Surface Story

Part II

Barry H Dayton
barryhdayton.space

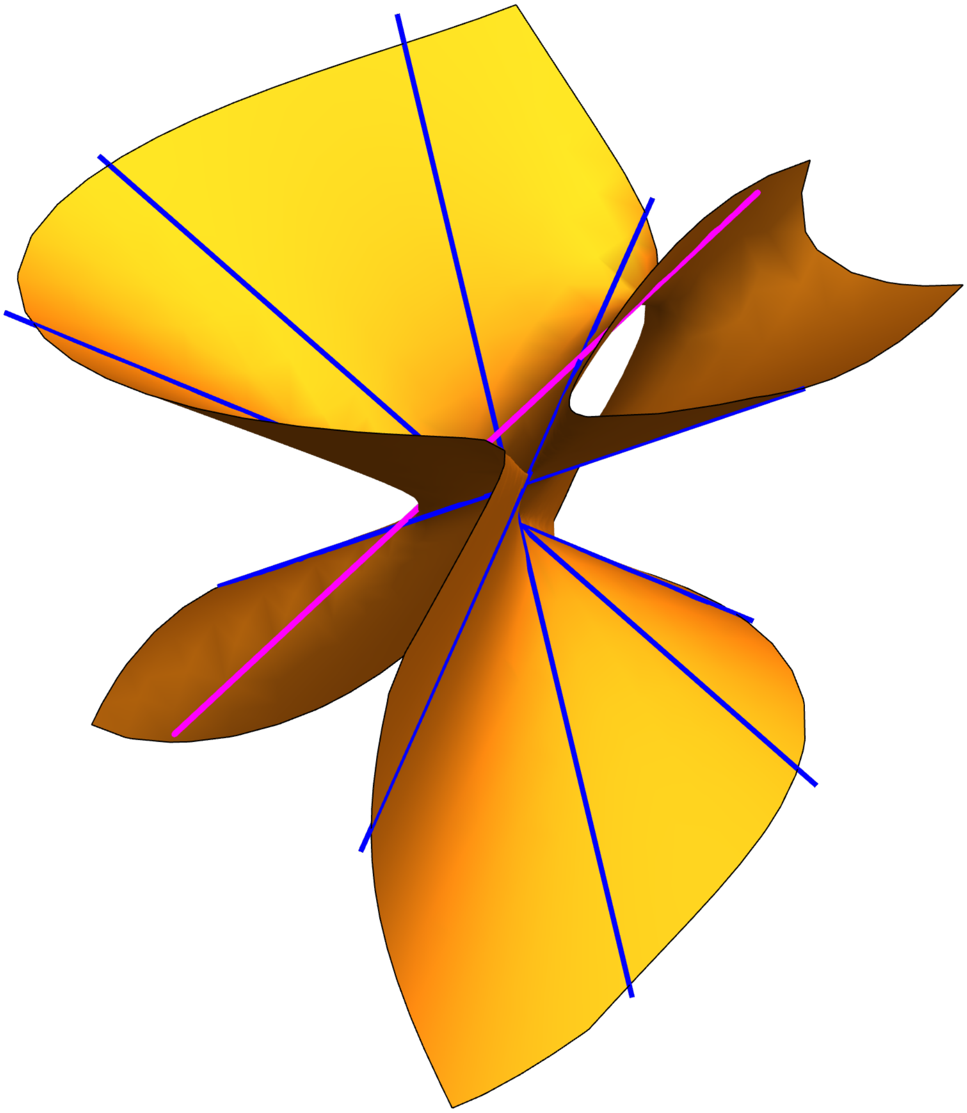


Table of Contents

The author makes no representations, express or implied, with respect to this documentation or software it describes, including, without limitation, any implied warranties of merchantability, interoperability or fitness for a particular purpose, all of which are expressly disclaimed. Use of Mathematica and other related software is subject to the terms and conditions as described at www.wolfram.com/legal .

In addition to the forgoing, users should recognize that all complex software systems and their documentation contain errors and omissions. Barry H. Dayton and Wolfram Research a) shall not be responsible under any circumstances for providing information or corrections to errors and omissions discovered at any time in this book or software; b) shall not be liable for damages of any kind arising out of the use of (or inability to use) this book or software; c) do not recommend the use of the software for applications in which errors or omissions could threaten life, or cause injury or significant loss.

Mathematica and Wolfram Language are trademarks of Wolfram Research Inc.

2 Quadric Surfaces in Projective Space

We will illustrate our transformations by discussing an important classical subject.

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.* The hyperboloid of 2 sheets is actually an ellipsoid. I will use some non-standard terminology but suggest that it be widely adopted.

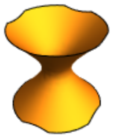
2.1 General Results

Quadric surfaces are defined from our 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 $a_1, a_2, \ldots, a_6$ not zero. For example a random quadric might be f231 = 0

```
In[ ∘ ]:= f231 = 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

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
```

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

One general comment is that since these actual surfaces are affine surfaces of degree 2 any line transversal to these surfaces intersects the surface in 2 points by multiplicity. Thus these are all orientable, that is 2 sided as projective surfaces. I will make some comments on the types .

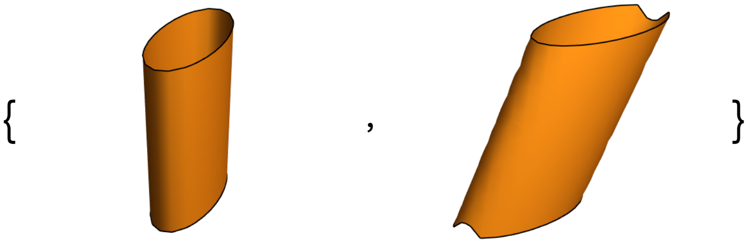
Since we are looking at real points we could get an empty set or a zero or one dimensional set. Also we could get a non-square free surface, that is a double plane with equation $(ax + by + cz - d)^2$. These are not surfaces, they have no regular points.

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. Since we are working strictly with real quadrics we also should include here empty quadrics, for example $x^2 + y^2 + z^2 + 1$.

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 ^ 2 + y ^ 2 − 1 where that on the right is a rotation applied to this first equation giving

```
In[ ∘ ]:= cyl = N[FLTNS[x ^ 2 + y ^ 2 - 1, m2TM[RotationMatrix [{{1, 0, 0}, {2, 1, 3}}]], {x, y, z}]]
Out[ ∘ ]= -1. + 0.357143 x^2 - 0.223927 x y + 0.9805 y^2 + 0.931785 x z + 0.162285 y z + 0.662357 z^2
```

In[∘]:=



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.

In particular note that the projective transformation with transformation matrix

```
In[ ∘ ]:= CC3 = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 0, 1}, {0, 0, 1, 0}};
```

```
In[ ∘ ]:= CC3 // MatrixForm
Out[ ∘ ]//MatrixForm=
```

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

In[ ]:=
```
FLTNS[x^2 + y^2 - z^2, CC3, {x, y, z}]
```

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

takes our standard cone to the circular cylinder and inversely.

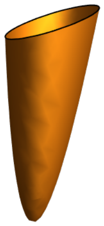
In[ ]:=
```
FLTNS[x^2 + y^2 - 1, Inverse[CC3], {x, y, z}]
```

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

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).

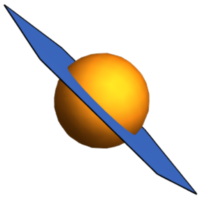
In[ ]:=

In[ ]:=

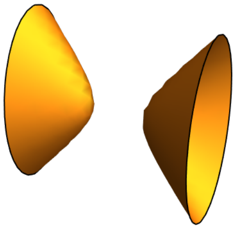


On the other hand if we cut the ellipsoid with a plane which goes to infinity we get

In[ ]:= {



,



}

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.

I mention here that we will record here and in GlobalFunctions.nb a projective transformation matrix taking the paraboloid $z = x^2 + y^2$ to the sphere $x^2 + y^2 + z^2 - 1$. This will be used in our “proof” of the chart as it is very easy to transform any type of ellipsoid to a paraboloid using iTransform.

In[ ]:=
```
paraboloid2sphere
```

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

```
In[ ]:= FLTNS[z - x ^ 2 - y ^ 2, paraboloid2sphere , {x, y, z}]
```

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

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.* The below right is the *parabolic hyperboloid* , otherwise known as the hyperbolic paraboloid or *saddle surface* . In the literature these are often considered to be different but again note

```
In[ ]:= Ht = {{1.421753448878254` , 2.4001312247824407` ,
          -1.4217534488782626` , -2.4001312247824362` }, {-1.3682399203220887` ,
          0.05585142943475707` , 0.7762628086454613` , 1.1281027939185155` },
        {-2.550704470740892` , -1.499868080914514` , 0.08976727903245796` ,
          2.957640849193627` }, {1.1547005383792515` , 0.5773502691896257` ,
          -1.1547005383792515` , -0.5773502691896266` }};
        Ht // MatrixForm
```

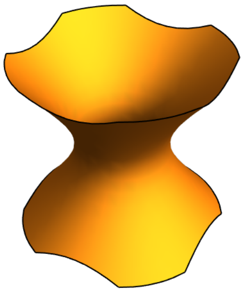
```
Out[ ]//MatrixForm=
   1.42175    2.40013   -1.42175   -2.40013
  -1.36824  0.0558514   0.776263     1.1281
   -2.5507   -1.49987  0.0897673    2.95764
    1.1547    0.57735    -1.1547   -0.57735
```

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

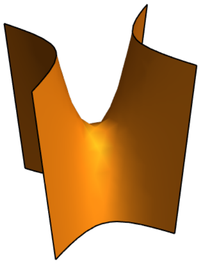
```
Out[ ]= 1. x y - 1. z
```

```
In[ ]:= {ContourPlot3D [x ^ 2 + y ^ 2 - z ^ 2 == 1, {x, -2, 2},
          {y, -2, 2}, {z, -2, 2}, Mesh → None, Boxed → False, Axes → False],
        ContourPlot3D [z == x y, {x, -8, 8}, {y, -8, 8}, {z, -5, 5}, Mesh → None,
          Boxed → False, Axes → False], ContourPlot3D [9 x ^ 2 + y ^ 2 - z ^ 2 - 5 x y - z == 15,
          {x, -8, 8}, {y, -8, 8}, {z, -5, 5}, Mesh → None, Boxed → False, Axes → False]}
```

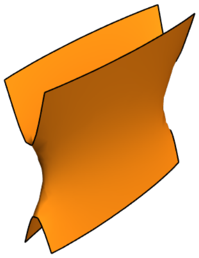
Out[ ]= {



,



,



}

The two hyperboloids do share 4 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.

