ContourPlot3D [cubic1 == 0, {x, -2, 2}, {y, -2, 2}, {z, -2, 2}, Mesh → None],
 ParametricPlot3D [{line[8], line[20], line[26]}, {t, -3, 3}, PlotStyle → {Red, Blue, Blue}],
 Axes → False, Boxed → False]
```

Out[ ]=



## 1.9 .8 Finding lines, Example 2

My second example is the famous surface known as the *Clebsch diagonal Cubic.* Not only does this surface have 27 real lines they lie in such a way as to make a pleasing plot. This is also symmetric in all the variables. One discussion is at http://mathworld.wolfram.com/ClebschDiagonalCubic.html. This is also known in the literature as *Klein's icosahedral cubic.* A more complete discussion with moving pictures is by John Baez in https://blogs.ams.org/visualinsight/2016/03/01/clebsch-surface/ where he includes several plots by the science fiction writer Greg Egan. So I will not attempt a full computation Another interesting thing is that there are reportedly 10 Eckardt points. I will find some of these points, following the method above.

In[ ]:=
```
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;
```

We first find all the lines .

```
In[ ]:= cdcEq = Collect[Expand[cdc /. Thread[{x, y, z} → F1]], t]
```

```
Out[ ]= 1 - 9 a1 - 9 a1^2 + 81 a1^3 - 9 a2 + 126 a1 a2 - 189 a1^2 a2 - 9 a2^2 - 189 a1 a2^2 + 81 a2^3 +
  (-9 + 126 a1 - 189 a1^2 + 126 a2 + 54 a1 a2 - 189 a2^2 - 9 b1 - 18 a1 b1 + 243 a1^2 b1 + 126 a2 b1 -
     378 a1 a2 b1 - 189 a2^2 b1 - 9 b2 + 126 a1 b2 - 189 a1^2 b2 - 18 a2 b2 - 378 a1 a2 b2 + 243 a2^2 b2)
   t + (-9 - 189 a1 - 189 a2 + 126 b1 - 378 a1 b1 + 54 a2 b1 - 9 b1^2 + 243 a1 b1^2 -
     189 a2 b1^2 + 126 b2 + 54 a1 b2 - 378 a2 b2 + 126 b1 b2 -
     378 a1 b1 b2 - 378 a2 b1 b2 - 9 b2^2 - 189 a1 b2^2 + 243 a2 b2^2) t^2 +
  (81 - 189 b1 - 189 b1^2 + 81 b1^3 - 189 b2 + 54 b1 b2 - 189 b1^2 b2 - 189 b2^2 - 189 b1 b2^2 + 81 b2^3) t^3
```

```
In[ ]:= cdc0 = 1 - 9 a1 - 9 a1^2 + 81 a1^3 - 9 a2 + 126 a1 a2 - 189 a1^2 a2 - 9 a2^2 - 189 a1 a2^2 + 81 a2^3;
cdc1 = -9 + 126 a1 - 189 a1^2 + 126 a2 + 54 a1 a2 - 189 a2^2 - 9 b1 - 18 a1 b1 + 243 a1^2 b1 + 126 a2 b1 -
    378 a1 a2 b1 - 189 a2^2 b1 - 9 b2 + 126 a1 b2 - 189 a1^2 b2 - 18 a2 b2 - 378 a1 a2 b2 + 243 a2^2 b2;
cdc2 = -9 - 189 a1 - 189 a2 + 126 b1 - 378 a1 b1 + 54 a2 b1 - 9 b1^2 + 243 a1 b1^2 - 189 a2 b1^2 + 126 b2 +
    54 a1 b2 - 378 a2 b2 + 126 b1 b2 - 378 a1 b1 b2 - 378 a2 b1 b2 - 9 b2^2 - 189 a1 b2^2 + 243 a2 b2^2;
cdc3 = 81 - 189 b1 - 189 b1^2 + 81 b1^3 - 189 b2 + 54 b1 b2 - 189 b1^2 b2 - 189 b2^2 - 189 b1 b2^2 + 81 b2^3;
```