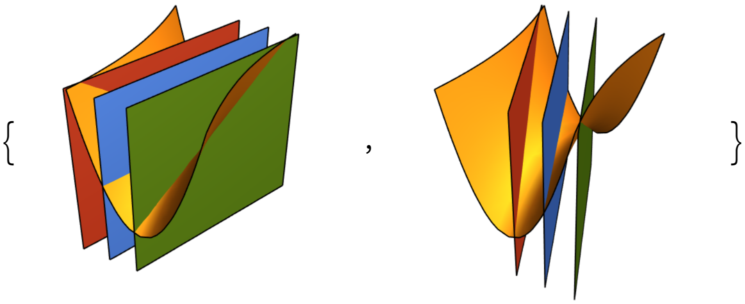
3) The hyperboloid is determined by any 3 skew lines, that is any three skew lines in 3-space are part of one ruling of a hyperboloid.

4) They are rationally parameterized surfaces.

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.

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.

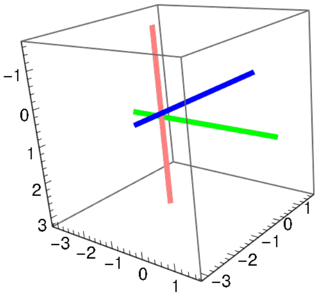
In fact a hyperbolic paraboloid is simply a hyperboloid which is tangent to the infinite plane. Note that the maximal form in either equation $z = x\,y$ or $z = x^2 - y^2$ are both a union of two lines so these hyperboloids have infinite curves which satisfy condition 2) above.

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

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

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

Out[∘]=



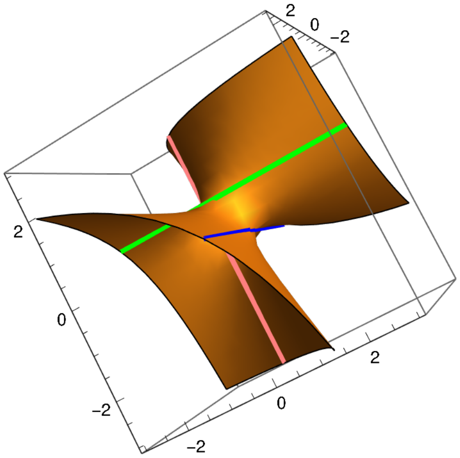
The equations are

```
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[ ]=



[

I remark in passing that if we were considering complex projective surfaces then the ellipsoid and hyperboloid are projectively equivalent. The tangent plane to a point in, say the real sphere, does contain two complex lines which lie in the complex sphere. But in the real projective space the sphere is not a ruled surface.

Our main result of this Section is that *Every real projective quadric surface is projectively equivalent to exactly one of our example surfaces.* The one minor exception is that the non-squarefree degenerate surface is not projectively equivalent to the squarefree degenerate surface.

The rest of this section will be devoted to proving this . Along with this constructive proof we will learn more about each of the types of quadric surface.

2.2 Strategy

Given a thee variable quadratic equation we first pick a random point, assuming it is not not the empty quadric. A good way to do this is to use the closestRealPointMD function and a random point. Here is an example

```
In[ ]:= f232 = 1.004299994444187` + 4.619946233491519` x +
           5.003986917416253` x^2 - 1.5312443087645962` y - 2.5456169581885573` x y -
           0.18725675804366315` y^2 - 1.4437724690088531` z -
           4.988262328875971` x z + 3.7338496490520834` y z - 1.7296244962937` z^2

Out[ ]= 1.0043 + 4.61995 x + 5.00399 x^2 - 1.53124 y - 2.54562 x y -
         0.187257 y^2 - 1.44377 z - 4.98826 x z + 3.73385 y z - 1.72962 z^2
```

```
In[ ]:= p232 = closestRealPointMD [{f232}, RandomReal [{- 5, 5}, 3], {x, y, z}]

Out[ ]= {1.15112 , -1.11181 , 1.3013}
```

If there is none the real quadric is probably empty . A plot may help confirm this .

We then eliminate the non-surface cases by checking the regularity of the random point. The probability that a random point of a surface is near zero. A good way to do this is to attempt to calculate the tangent plane at this point. This does a check but one may need to look at the tangent plane if it is given, all very small (eg 10^{-4}) coefficients look suspiciously like a singular point.

```
In[ ]:= Tp232 = tangentPlaneNS [f232 , p232 , {x, y, z}]

Out[ ]= 7.1504 + 12.4794 x + 0.813695 y - 15.8387 z
```

This looks good . Next we check to see if this is the degenerate case. We use nDivideMDQ. This is given in global function pages after March 2022 or below.

```
In[ ]:= nDivideMDQ [f232 , Tp232 , {x, y, z}, .0003]

Out[ ]= True
```

Now this looks degenerate.

If the preceding does not happen we proceed to calculate any lines in the quadric through this random regular point. We adopt the trick used in Section 1.9.7 of this book to calculate lines on a cubic. If there is one line we have a cone (cylinder), no real lines give an ellipsoid while two real lines indicate a hyperboloid. In each case we will show that the information from the point and lines on the quadric through this point are sufficient to find a transformation function taking the quadric to the standard quadric of its type.

We use the following black box code to find the random point and make the checks above for non-surface or degenerate surfaces and if that is not the case looks for lines through the random point. Note for technical reasons this function will not handle cylinders defined by only 2 variables, that is cylinders parallel to one axis. But then there is one line in the direction of the missing variable through each point so this function is not needed anyway.

Example : We use the quadric at the beginning of this section

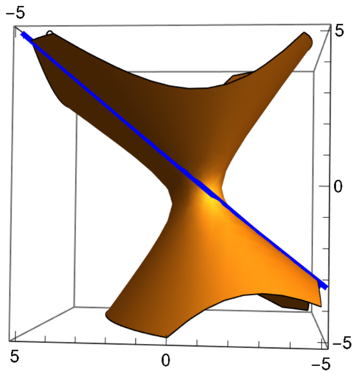
```
In[ ]:= f231

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

In[ ]:= Lines = analyzeQSNS[f231, {x, y, z}]

   » 2 Lines

Out[ ]= {{-0.686027 - 3.31495 t, -2.74836 + 8.54218 t, -1.54699 + 7.20492 t},
         {-0.686027 - 0.591156 t, -2.74836 - 0.65952 t, -1.54699 - 0.492003 t}}
```

```
In[ ]:= analyzeQSNS[f_, V_] := Module[{Tp, p, F, G, ct, ct2, sol, ln1, ln2, a, b, c},
          If[Length[Variables[Chop[f, 1.*^-6]]] < 3,
           Echo["Quadrics must use all 3 variables"];
           Abort[]];
          p = Quiet[closestRealPointMD[{f}, RandomReal[{-5, 5}, 3], V]];
          If[Abs[f /. Thread[V → p]] > .003, Echo["Possible Empty Quadric"];
           Abort[]];
          Tp = With[{Gr = Grad[f, V] /. Thread[{x, y, z} → p]},
            If[Norm[Gr] > 1.*^-5, Expand[Gr.(V - p)], Echo["Not Regular at"];
             Return[p]]];
          If[Abs[Tp /. Thread[V → Normalize[RandomReal[{-1, 1}, 3]]]] < .003,
           Echo["Random non-regular point, Possibly not a Surface"];
           Return[p]];
          If[nDivideMDQ[f, Tp, V, .001], Echo["Possibly Degenerate"];
           Return[p]];
          F = {p⟦1⟧ + a t, p⟦2⟧ + b t, p⟦3⟧ + c t};
          G = Expand[f /. Thread[V → F]];
          ct = Coefficient[G, t];
          ct2 = Coefficient[G, t^2];
          sol = Quiet[NSolve[{ct, ct2}, {a, b, c}, Reals]];
          If[Length[sol] == 0, Echo["No lines, random point given"]; Return[p]];
          ln1 = F /. sol⟦1⟧;
          If[Length[sol] == 1, Return[ln1]];
          ln2 = F /. sol⟦2⟧;
          n = Length[pLineIntersectionMD[ln1, ln2, t, V, .03]];
          Which[n == 1, Echo["One Line"];
           Return[ln1], n == 3, Echo["2 Lines"];
           {ln1, ln2}, True, Fail]]
```

So this will be a hyperboloid .

```
In[ ∘ ]:= Show[ContourPlot3D [f231 == 0, {x, -5, 5},
    {y, -5, 5}, {z, -5, 5}, Mesh → None, MaxRecursion → 4],
  ParametricPlot3D [Lines, {t, -5, 5}, PlotStyle → Blue], ImageSize → Small]
```

Out[∘]=



We will show later how this can be wrangled to our standard hyperboloid $x^2 + y^2 - z^2 = 1$.

Remark 1: If you do not have an updated GlobalFunctions.nb the ndivideMDQ is given by

```
In[ ∘ ]:= nDivideMDQ [h_, g_, X_, tol_] := Module[{n, l, m, d1, d2, P, S, f, ex, t},
    n = Length[X];
    d1 = tDegMD[g, X];
    d2 = tDegMD[h, X];
    If [d1 > d2, Return[False]];
    P = PseudoInverse [N[sylMD[g, d2, X]], Tolerance → tol];
    S = Chop[sylMD[h, d2, X].P];
    ex = expsMD[n, d2 - d1];
    l = Length[ex];
    f = FromCoefficientRules [Table[ex⟦i⟧ → S⟦1, i⟧, {i, l}], X];
    t = Expand[f * g - h];
    If[NumberQ[t],
     If[Abs[t] < d2 * tol, Return[True], Return[False]]];
    If[Norm[Flatten[sylMD[Expand[f * g - h], d2, X]]] > d2 * tol, Return[False]];
    True];
```

REMARK 2: The next part of this section gets very long and technical. The reader who is just happy to know our basic classification may skip to subsection 2.3.7. The reader who wants to know why this classification works may skim the following subsections. These subsections are given for completeness and the occasional reader who actually needs to transform a complicated quadric surface to one in standard form. However subsections 2.3.7 to 2.3.10 give new material.

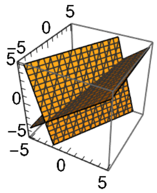
```
In[ ∘ ]:= Clear[f, g, h, p, q, p2, tplane]
```

2.3 Degenerate Case

Here the quadric has real points but is singular. We are sent here if **NSolve[Grad[f,{x,y,z}]]** appears infinite. We have already found a random point so we can check if it is regular. For example consider f232 above

```
In[ ]:= f232
Out[ ]= 1.0043 + 4.61995 x + 5.00399 x^2 - 1.53124 y - 2.54562 x y -
          0.187257 y^2 - 1.44377 z - 4.98826 x z + 3.73385 y z - 1.72962 z^2
```

```
In[ ]:= ContourPlot3D [f232 == 0, {x, -5, 5}, {y, -5, 5}, {z, -5, 5}, ImageSize → Tiny]
Out[ ]=
```



```
In[ ]:= TP232 = tangentPlaneNS [f232 , p232 , {x, y, z}]
Out[ ]= 7.1504 + 12.4794 x + 0.813695 y - 15.8387 z
```

Since the result is a plane the point was regular. We can see if this is a component

```
In[ ]:= PL232 = nDivideMD [f232 , TP232 , {x, y, z}, 1.*^-6]
Out[ ]= 0.140454 + 0.400981 x - 0.230131 y + 0.109202 z
```

Thus f232 is the union of two planes . The intersecting line is given by

```
In[ ]:= NSolve[Grad[f232 , {x, y, z}]]
Out[ ]= {{x → -0.550258 + 1.11193 z, y → -0.348449 + 2.41194 z}}
```

which by inspection contains the point

```
In[ ]:= pt232 = {-0.5502577015419445` , -0.3484492995311171` , 0}
Out[ ]= {-0.550258 , -0.348449 , 0}
```

```
In[ ]:= Check :
```

```
In[ ]:= TP232 /. Thread[{x, y, z} → pt232]
        PL232 /. Thread[{x, y, z} → pt232]
Out[ ]= 4.6595 × 10^-12
Out[ ]= -2.58127 × 10^-14
```

I now translate f232 and its factor planes TP232 and PL232 so that one the point pt232 is moved to the origin. This is done by the transformation matrix

```
In[ ]:= T232 = {{1, 0, 0, -pt232⟦1⟧}, {0, 1, 0, -pt232⟦2⟧}, {0, 0, 1, 0}, {0, 0, 0, 1}};
        T232 // MatrixForm
Out[ ]//MatrixForm=
        ⎛1  0  0  0.550258⎞
        ⎜0  1  0  0.348449⎟
        ⎜0  0  1     0    ⎟
        ⎝0  0  0     1    ⎠
```

```
In[ ]:= f232a = FLTNS[f232, T232, {x, y, z}]
        TP232a = Chop[FLTNS[TP232, T232, {x, y, z}], 10*^-9]
        PL232a = FLTNS[PL232, T232, {x, y, z}]
```

$5.00399\, x^2 - 2.54562\, x\, y - 0.187257\, y^2 - 4.98826\, x\, z + 3.73385\, y\, z - 1.72962\, z^2$

$12.4794\, x + 0.813695\, y - 15.8387\, z$

$0.400981\, x - 0.230131\, y + 0.109202\, z$

Since there are no constant terms they pass through the origin . Next we rotate our planes to send TP232 to the horizontal plane $z = 0$.

```
In[ ]:= PR232 = planeRotate3D[TP232a, z];
        PR232 // MatrixForm
Out[ ]//MatrixForm=
        ⎛-0.777288  -0.115885   -0.61838   0.⎞
        ⎜-0.115885   0.992444  -0.0403204  0.⎟
        ⎜ 0.61838    0.0403204  -0.784844  0.⎟
        ⎝    0.         0.          0.     1.⎠
```

```
In[ ]:= f232b = Chop[FLTNS[f232a, PR232, {x, y, z}], 1.*^-9]
        TP232b = FLTNS[TP232a, PR232, {x, y, z}]
        PL232b = Chop[FLTNS[PL232a, PR232, {x, y, z}], 1.*^-10]
```

$-7.11447\, x\, z - 5.63574\, y\, z + 3.08711\, z^2$

$20.1807\, z$

$-0.352537\, x - 0.279263\, y + 0.152973\, z$

Our planes still go through the origin but the first factor is now the $z = 0$ plane. When $y = 1$, $z = 0$ then solving

```
In[ ]:= gc = PL232b /. {y → 1, z → 0}
Out[ ]= -0.279263 - 0.352537 x
```

```
In[ ]:= gcx = SolveValues[gc == 0, x]⟦1⟧
Out[ ]= -0.792152
```

so PL232b goes through the point

```
In[ • ]:= q232b = {gcx, 1, 0}
          PL232b /. Thread[{x, y, z} → q232b]
Out[ • ]= {-0.792152, 1, 0}
Out[ • ]= 0.
```

```
In[ • ]:= q232bt = Take[q232b, 2]
Out[ • ]= {-0.792152, 1}
```

So we take the 2 dimensional rotation about the origin that takes the vector **q232bt** to {0,1} and extend it to a three dimensional transformation matrix leaving the z-plane fixed.

```
In[ • ]:= RM232 = Simplify[Join[Join[RotationMatrix[{{gcx, 1}, {0, 1}}], 0 * IdentityMatrix[2], 2],
              {{0, 0, 1, 0}, {0, 0, 0, 1}}]];
          RM232 // MatrixForm
Out[ • ]//MatrixForm=
          ( 0.783861   0.620937  0  0 )
          (-0.620937   0.783861  0  0 )
          (    0          0      1  0 )
          (    0          0      0  1 )
```

Now

```
In[ • ]:= f232c = Chop[FLTNS[f232b, RM232, {x, y, z}], 1.*^-11]
          TP232c = FLTNS[TP232b, RM232, {x, y, z}]
          PL232c = FLTNS[PL232b, RM232, {x, y, z}]
Out[ • ]= -9.07619 x z + 3.08711 z^2
Out[ • ]= 20.1807 z
Out[ • ]= -0.449745 x + 0.152973 z
```

We note the line {y=0,z=0} now lies on all three planes so is the intersection of **TP232c** and **PL232c.**

Finally we do a 3 dimensional shear

```
In[ • ]:= Clear[a]
```

```
In[ • ]:= Sh232 = {{1, 0, a, 0}, {0, 1, 0, 0}, {0, 0, -9.076189692468215` , 0}, {0, 0, 0, 1}}
Out[ • ]= {{1, 0, a, 0}, {0, 1, 0, 0}, {0, 0, -9.07619, 0}, {0, 0, 0, 1}}
```

Note

```
In[ • ]:=
```

```
In[ • ]:= gd = Chop[FLTNS[f232c, Sh232, {x, y, z}], 1.*^-10]
          gdz = Chop[gd /. {x → 0, z → 1}, 1.*^-10]
Out[ • ]= 1. x z + 0.0374752 z^2 + 0.110178 a z^2
Out[ • ]= 0.0374752 + 0.110178 a
```

```
In[ ]:= sol232 = Solve[gdz == 0, a]
Out[ ]= {{a → -0.340132 }}
```

```
In[ ]:= Sh232a = Sh232 /. sol232⟦1⟧
Out[ ]= {{1, 0, -0.340132 , 0}, {0, 1, 0, 0}, {0, 0, -9.07619 , 0}, {0, 0, 0, 1}}
```

our result is

```
In[ ]:=
        Chop[FLTNS[f232c , Sh232a , {x, y, z}], 1.*^-10]
Out[ ]= 1. x z
```

which was our target equation!

Letting

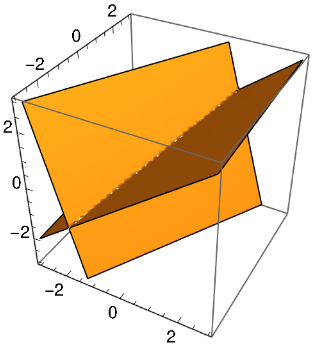
```
In[ ]:= A232 = Sh232a .RM232 .PR232 .T232 ;
        A232 // MatrixForm
Out[ ]//MatrixForm=
        ( -0.891574   0.511693  -0.242809  -0.312296 )
        (  0.391809   0.849895   0.352369   0.511741 )
        ( -5.61253   -0.365955   7.1234    -3.21586  )
        (  0.         0.         0.         1.       )
```

```
In[ ]:= Chop[FLTNS[f232 , A232 , {x, y, z}], 1.*^-9]
Out[ ]= 1. x z
```

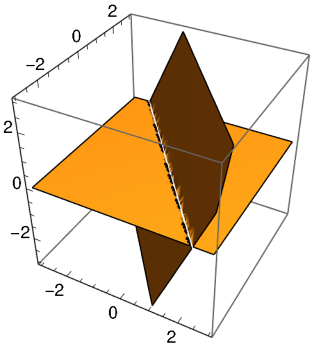
so up to a tiny numerical error we have transformed f232 to the standard example, in this case with an affine transformation. Here are some plots