```
In[ ]:= solcdc = NSolve[{cdc0, cdc1, cdc2, cdc3}];
Do[Print["cline[", i, "]=", cline[i] = F1 /. solcdc[[i]], {i, 22}]
```

```
cline[1]={t, 2.2847 - 5.23607 t, 0.872678 - 2.23607 t}
cline[2]={t, 0.390273 - 0.447214 t, 0.241202 + 2.34164 t}
cline[3]={t, -0.333333 + 3. t, 0.}
cline[4]={t, 0.0486327 - 0.763932 t, 0.127322 + 2.23607 t}
cline[5]={t, 0.127322 + 2.23607 t, 0.0486327 - 0.763932 t}
cline[6]={t, 0. , -0.333333 + 3. t}
cline[7]={t, 0.666667 - 1. t, 0.333333 }
cline[8]={t, 0.269672 - 2.92705 t, 0.063661 - 1.30902 t}
cline[9]={t, 0.241202 + 2.34164 t, 0.390273 - 0.447214 t}
cline[10]={t, 0.872678 - 2.23607 t, 2.2847 - 5.23607 t}
cline[11]={t, 0.333333 - 1. t, 0.}
cline[12]={t, -0.333333 , 0. - 1. t}
cline[13]={t, 0.436339 - 0.190983 t, -0.103006 + 0.427051 t}
cline[14]={t, 0.063661 - 1.30902 t, 0.269672 - 2.92705 t}
cline[15]={t, 0.0921311 - 0.341641 t, -0.0569401 + 0.447214 t}
cline[16]={t, -0.0569401 + 0.447214 t, 0.0921311 - 0.341641 t}
cline[17]={t, 0. - 1. t, -0.333333 }
cline[18]={t, 0., 0.333333 - 1. t}
cline[19]={t, 0., 0.111111 + 0.333333 t}
cline[20]={t, 0.333333 , 0.666667 - 1. t}
cline[21]={t, 0.111111 + 0.333333 t, 0.}
cline[22]={t, -0.103006 + 0.427051 t, 0.436339 - 0.190983 t}
```

In[ ]:=
```
Length[solcdc]
```

Out[ ]= 22

So we don' t get all the lines but one can get the other lines by symmetry .

In[ ]:=
```
cline[23] = {0, -1 / 3 + 3 t, t};
cline[24] = {0, 1 / 3 - t, t};
cline[25] = {-1 / 3, -t, t};
cline[26] = {0, t, -1 / 3 + 3 t};
cline[27] = {1 / 3, t, 2 / 3 - t};
```

In[ ]:=
```
Simplify[cdc /. Thread[{x, y, z} → cline[27]]]
```

Out[ ]= 0

Our incidence chart can be calculated .

```
In[ ]:= lineList = Range[27]
Out[ ]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
  14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27}

In[ ]:= incidence2 =
  SparseArray[Flatten[Table[{i, j} → Length[pLineIntersectionMD [cline[lineList⟦i⟧],
      cline[lineList⟦j⟧], t, {x, y, z}, .003]], {i, 27}, {j, 27}], 1]]

Out[ ]= SparseArray[  Specified elements : 297
Dimensions : {27, 27} ]