```
In[ ]:= {ContourPlot3D[f232 == 0, {x, -3, 3}, {y, -3, 3}, {z, -3, 3}, Mesh → None],
          ContourPlot3D[f232b == 0, {x, -3, 3}, {y, -3, 3}, {z, -3, 3}, Mesh → None],
          ContourPlot3D[f232c == 0, {x, -3, 3}, {y, -3, 3}, {z, -3, 3}, Mesh → None],
          ContourPlot3D[x z == 0, {x, -3, 3}, {y, -3, 3}, {z, -3, 3}, Mesh → None]}
```

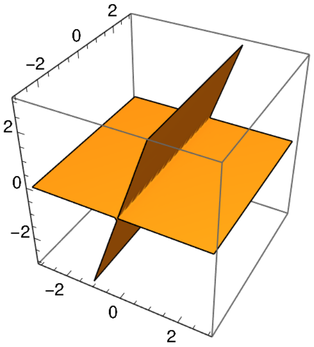
Out[ ]= {



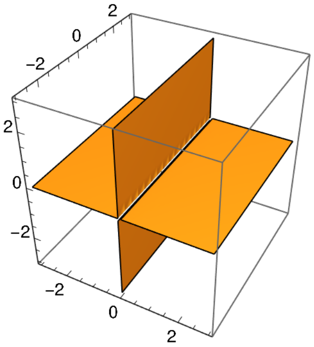
,



,



,



}

2.4 Case of single line

I give two random examples .

2.4.1 First Example

```
In[ ]:= f234 = -2.3020166207367843` - 2.6858797219485577` x +
           1.0131161481023399` x^2 + 1.4721025020329819` y +
           3.6587010950658008` x y + 2.676268803498578` y^2 + 3.662928463334536` z +
           5.874375409773489` x z + 3.229386168894008` y z + 0.7908431365144266` z^2
```

Out[ ]= $-2.30202 - 2.68588\ x + 1.01312\ x^2 + 1.4721\ y + 3.6587\ x\ y + 2.67627\ y^2 + 3.66293\ z + 5.87438\ x\ z + 3.22939\ y\ z + 0.790843\ z^2$

We start by analyzing our quadric .

```
In[ ]:= Line234a = analyzeQSNS[f234, {x, y, z}]
```

```
» One Line
```

```
Out[ ]= {3.03683 - 1.6243 t, -5.2475 + 2.13762 t, -2.75568 + 1.61337 t}
```

We see there is a single line through the point

```
In[ • ]:= p1 = Line234a /. {t → 0}
Out[ • ]= {3.03683, -5.2475, -2.75568}
```

This says we have a cone or a cylinder. To find out witch we run this again

```
In[ • ]:= Line234b = analyzeQSNS[f234, {x, y, z}]
   » One Line
Out[ • ]= {-2.4303 - 1.004 t, 0.00758587 + 0.218295 t, 1.03954 + 0.067496 t}
```

which is through the point

```
In[ • ]:= p2 = Line234b /. {t → 0}
Out[ • ]= {-2.4303, 0.00758587, 1.03954}
```

Now we check to see if they intersect.

```
In[ • ]:= p3 = pLineIntersectionMD[Line234a, Line234b, t, {x, y, z}, .003]
Out[ • ]= {-0.664603, -0.376321, 0.920841}
```

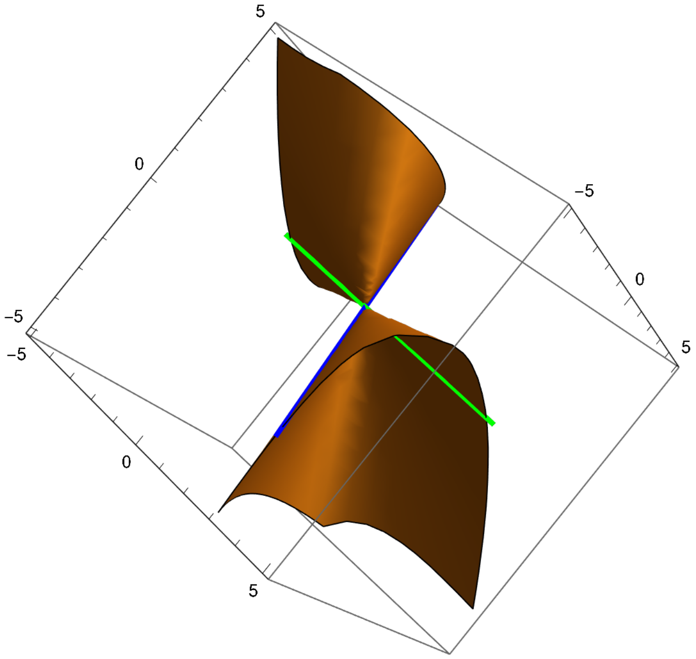
They do. We check for regularity

```
In[ • ]:= tangentPlaneNS[f234, p3, {x, y, z}]
   » Not Regular at
Out[ • ]= {-0.664603, -0.376321, 0.920841}
```

So this is a singular point. We have a cone. Here is a picture

In[ ]:=
```
Show[ContourPlot3D[f234 == 0, {x, -5, 5}, {y, -5, 5}, {z, -5, 5}, Mesh → None],
  ParametricPlot3D[{Line234a, Line234b}, {t, -15, 5}, PlotStyle → {Blue, Green}]]
```

Out[ ]=



To put this in our standard surface form we first move our singular point to the invisible plane, preferably a unit coordinate point. We first take it to zero and then use a transformation from 2.3.1 to put the singular point at invisible point {0,0,1,0}.

In[ ]:=
```
T234 = {{1, 0, 0, -p3[[1]]}, {0, 1, 0, -p3[[2]]}, {0, 0, 1, -p3[[3]]}, {0, 0, 0, 1}}
CC3 = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 0, 1}, {0, 0, 1, 0}}
B234 = CC3.T234;
B234 // MatrixForm
```

Out[ ]=
```
{{1, 0, 0, 0.664603}, {0, 1, 0, 0.376321}, {0, 0, 1, -0.920841}, {0, 0, 0, 1}}
```

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

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1. & 0. & 0. & 0.664603 \\ 0. & 1. & 0. & 0.376321 \\ 0. & 0. & 0. & 1. \\ 0. & 0. & 1. & -0.920841 \end{pmatrix}$$

Note

In[ ]:=
```
fltiMD[p3, B234]
```

Out[ ]=
```
{0., 0., 1., 0}
```

In[ ]:=
```
f234b = Chop[FLTNS[f234, B234, {x, y, z}], 1.*^-6]
```

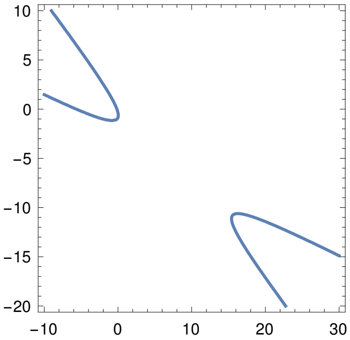
Out[ ]=

$$0.790843 + 5.87438\ x + 1.01312\ x^2 + 3.22939\ y + 3.6587\ x\ y + 2.67627\ y^2$$

There is no z term! As a surface this is a cylinder. But its intersection with the z-plane is the curve with the same equation but considered as a plane curve instead of a space cylinder.

```
In[ ]:= ContourPlot[f234b == 0, {x, -10, 30}, {y, -20, 10}, ImageSize → Small]
```

Out[ ]=



This plane curve will be some conic, in this case a hyperbola . But it is not actually important as in my *Plane Curve Book* Chapter 7 there is a single method for reducing any non-singular conic to the unit circle which involves the cTransform2D **(Paragraph 70.1 GlobalFunctionsS.nb)** which takes this to a parabola which can be transformed to $y = x^2$ followed by a standard transformation taking this parabola to the unit circle.

```
In[ ]:= p2cTransform2D
Out[ ]= {{1, 0, 0}, {0, -0.5, 0.5}, {0, -0.5, -0.5}}
```

Another trick used is to escalate a FLT on 2 - space to 3 space by changing

Out[ ]//MatrixForm=

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

To

Out[ ]//MatrixForm=

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

where the * indicate numbers in the same position. We have a function `escalate2D`

So now we find some **critical points** and apply the `cTransform2D`

```
In[ ]:= cpf234b = criticalPoints2D[f234b, x, y]⟦2⟧
Out[ ]= {-0.112914, -0.0524928}

In[ ]:= ctf234b = Chop[cTransform2D[f234b, cpf234b, x, y], dTol]
Out[ ]= {{0.421563, -0.906799, 0},
         {-0.112049, -0.0520905, 0.992336}, {0.89985, 0.418332, 0.123565}}

In[ ]:= gc = FLT3D[{f234b}, ctf234b, {x, y}]⟦1⟧
Out[ ]= 3.49815 - 3.65788 x + 0.982079 x^2 + 6.01404 y
```

We do our translation trick in 2 D

```
In[∘]:= Clear[a, b, c]

In[∘]:= T2D = {{1, 0, a}, {0, 1, b}, {0, 0 , 1}};

In[∘]:= gt = FLT3D[{gc}, T2D, {x, y}]⟦1⟧
Out[∘]= 3.49815 + 3.65788 a + 0.982079 a^2 - 6.01404 b - 3.65788 x - 1.96416 a x + 0.982079 x^2 + 6.01404 y

In[∘]:= c0 = gt /. Thread[{x, y} → {0, 0}]
Out[∘]= 3.49815 + 3.65788 a + 0.982079 a^2 - 6.01404 b

In[∘]:= cx = Coefficient[gt, x] /. {x → 0}
Out[∘]= -3.65788 - 1.96416 a

In[∘]:= cy = Coefficient[gt, y]
Out[∘]= 6.01404

In[∘]:= solgt = Solve[c0 == 0 && cx == 0, {a, b}]
        ••• 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 .
Out[∘]= {{a → -1.86231 , b → 0.0153135 }}

In[∘]:= T234c = T2D /. solgt⟦1⟧
Out[∘]= {{1, 0, -1.86231}, {0, 1, 0.0153135 }, {0, 0, 1}}

In[∘]:= g234c = FLT3D[{gc}, T234c , {x, y}]
Out[∘]= {0.982079 x^2 + 6.01404 y}
```

A parabola, now almost here, we modify T2D by

```
In[∘]:= TT234d =
         ReplacePart [T234c , {3, 3} → First[-Coefficient [g234c , y] / Coefficient [g234c , x ^ 2]]]
Out[∘]= {{1, 0, -1.86231}, {0, 1, 0.0153135 }, {0, 0, -6.12379 }}

In[∘]:= f234d = FLT3D[{gc}, p2cTransform2D .TT234d , {x, y}]⟦1⟧
Out[∘]= -0.982079 + 0.982079 x^2 + 0.982079 y^2
```

But this is equivalent to

```
In[∘]:= Expand[f234d / f234d⟦1⟧]
Out[∘]= 1. - 1. x^2 - 1. y^2
```

That is the unit circle, our goal . Putting this together

```
In[∘]:= B2D = p2cTransform2D .TT234d .ctf234b
Out[∘]= {{-1.25424 , -1.68586 , -0.230117 },
         {-2.70611 , -1.25805 , -0.875458 }, {2.80438 , 1.30373 , -0.118771 }}
```

```
In[ ]:= FLT3D[{f234b}, B2D, {x, y}]
Out[ ]= {-0.982079 + 0.982079 x^2 + 0.982079 y^2}
```

Using our trick above, editing manually

```
In[ ]:= B3D = escalate2D[B2D];
        B3D // MatrixForm
Out[ ]//MatrixForm=
        ( -1.25424  -1.68586  0  -0.230117 )
        ( -2.70611  -1.25805  0  -0.875458 )
        (     0         0     1      0     )
        (  2.80438   1.30373  0  -0.118771 )
```

```
In[ ]:= FLTNS[f234b, B3D, {x, y, z}]
Out[ ]= -0.982079 + 0.982079 x^2 + 0.982079 y^2
```

Note this last equation is now in 3 dimensions, that is, a right circular cylinder of radius 1. But inserting the transformation getting f234b we get, eliminating some small error on the magnitude of 10^{-8}

```
In[ ]:= cyl = Chop[FLTNS[f234, CC3.B3D.B234, {x, y, z}], 1.*^-6]
Out[ ]= 0.982079 x^2 + 0.982079 y^2 - 0.982079 z^2
```

or equivalently

```
In[ ]:= roundPolyMD[Expand[cyl / Coefficient[cyl, x^2]], {x, y, z}, 1]
Out[ ]= x^2 + y^2 - z^2
```

which has converted our original quadric f234 to the standard cone. Done! Note for reference the transformation matrix is

```
In[ ]:= A = CC3.B3D.B234;
        A // MatrixForm
Out[ ]//MatrixForm=
        ( -1.25424  -1.68586  -0.230117  -1.25609 )
        ( -2.70611  -1.25805  -0.875458  -1.46576 )
        (  2.80438   1.30373  -0.118771   2.46379 )
        (    0.        0.         0.         1.   )
```

so even though we did use some projective transformations the end transformation is just an affine transformation .

Finally we saw in Chapter 1 that the cone had trigonometric parameterization

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

So f234 has trigonometric parameterization

```
In[ ]:= TransformationFunction [Inverse[A]][{s Cos[t], s Sin[t], s}]
```

Out[ ]=
$$\{-0.664603+0.387474\ s+0.421563\ s\ \mathrm{Cos}[t]-0.163377\ s\ \mathrm{Sin}[t],\ -0.376321-0.155234\ s-0.906799\ s\ \mathrm{Cos}[t]+0.259415\ s\ \mathrm{Sin}[t],\ 0.920841-0.97464\ s+1.26591\times10^{-15}\ s\ \mathrm{Cos}[t]-1.01003\ s\ \mathrm{Sin}[t]\}$$

or rational parameterization

```
In[ ]:= prcone = TransformationFunction [Inverse[A]][
          {2 s u / (1 + u ^ 2), s (1 - u ^ 2) / (1 + u ^ 2), s (1 + u ^ 2) / (1 + u ^ 2)}]
```

Out[ ]=
$$\left\{-0.664603+0.387474\ s+\frac{0.843126\ s\ u}{1+u^2}-\frac{0.163377\ s\ (1-u^2)}{1+u^2},\ -0.376321-0.155234\ s-\frac{1.8136\ s\ u}{1+u^2}+\frac{0.259415\ s\ (1-u^2)}{1+u^2},\ 0.920841-0.97464\ s+\frac{2.53181\times10^{-15}\ s\ u}{1+u^2}-\frac{1.01003\ s\ (1-u^2)}{1+u^2}\right\}$$

2.4.2 Second Example

We look at the example of a cylinder above. We start out the same

```
In[ ]:= g234 = cyl
```

Out[ ]=
$$0.982079\ x^2+0.982079\ y^2-0.982079\ z^2$$

```
In[ ]:= Linesg234a = analyzeQSNS[g234, {x, y, z}]

   » One Line

Out[ ]= {0.773121 - 0.0881251 t, 3.41867 - 0.38968 t, -3.505 + 0.399521 t}
```

So we do have a cone or cylinder .

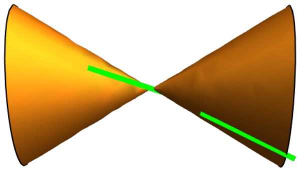
```
In[ ]:= pg234a = Linesg234a /. {t → 0}

Out[ ]= {0.773121, 3.41867, -3.505}

In[ ]:= Linesg234b = analyzeQSNS[g234, {x, y, z}]

   » One Line

Out[ ]= {1.84396 - 0.525789 t, -0.578447 + 0.164939 t, -1.93256 + 0.551053 t}

In[ ]:= pg234b = Linesg234b /. {t → 0}

Out[ ]= {1.84396, -0.578447, -1.93256}

In[ ]:= pg234c = pLineIntersectionMD [Linesg234a, Linesg234b, t, {x, y, z}, .003]
```

Out[ ]=
$$\{6.98723\times10^{-11},\ 8.19944\times10^{-11},\ 1.41363\times10^{-10}\}$$

We see this lines are parallel, intersecting in an infinite point .

```
In[ • ]:= Show[ContourPlot3D [{g234 == 0}, {x, -4, 4},
            {y, -6, 6}, {z, -3, 3}, Mesh → None, MaxRecursion → 5],
          ParametricPlot3D [{Linesg234a , Linesg234b }, {t, -6, 6}, PlotStyle → Green],
          Graphics3D [{Red, PointSize [.04], Point[{pg234a , pg234b}]}],
          Axes → False, Boxed → False, ImageSize → Small]
```

Out[•]=



Now we rotate our cylinder by

```
In[ • ]:= A234 = m2TM[RotationMatrix [{Take[pg234c , 3], {0, 0, 1}}]];
          A234 // MatrixForm
```

```
Out[ • ]//MatrixForm=
 ( 0.913915   -0.101019  -0.393133  0 )
 (-0.101019    0.881455  -0.461337  0 )
 ( 0.393133    0.461337   0.79537   0 )
 (    0           0          0      1 )
```

```
In[ • ]:= k = FLTNS[g234 , A234 , {x, y, z}]
```

Out[•]= $0.678511\, x^2 - 0.712467\, x\, y + 0.564043\, y^2 + 1.22833\, x\, z + 1.44143\, y\, z - 0.260475\, z^2$

so

```
In[ • ]:= FLTNS[k, CC3 , {x, y, z}]
```

Out[•]= $-0.260475 + 1.22833\, x + 0.678511\, x^2 + 1.44143\, y - 0.712467\, x\, y + 0.564043\, y^2$

is the cone, which was our goal . So we have transformed g234 to the standard cone as advertised. Note that unlike our cone this required a projective transformation.

```
In[ • ]:= CC3 .A234 // MatrixForm
```

```
Out[ • ]//MatrixForm=
 ( 0.913915   -0.101019  -0.393133  0. )
 (-0.101019    0.881455  -0.461337  0. )
 (    0.          0.         0.     1. )
 ( 0.393133    0.461337   0.79537   0. )
```

2.5 Case of no real lines

I illustrate with a randomly generated quadric without looking at the plot.

```
In[ ]:= f235 = -2.031178358884528` + 4.17957755275523` x +
           4.997732894861038` x^2 + 4.016412314718252` y + 5.405655213618456` x y -
           4.774616171824391` y^2 - 0.3635208665574865` z + 2.1158591510475233` x z -
           1.9584210592684848` y z - 3.6055350881202237` z^2

Out[ ]= -2.03118 + 4.17958 x + 4.99773 x^2 + 4.01641 y + 5.40566 x y -
          4.77462 y^2 - 0.363521 z + 2.11586 x z - 1.95842 y z - 3.60554 z^2
```

Our usual first step is

```
In[ ]:= p235 = analyzeQSNS[f235, {x, y, z}]

     » No lines, random point given

In[ ]:= p235 = {-3.6256394688081572`, 2.2338455675877786`, -2.1132818872948613`}

Out[ ]= {-3.62564, 2.23385, -2.11328}
```

We must show this is projectively equivalent to the unit sphere . We next find the tangent plane.

```
In[ ]:= tp235 = tangentPlaneNS[f235, p235, {x, y, z}]

Out[ ]= -9.47573 - 24.4564 x - 32.7754 y + 2.82935 z
```

We apply our `iTransform3D` specializing at this tangent plane, that is making **tp235** invisible.

```
In[ ]:= iT235 = iTransform3D[tp235]

Out[ ]= {{0.309503, -0.941557, 0.132962, -0.381119}, {-0.70722, -0.134457, 0.69409, -1.58421},
          {-0.635647, -0.308856, -0.707503, 3.56495}, {-1.03337, -1.38487, 0.11955, -0.400382}}
```

In particular p235 goes to

```
In[ ]:= q235 = fltiMD[p235, iT235]

Out[ ]= {-3.88754, -0.787249, 6.67479, 0}
```

and our quadric now has equation in this specialization

```
In[ ]:= f235b = FLTNS[f235, iT235, {x, y, z}]

Out[ ]= -7.0792 + 10.9293 x - 8.91098 x^2 + 32.5164 y - 28.4045 x y -
          25.1027 y^2 + 13.7463 z - 13.73 x z - 22.4648 y z - 5.32312 z^2
```

We now rotate a vector in the direction of q235 to {0,0,1} using a rotation matrix about the origin.