In[ ]:= M2 = Join[Partition[Prepend[lineList, 0], 1], Prepend[incidence2, lineList], 2];
Grid[M2,
  Background → {None, None, {{{1, 1}, {1, 28}} → LightGray, {{1, 28}, {1, 1}} → LightGray}}]
```

Out[ ]=

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 3 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 3 | 0 | 3 | 0 | 0 | 0 | 3 |
| 2 | 0 | 1 | 0 | 3 | 0 | 3 | 0 | 3 | 3 | 3 | 3 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 3 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 3 | 0 | 0 | 0 | 3 |
| 4 | 0 | 3 | 0 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 3 | 3 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 0 | 3 |
| 5 | 0 | 0 | 0 | 3 | 1 | 0 | 0 | 3 | 3 | 0 | 3 | 3 | 3 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 |
| 6 | 0 | 3 | 3 | 0 | 0 | 1 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 3 | 3 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 3 | 0 | 4 | 0 | 0 | 0 | 3 | 0 | 4 | 0 | 3 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 3 |
| 8 | 3 | 3 | 0 | 0 | 3 | 3 | 3 | 1 | 0 | 0 | 0 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 0 |
| 9 | 3 | 3 | 3 | 0 | 3 | 0 | 3 | 0 | 1 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 3 | 0 |
| 10 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 3 | 3 |
| 11 | 0 | 3 | 3 | 0 | 3 | 0 | 4 | 0 | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 3 | 4 | 3 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 0 |
| 12 | 0 | 0 | 3 | 0 | 3 | 0 | 0 | 3 | 0 | 3 | 0 | 1 | 0 | 0 | 0 | 0 | 3 | 4 | 0 | 4 | 0 | 3 | 3 | 0 | 3 | 0 | 0 |
| 13 | 3 | 3 | 3 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 0 | 3 | 0 | 3 | 3 | 0 | 3 | 0 | 3 | 0 | 0 | 0 |
| 14 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 0 | 1 | 0 | 3 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| 15 | 0 | 0 | 3 | 3 | 0 | 0 | 3 | 3 | 0 | 3 | 0 | 0 | 3 | 0 | 1 | 3 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 0 |
| 16 | 3 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 3 | 1 | 0 | 0 | 0 | 3 | 0 | 3 | 3 | 0 | 3 | 0 | 0 |
| 17 | 3 | 0 | 0 | 3 | 0 | 3 | 4 | 0 | 0 | 0 | 4 | 3 | 3 | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 0 |
| 18 | 3 | 0 | 0 | 3 | 0 | 3 | 0 | 0 | 3 | 0 | 3 | 4 | 0 | 0 | 3 | 0 | 0 | 1 | 3 | 4 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 3 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 3 | 1 | 0 | 3 | 0 | 3 | 0 | 3 | 0 | 0 |
| 20 | 0 | 3 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 4 | 3 | 3 | 0 | 3 | 0 | 4 | 0 | 1 | 3 | 0 | 0 | 0 | 0 | 3 | 3 |
| 21 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 1 | 3 | 0 | 0 | 3 | 3 | 0 |
| 22 | 0 | 0 | 0 | 3 | 0 | 3 | 3 | 0 | 3 | 3 | 0 | 3 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 3 | 1 | 0 | 3 | 0 | 0 | 0 |
| 23 | 3 | 3 | 3 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 1 | 3 | 0 | 3 | 0 |
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 3 | 3 | 1 | 4 | 3 | 4 |
| 25 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 0 | 0 | 3 | 3 | 3 | 0 | 3 | 0 | 3 | 0 | 0 | 4 | 1 | 0 | 4 |
| 26 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 3 | 0 | 3 | 0 | 0 | 3 | 3 | 0 | 3 | 3 | 0 | 1 | 0 |
| 27 | 3 | 0 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 4 | 4 | 0 | 1 |

We don't have any duplicates so this must be all.

We now look for the famous Eckart points in this example.

```
In[ ]:= otherAssoc2 = <| Table[{i, j} → pLineIntersectionMD [cline[i], cline[j], t, {x, y, z}, .003],
          {i, 26}, {j, i + 1, 27}]|> ;
        V2 = KeySelect[otherAssoc2 , Length[otherAssoc2 [#]] == 3 &];

In[ ]:= st = Select[Tally[Values[V2], Norm[#1 - #2] < 1.*^-9 &], #[[2]] > 1 &]

Out[ ]= {{{0.166667 , 0.166667 , 0.}, 3}, {{1.4866 × 10^-14, -0.333333 , -4.91517 × 10^-15}, 3},
         {{-8.17955 × 10^-14, 2.72734 × 10^-14, -0.333333}, 3},
         {{0.166667 , -1.48845 × 10^-16, 0.166667}, 3}, {{0.333333 , 0.333333 , 0.333333}, 3},
         {{-0.333333 , 1.02521 × 10^-14, -1.01915 × 10^-14}, 3},
         {{1.04294 × 10^-17, 0.166667 , 0.166667}, 3}}