```
In[ ]:= R235 = m2TM[RotationMatrix[{Take[q235, 3], {0, 0, 1}}]]

Out[ ]= {{0.865196, -0.0272985, 0.500689, 0}, {-0.0272985, 0.994472, 0.101392, 0},
          {-0.500689, -0.101392, 0.859668, 0}, {0, 0, 0, 1}}
```

Rotating our quadric **f235b**

```
In[ ]:= f235c = Chop[FLTNS[f235b, R235, {x, y, z}], 1.*^-7]

Out[ ]= -7.0792 + 15.451 x - 12.9935 x^2 + 33.4321 y - 35.3577 x y - 26.3433 y^2 + 3.04812 z
```

We see that this is a paraboloid from the equation. We will put this in standard form $z = x^2 + y^2$ and then transform using our **GlobalFunctions.nb** transformation `paraboloid2sphere.`

An affine transformation involving a shear and translation will be sufficient.

```
In[ ]:= Clear[a, b, c, u]
        Tgen = {{1, 0, 0, a}, {u, 1, 0, b}, {0, 0, 1, c}, {0, 0, 0, 1}};

In[ ]:= f235tg = FLTNS[f235c, Tgen, {x, y, z}]

Out[ ]= -7.0792 - 15.451 a - 12.9935 a^2 - 33.4321 b - 35.3577 a b - 26.3433 b^2 -
         3.04812 c + 33.4321 a u + 35.3577 a^2 u + 52.6867 a b u - 26.3433 a^2 u^2 + 15.451 x +
         25.9869 a x + 35.3577 b x - 33.4321 u x - 70.7155 a u x - 52.6867 b u x +
         52.6867 a u^2 x - 12.9935 x^2 + 35.3577 u x^2 - 26.3433 u^2 x^2 + 33.4321 y + 35.3577 a y +
         52.6867 b y - 52.6867 a u y - 35.3577 x y + 52.6867 u x y - 26.3433 y^2 + 3.04812 z

In[ ]:= c0 = f235tg /. Thread[{x, y, z} → {0, 0, 0}]

Out[ ]= -7.0792 - 15.451 a - 12.9935 a^2 - 33.4321 b - 35.3577 a b - 26.3433 b^2 -
         3.04812 c + 33.4321 a u + 35.3577 a^2 u + 52.6867 a b u - 26.3433 a^2 u^2

In[ ]:= cx = Coefficient[f235tg, x] /. {y → 0}

Out[ ]= 15.451 + 25.9869 a + 35.3577 b - 33.4321 u - 70.7155 a u - 52.6867 b u + 52.6867 a u^2

In[ ]:= cy = Coefficient[f235tg, y] /. {x → 0}

Out[ ]= 33.4321 + 35.3577 a + 52.6867 b - 52.6867 a u

In[ ]:= cxy = Coefficient[f235tg, x y]

Out[ ]= -35.3577 + 52.6867 u

In[ ]:= sol235tg = Solve[c0 == 0 && cx == 0 && cy == 0 && cxy == 0, {a, b, c, u}]

        ••• 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 .

Out[ ]= {{a → 3.09277, b → -0.634545, c → 4.70113, u → 0.671095}}

In[ ]:= T235 = Tgen /. sol235tg[[1]];
        T235 // MatrixForm

Out[ ]//MatrixForm=
        (    1      0  0   3.09277  )
        ( 0.671095  1  0  -0.634545 )
        (    0      0  1   4.70113  )
        (    0      0  0      1     )

In[ ]:= f235d = FLTNS[f235c, T235, {x, y, z}]

Out[ ]= -1.12927 x^2 - 26.3433 y^2 + 3.04812 z
```

We can now put this in standard form for a paraboloid (parabolic ellipsoid in my notation) using a homothety

```
In[ ]:= Homth235 =
          {{Sqrt[-Coefficient[f235d, x^2]], 0, 0, 0}, {0, Sqrt[-Coefficient[f235d, y^2]], 0, 0},
           {0, 0, Coefficient[f235d, z], 0}, {0, 0, 0, 1}};
        Homth235 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1.06267 & 0 & 0 & 0 \\ 0 & 5.13258 & 0 & 0 \\ 0 & 0 & 3.04812 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
In[ ]:= FLTNS[f235d, Homth235, {x, y, z}]
```

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

We are done as we have in GlobalFunctionsS.nb

```
In[ ]:= FLTNS[z - x^2 - y^2, paraboloid2sphere , {x, y, z}]
```

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

so the projective transformation

```
In[ ]:= A235 = paraboloid2sphere .Homth235.T235.R235.iT235;
        A235 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -7.84687 & -8.89527 & -0.338903 & 2.53791 \\ -3.42938 & -5.57762 & 0.118585 & 0.276449 \\ -0.72582 & 0.469251 & 1.87743 & 0.287739 \\ -8.88023 & -10.2801 & -0.219354 & 2.13753 \end{pmatrix}$$

gives the sphere to 7 decimal places

```
In[ ]:= roundPolyMD[FLTNS[f235, A235, {x, y, z}], {x, y, z}, 5.*^-7]
```

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

2.6 Case of 2 real lines

This is the hard case of this section since a hyperboloid can be either a parabolic or an elliptic hyperboloid a situation where we really need projective geometry . It is the reason this section is in Chapter 2.

2.6.1 Special case

Here we start with the standard hyperboloid and a specific, not random, point and transform this to the saddle surface $z = x\, y$. This is an easier target than the standard hyperboloid.

```
In[ ]:= f2361 = x^2 + y^2 - z^2 - 1;
        p0 = {1, 0, 0};
```

Rather than using **analyzeQSNS** we manually find the lines on the hyperboloid through **p0**. Without repeating the work we get parametric lines

```
In[ ]:= l2361a = {1, t, t};
        l2361b = {1, t, -t};
```

These clearly go through point p0 = {1, 0, 0} and checking

```
In[ ]:= f2361 /. Thread[{x, y, z} → l2361a]
        f2361 /. Thread[{x, y, z} → l2361b]

Out[ ]= 0

Out[ ]= 0
```

These lines must lie in the tangent plane

```
In[ ]:= tp2361 = tangentPlaneNS [f2361 , p0, {x, y, z}]

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

Or equivalently, $x-1$ We now, as we have been doing, make this plane invisible with a Transformation function which will actually put these lines in the invisible plane of the target surface. In this case we use a special transformation to keep it as exact as possible but with last row {1, 0, 0, −1} to make $x-1$ invisible.

```
In[ ]:= A2361 = {{0, 1, 1, 0}, {0, -1, 1, 0}, {1, 0, 0, 1}, {1, 0, 0, -1}};
        f2361b = FLTNS[f2361 , A2361 , {x, y, z}]

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

Magically this works perfectly already. This is a saddle surface. We notice what happens to the lines.

```
In[ ]:= pl2361a = fltiMD[l2361a , A2361]

Out[ ]= {2 t, 0, 2, 0}

In[ ]:= pl2361b = fltiMD[l2361b , A2361]

Out[ ]= {0, -2 t, 2, 0}
```

Inversely the transformation matrix taking the saddle surface back to the sphere is

```
In[ ]:= ss2stdHyperboloid  = Inverse[A2361]
```

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

We have seen transformation matrices are homogeneous so this could be multiplied by 2 to get integer coordinates, but then the standard formula would be multiplied by the constant $\frac{1}{4}$.

2.6.2 General Case

We consider the following randomly defined hyperboloid . Some of the intermediate calculations will be suppressed in the interest of readability.

```
In[ • ]:= f2362 = -5.798523022437465` + 4.434386417880354` x +
            3.667824022372237` x^2 - 4.502072645249173` y + 2.7484271965897165` x y -
            1.6804920132021834` y^2 + 3.698654556429698` z - 3.6995461041438222` x z -
            2.747070301170911` y z + 2.4907568140405516` z^2

Out[ • ]= -5.79852 + 4.43439 x + 3.66782 x^2 - 4.50207 y + 2.74843 x y -
           1.68049 y^2 + 3.69865 z - 3.69955 x z - 2.74707 y z + 2.49076 z^2
```

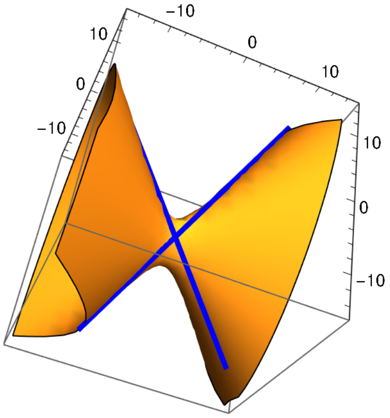
We apply, as usual, **analyzeQSNS**, but to avoid the randomness we just give the answer which does identify this quadric surface as a hyperboloid.

```
In[ • ]:= Lines2362 = {{0.19748140781913295` + 0.6308293338440896` t, -0.4728487245016963` -
              0.8387004948688648` t, -2.2890238330740265` + 0.6933215712245515` t},
            {0.19748140781913295` - 2.9636082441151346` t, -0.4728487245016963` -
              4.785522759124684` t, -2.2890238330740265` - 8.048199525301122` t}}

Out[ • ]= {{0.197481 + 0.630829 t, -0.472849 - 0.8387 t, -2.28902 + 0.693322 t},
           {0.197481 - 2.96361 t, -0.472849 - 4.78552 t, -2.28902 - 8.0482 t}}
```

The plot is

```
In[ • ]:= Show[ContourPlot3D [f2362 == 0, {x, -15, 15}, {y, -15, 15}, {z, -15, 15}, Mesh → None],
           ParametricPlot3D [Lines2362 , {t, -15, 15}, PlotStyle → Blue]]
```

Out[•]=



The intersection point and tangent plane at that point are

```
In[ • ]:= p2362 = pLineIntersectionMD [Lines2362⟦1⟧, Lines2362⟦2⟧, t, {x, y, z}, dTol]

Out[ • ]= {0.197481 , -0.472849 , -2.28902}

In[ • ]:= tp2362 = Expand[tangentPlaneNS [f2362 , p2362 , {x, y, z}] / 3]

Out[ • ]= -5.68628 + 4.3506 x + 1.30601 y - 2.3786 z
```

The main trick is to make this tangent plane invisible by a FLT transformation, in particular the transformation

```
In[ ]:= iT2362 = iTransform3D[tp2362];
        iT2362 // MatrixForm
Out[ ]//MatrixForm=
        ⎛0.494825  -0.572427  -0.653816   1.07695  ⎞
        ⎜0.113209   0.788436  -0.604609   0.0951902⎟
        ⎜0.861587   0.225158   0.454942  -1.09295  ⎟
        ⎝ 1.46962   0.441167  -0.803484  -1.92081  ⎠
```

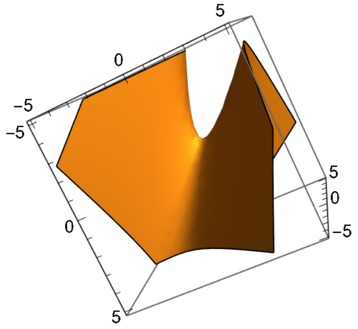
```
In[ ]:= f2362b = FLTNS[f2362, iT2362, {x, y, z}]
Out[ ]= -0.788052 - 0.278701 x + 1.86919 x^2 + 6.96613 y + 1.52586 x y -
         4.45715 y^2 - 0.8878 z + 6.14318 x z - 2.69125 y z + 3.63058 z^2
```

Note that the following points on these lines are invisible (infinite) points and hence the two lines go to invisible lines.

```
In[ ]:= q1 = fltiMD[p2362, iT2362]
        q2 = fltiMD[Lines2362[[1]] /. {t → 1}, iT2362]
        q3 = fltiMD[Lines2362[[2]] /. {t → .1}, iT2362]
Out[ ]= {2.94194, 1.1287, -2.07064, 0}
Out[ ]= {3.28088, 0.119666, -1.40054, 0}
Out[ ]= {3.59543, 1.20444, -2.79987, 0}
```

This surface is a saddle surface .

```
In[ ]:= ContourPlot3D[f2362b == 0, {x, -5, 5},
         {y, -5, 5}, {z, -5, 5}, Mesh → None, ImageSize → Small]
Out[ ]=
```



Here is where we bring in homogeneous coordinates to describe the invisible set of this saddle surface. The saddle surface itself has homogeneous coordinates which we denote by different font letters to emphasize that these coordinates are homogeneous.

```
In[ ]:= F2362h = HomogNS[f2362b, {x, y, z}, {𝕏, 𝕐, ℤ, 𝕎}]
Out[ ]= -0.788052 𝕎^2 - 0.278701 𝕎 𝕏 + 1.86919 𝕏^2 + 6.96613 𝕎 𝕐 +
         1.52586 𝕏 𝕐 - 4.45715 𝕐^2 - 0.8878 𝕎 ℤ + 6.14318 𝕏 ℤ - 2.69125 𝕐 ℤ + 3.63058 ℤ^2
```

The invisible set is then the union of these two lines

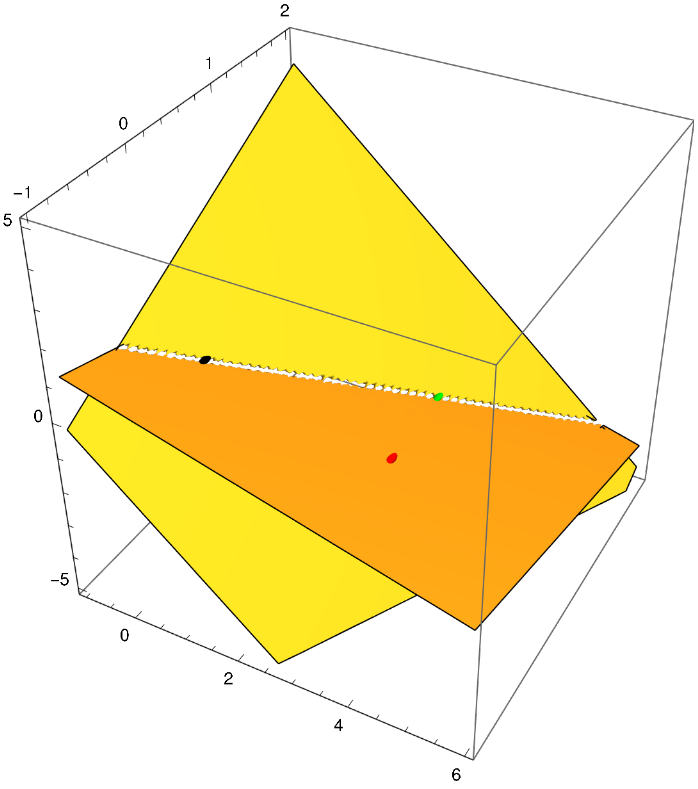
```
In[ ]:= f2362h = F2362h /. {𝕎 → 0}
```

Out[ ]= $0. + 1.86919\ \mathbb{X}^2 + 1.52586\ \mathbb{X}\ \mathbb{Y} - 4.45715\ \mathbb{Y}^2 + 6.14318\ \mathbb{X}\ \mathbb{Z} - 2.69125\ \mathbb{Y}\ \mathbb{Z} + 3.63058\ \mathbb{Z}^2$

We plot this viewing each homogeneous point as an affine line though the non- point {0,0,0}. So the picture of this looks like an affine surface, in fact a degenerate quadric consisting of two planes. One reason we gave the details of this rather obvious case is that we can, to some extent just follow our earlier work, however we must preserve homogeneity and in particular our transformation matrices must have the last column as {{0},{0},{0},{1}}, the final 1 could actually be any non-zero number by homogeneity. In particular we can not use translations which will make it a bit more difficult.

```
In[ ]:= Show[ContourPlot3D[f2362h == 0, {𝕏, -1, 6}, {𝕐, -1, 2}, {ℤ, -5, 5}, Mesh → None,
          MaxRecursion → 3], Graphics3D[{{Red, Ball[Take[q2, 3], .07], Ball[Take[q3, 3], .07]},
          {Green, Ball[Take[q1, 3], .07]}, {Black, Ball[{0, 0, 0}, .07]}}]]
```

Out[ ]=



Note the black ball indicates the irrelevant point, {0,0,0}, as it is sometimes called, the green point is a representative of the point of intersections of this projective lines and the red points represent the image of q2, q3 above.

We can still decompose f2362h as the union of this two planes but we have to think affinely, the results are

```
In[ ]:= plane2362h1 = 3.8439595680212264` 𝕏 + 7.708625536063979` 𝕐 + 9.66340805980829` ℤ
```

```
Out[ ]= 3.84396 𝕏 + 7.70863 𝕐 + 9.66341 ℤ
```

```
In[ ]:= plane2362h2 = 0.4862666360716048` 𝕏 - 0.5782033378346113` 𝕐 + 0.37570414375418054` ℤ
Out[ ]= 0.486267 𝕏 - 0.578203 𝕐 + 0.375704 ℤ
```

Note they are homogeneous , no constant term. We rotate the first plane to the x-plane, I mean the first line to 𝕏=0, again we cheat and work as if affine planes. Our transform is scalar, independent of variables.

```
In[ ]:= Rot1 = planeRotate3D [plane2362h1 /. Thread[{𝕏 , 𝕐 , ℤ} → {x, y, z}], x]
Out[ ]= {{0.296939 , 0.595477 , 0.746481 , 0.}, {-0.595477 , 0.726592 , -0.34274 , 0.},
         {-0.746481 , -0.34274 , 0.570347 , 0.}, {0. , 0. , 0. , 1.}}

In[ ]:= f2362h3 = Chop[FLTNS[f2362h , Rot1 , {𝕏 , 𝕐 , ℤ}], 1.*^-10]
Out[ ]= 1.04262 𝕏^2 - 10.8539 𝕏 𝕐 + 0.640354 𝕏 ℤ

In[ ]:= rotpl2 = Chop[FLTNS[plane2362h2 /. Thread[{𝕏 , 𝕐 , ℤ} → {x, y, z}], Rot1 , {x, y, z}], 1.*^-10]
Out[ ]= 0.0805404 x - 0.838448 y + 0.0494662 z

In[ ]:= u1 = Chop[fltMD[Take[q1, 3], Rot1], 1.*^-10]
Out[ ]= {0, -0.222062 , -3.76393}

In[ ]:= u3 = fltMD[Take[q2, 3], Rot1]
Out[ ]= {-1.37663 × 10^-11 , -1.38672 , -3.28892}
```

We need another rotation to take u1 to a point on the y - axis

```
In[ ]:= Rot2 = m2TM[Chop[RotationMatrix [{u1, {0, 0, -4}}], 1.*^-11 ]]
Out[ ]= {{1. , 0, 0, 0}, {0, 0.998264 , -0.058895 , 0}, {0, 0.058895 , 0.998264 , 0}, {0, 0, 0, 1}}

In[ ]:= f2362h4 = FLTNS[f2362h3 , Rot2 , {𝕏 , 𝕐 , ℤ}]
Out[ ]= 1.04262 𝕏^2 - 10.8728 𝕏 𝕐 + 3.11173 × 10^-11 𝕏 ℤ
```

We find a point on the other component and use a shear to place the second affine plane, (projective line) to 𝕐=0.

```
In[ ]:= Sh2 = {{-10.872815069250905` , 0, 0, 0}, {b, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}}
Out[ ]= {{-10.8728 , 0, 0, 0}, {b, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}}

In[ ]:= f2362h5 = Chop[FLTNS[f2362h4 , Sh2 , {𝕏 , 𝕐 , ℤ}], 1.*^-11]
Out[ ]= 0.00881945 𝕏^2 + 0.0919725 b 𝕏^2 + 1. 𝕏 𝕐
```