In[ ]:= KeySelect[V2, Norm[V2[#] - st[[1, 1]]] < 1.*^-9 &]

Out[ ]= <|{3, 11} → {0.166667 , 0.166667 , 0.},
         {3, 21} → {0.166667 , 0.166667 , 0.}, {11, 21} → {0.166667 , 0.166667 , 0.}|>

In[ ]:= KeySelect[V2, Norm[V2[#] - st[[2, 1]]] < 1.*^-9 &]

Out[ ]= <|{3, 12} → {1.4866 × 10^-14, -0.333333 , -4.91517 × 10^-15},
         {3, 23} → {2.1065 × 10^-15, -0.333333 , -2.1065 × 10^-15},
         {12, 23} → {-2.79385 × 10^-15, -0.333333 , 8.13327 × 10^-15}|>

In[ ]:= KeySelect[V2, Norm[V2[#] - st[[3, 1]]] < 1.*^-9 &]

Out[ ]= <|{6, 17} → {-8.17955 × 10^-14, 2.72734 × 10^-14, -0.333333},
         {6, 26} → {-4.60317 × 10^-15, 4.3122 × 10^-15, -0.333333},
         {17, 26} → {2.27423 × 10^-14, -6.82551 × 10^-14, -0.333333}|>

In[ ]:= KeySelect[V2, Norm[V2[#] - st[[4, 1]]] < 1.*^-9 &]

Out[ ]= <|{6, 18} → {0.166667 , -1.48845 × 10^-16, 0.166667},
         {6, 19} → {0.166667 , 6.92135 × 10^-18, 0.166667},
         {18, 19} → {0.166667 , -4.11295 × 10^-17, 0.166667}|>

In[ ]:= KeySelect[V2, Norm[V2[#] - st[[5, 1]]] < 1.*^-9 &]

Out[ ]= <|{7, 20} → {0.333333 , 0.333333 , 0.333333},
         {7, 27} → {0.333333 , 0.333333 , 0.333333}, {20, 27} → {0.333333 , 0.333333 , 0.333333}|>

In[ ]:= KeySelect[V2, Norm[V2[#] - st[[6, 1]]] < 1.*^-9 &]

Out[ ]= <|{19, 21} → {-0.333333 , 1.02521 × 10^-14, -1.01915 × 10^-14},
         {19, 25} → {-0.333333 , 2.80014 × 10^-14, -8.34944 × 10^-14},
         {21, 25} → {-0.333333 , -5.30136 × 10^-14, 1.76712 × 10^-14}|>

In[ ]:= KeySelect[V2, Norm[V2[#] - st[[7, 1]]] < 1.*^-9 &]

Out[ ]= <|{23, 24} → {1.04294 × 10^-17, 0.166667 , 0.166667},
         {23, 26} → {1.88326 × 10^-18, 0.166667 , 0.166667},
         {24, 26} → {5.73977 × 10^-17, 0.166667 , 0.166667}|>
```

So we find 7 Eckardt points ,these are all rational. The others are infinite.

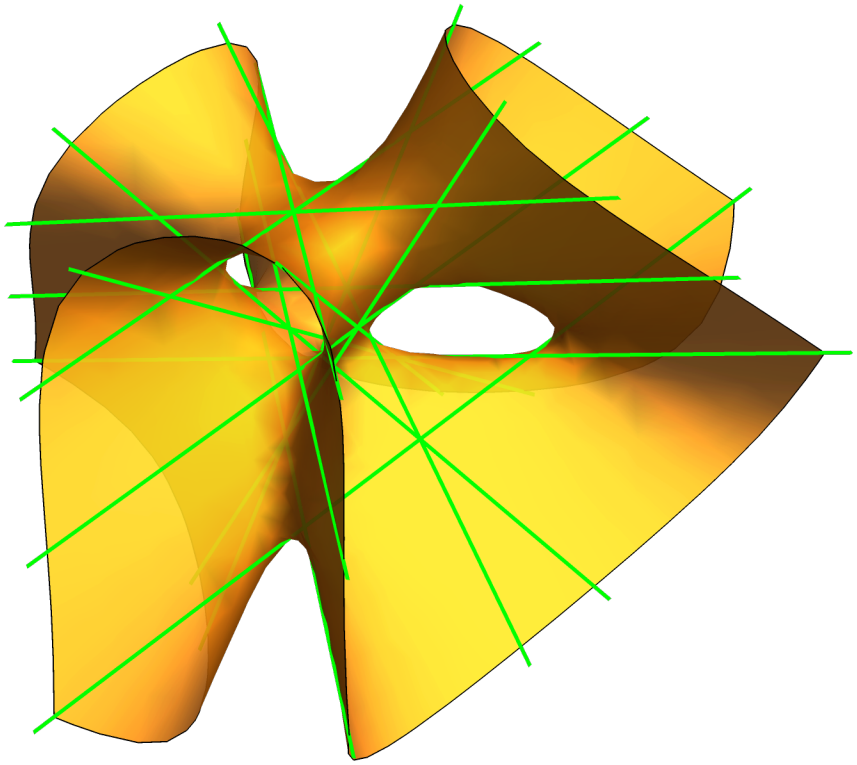
```
In[ ]:= epoints = {{1/6, 1/6, 0}, {0, -1/3, 0}, {0, 0, -1/3},
    {1/6, 1/6, 0}, {1/3, 1/3, 1/3}, {-1/3, 0, 0}, {0, 1/6, 1/6, 0}};
```

Note by symmetry there are only 3 different orbits, one of length 1.

```
In[ ]:= elines = DeleteDuplicates [
   {3, 11, 21, 12, 23, 6, 17, 26, 6, 18, 19, 7, 20, 27, 19, 21, 25, 23, 24, 26}]
Out[ ]= {3, 11, 21, 12, 23, 6, 17, 26, 18, 19, 7, 20, 27, 25, 24}
```

```
In[ ]:= Show[ContourPlot3D [cdc == 0, {x, -1, 1},
    {y, -1, 1}, {z, -1, 1}, Mesh → None , ContourStyle → Opacity[0.9]],
  ParametricPlot3D [cline[#] & /@ elines , {t, -3, 3}, PlotStyle → Green],
  Axes → False , Boxed → False]
```

Out[ ]=



# References


Barry H Dayton *Plane Curve Book* A numerical Approach to Real Algebraic Curves, Wolfram Media, 2018. Paperback, Kindle versions available from Amazon.com, notebook versions from **wolfr.am/DaytonCurves , barryhdayton.space**

Barry H Dayton *Space Curve Book,* 2020, available at **barryhdayton.space/SpaceCurves/SpaceCurveBook_v2c.pdf** Notebook version at **notebookarchive.org**

Barry H Dayton *Ideals of numeric realizations of configurations of lines.* Contemporary Mathematics 496: Interactions of Classical and Numerical Algebraic Geometry, AMS, pp.181--191, 2009.

Barry H Dayton Degree vs *Dimension for Rational Parametric Curves,* Mathematica Journal 22, 2020.

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

Sebastián Montiel, Antonio Ros *Curves and Surfaces,* Graduate Studies in Mathematics 69, AMS and Real Sociedad Matemática Española, 2005.

David H von Seggern , *CRC Standard Curves and Surfaces with Mathematica,* CRC press, Third Edition, 2016.

David Hilbert, S . Cohn - Vossen, *Geometry and the Imagination* , Chelsea, 1952.

Eric Weisstein **http://mathworld.wolfram.com/ClebschDiagonalCubic.html**

John Baez **https://blogs.ams.org/visualinsight/2016/03/01/clebsch-surface/**