So we have transformed the invisible curve of saddle surface **f2362b** to the reducible plane projective curve 𝕏 𝕐. Our big trick, which will be used also in the next section, is that since our transformation functions were actually 3 dimensional they will work on our original affine saddle surface.

```
In[ ]:= f2362c = Chop[FLTNS[f2362b , Sh2.Rot2.Rot1 , {x, y, z}], 1.*^-10]
Out[ ]= -0.788052 - 0.312954 x + 0.522437 b x +
         0.00881945 x^2 + 0.0919725 b x^2 + 5.68036 y + 1. x y - 2.35542 z
```

The basic surface is what we want but we have introduced some unwanted translations that we can remove and some coefficients that can be adjusted. Without going through the details since we have done this before we find the correct translation/homothety that does the trick

```
In[ ]:= T2362 = {{1, 0, 0, 5.680359335436788` }, {0, 1, 0, -0.3630514858797215` },
           {0, 0, 1, -0.5409691513012265` }, {0, 0, 0, 2.35542208613845` }};
        T2362 // MatrixForm
```

```
Out[ ]//MatrixForm=
        ⎛1  0  0   5.68036  ⎞
        ⎜0  1  0  -0.363051 ⎟
        ⎜0  0  1  -0.540969 ⎟
        ⎝0  0  0   2.35542  ⎠
```

```
In[ ]:= FLTNS[f2362c, T2362, {x, y, z}]

Out[ ]= -0.0212691 x - 0.221802 b x + 0.00881945 x^2 + 0.0919725 b x^2 + 1. x y - 1. z
```

which was our goal. Given differently

```
In[ ]:=  A2362 = T2362.Sh2.Rot2.Rot1.iT2362;

                 ⎛ -0.2085799208652234`   0.7465627495536818`  -0.7044539966396092`   -1.2705164
         A2362 = ⎜-0.18615512434182432`   0.6542416058450221`   1.0105022947355933`   -0.9752829
                 ⎜ -1.2120746510033826`   0.6394032181172619`  0.05380172151374432`   -0.6045336
                 ⎝  0.3444419634581145`   1.4443536459572852`   0.0697838356683156`   -0.8308561
```

```
Out[ ]= {{-0.20858, 0.746563, -0.704454, -1.27052}, {-0.186155, 0.654242, 1.0105, -0.975283},
         {-1.21207, 0.639403, 0.0538017, -0.604534}, {0.344442, 1.44435, 0.0697838, -0.830856}}
```

```
In[ ]:= Chop[FLTNS[f2362, A2362, {x, y, z}], 1.*^-10]

Out[ ]= -0.0212691 x - 0.221802 b x + 0.00881945 x^2 + 0.0919725 b x^2 + 1. x y - 1. z
```

Finally, bringing in the transform **ss2stdHyperboloid** we transform our random hyperboloid **f2362** to the standard hyperboloid .

```
In[ ]:= Chop[FLTNS[f2362, ss2stdHyperboloid.A2362, {x, y, z}], 1.*^-10]

Out[ ]= 1. - 1. x^2 + 0.0212691 y + 0.221802 b y - 0.0212691 x y - 0.221802 b x y -
         0.991181 y^2 + 0.0919725 b y^2 + 0.0212691 z + 0.221802 b z - 0.0212691 x z -
         0.221802 b x z + 0.0176389 y z + 0.183945 b y z + 1.00882 z^2 + 0.0919725 b z^2
```

2.7 Rationality of quadric surfaces.

The results of this section show that each non-degenerate quadric surface is a rational surface, since each can be given as a Transformation Function applied to one of the standard types and we know each standard type is rational. It is actually easier to work from our paraboloid and saddle surface as they have obvious parameterizations:

```
In[ ]:= Clear[x, y, z, s, t]
```

In[∘]:=

$$\texttt{paraboloid} := \left\{\frac{2\,t}{1+t\wedge 2}\,s,\ \frac{1-t\wedge 2}{1+t\wedge 2}\,s,\ s\wedge 2\right\}$$

In[∘]:=

```
saddleSurface  := {s, t, s t}
```

In[∘]:=

```
(z - x y) /. Thread[{x, y, z} → saddleSurface ]
```

Out[∘]=

```
0
```

In particular we get the parameterizations

In[∘]:=

```
sphere = Simplify[fltMD[paraboloid , paraboloid2sphere ]];
```

In[∘]:=

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

In[∘]:=

```
hyperboloid  = Simplify[fltMD[saddleSurface , ss2stdHyperboloid ]];
```

In[∘]:=

$$\texttt{hyperboloid} = \left\{\frac{1+s\,t}{-1+s\,t},\ \frac{s-t}{-1+s\,t},\ \frac{s+t}{-1+s\,t}\right\};$$

2.8 Transitivity of symmetries of non-singular quadric surfaces.

Perhaps you have noticed that if one point on a non-singular quadric surface , ellipsoid or hyperboloid, lies on no lines in the conic then this is true for all points. Also with hyperboloids if one point on a hyperboloid lies on two lines then all points share this property. This is because projective linear transformations, in particular FLT's, are *transitive* on these surfaces, that is given two points on the surface there is at least one such transformation taking the first point to the second.

For ellipsoids this is now obvious, since the standard example is the unit sphere about the origin and any point on the sphere can be rotated to any other point with a FLT rotation, in fact a linear one. This is not so obvious for the hyperboloid.

But note that for the the random hyperboloid in 2.3.6 the random point is mapped by our transformation to the point {1,0,0} in the standard hyperboloid.

In[∘]:=

```
fltMD[p2362, ss2stdHyperboloid .A2362]
```

Out[∘]=

```
{1., -0.530437 × (0.219697 × (-15.2929 + 1. b) - 0.245577 × (-10.5376 + 1. b) -
      0.0846706 × (-0.0813494 + 1. b) + 0.110551 × (6.921 + 1. b)),
  -0.530437 × (-0.110551 × (-4.10603 + 1. b) + 0.0846706 × (-2.35658 + 1. b) +
      0.245577 × (10.2585 + 1. b) - 0.219697 × (12.6249 + 1. b))}
```

So following this method we expect that any given point **p** on any hyperboloid we can find a FLT taking that hyperboloid to the standard one with **p** going to {1,0,0}. Thus if **p, q** are points on a given hyperboloid there are FLT transformation F1, F2 taking **p, q** to {1,0,0}. But then $\mathbf{F2^{-1}.F1}$ takes **p** to **q.**

As with ellipsoids it is enough to illustrate on the standard hyperboloid. The rational transformation

```
In[ ]:= hyperboloid
```

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

gives a pseudo-random rational point on the standard hyperboloid chosen for a nice plot below

```
In[ ]:= psh = hyperboloid /. {s → 3, t → 33 / 50}
```

Out[ ]= $\left\{\frac{149}{49}, \frac{117}{49}, \frac{183}{49}\right\}$

In[ ]:= psh = $\text{N}\left[\left\{\frac{149}{49}, \frac{117}{49}, \frac{183}{49}\right\}\right]$

```
Out[ ]= {3.04082 , 2.38776 , 3.73469}
```

We use the method of 2.3.6.2 to find a FLT symmetry of this standard hyperboloid which takes psh to {1,0,0}.

```
In[ ]:= B2362 = {{0.13816845787030158` , 0.12331372237686955` ,
           -0.8029079916135967` , 0.2281668087676449` }, {-0.49454148501616907` ,
           -0.43338569397869237` , 0.42355638183243205` , 0.9567752977057387` },
         {-0.466647613790002` , 0.6693815164894659` , -0.03563958055734917` ,
           -0.046226525154333176` }, {-0.8416212606866844` ,
           -0.6460513586376766` , 0.40045790433223194` , 0.5503795350045073` }};
        B2362 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0.138168 & 0.123314 & -0.802908 & 0.228167 \\ -0.494541 & -0.433386 & 0.423556 & 0.956775 \\ -0.466648 & 0.669382 & -0.0356396 & -0.0462265 \\ -0.841621 & -0.646051 & 0.400458 & 0.55038 \end{pmatrix}$$

Check

```
In[ ]:= h1 = FLTNS[x ^ 2 + y ^ 2 - z ^ 2 - 1, B2362 , {x, y, z}];
        h1 = Expand[h1 / Coefficient [h1, x ^ 2]]
```

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

```
In[ ]:= fltMD[psh, B2362]
```

Out[ ]= $\left\{1., -4.32023 \times 10^{-16}, 1.75509 \times 10^{-16}\right\}$

This works! The reader, however, should be beware that numerical issues can arise if these points are two close together so I am not attempting a black box algorithm to find all such transformations . The transitivity property should be considered theoretical rather than algorithmic .

Thus B2362 is a projective symmetry of the standard hyperbola

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

because the last row is not {0,0,0,1}, with inverse

```
In[ ]:= A2362 = Inverse[B2362];
        A2362 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -0.20858 & 0.746563 & -0.704454 & -1.27052 \\ -0.186155 & 0.654242 & 1.0105 & -0.975283 \\ -1.21207 & 0.639403 & 0.0538017 & -0.604534 \\ 0.344442 & 1.44435 & 0.0697838 & -0.830856 \end{pmatrix}$$

This takes {1, 0, 0} to

```
In[ ]:= psh = fltMD[{1, 0, 0}, A2362]
Out[ ]= {3.04082, 2.38776, 3.73469}
```

It is interesting to see how this transformation really works. It is easier to look at the transform of curve on the surface rather than just points. So consider the parametric circle where h intersects the $z = 0$ plane

In[ ]:=

$$\text{circ} = \left\{\frac{2\,t}{1+t^2}, \frac{1-t^2}{1+t^2}, 0\right\};$$

We might expect the image to be the horizontal circle through **psh**.

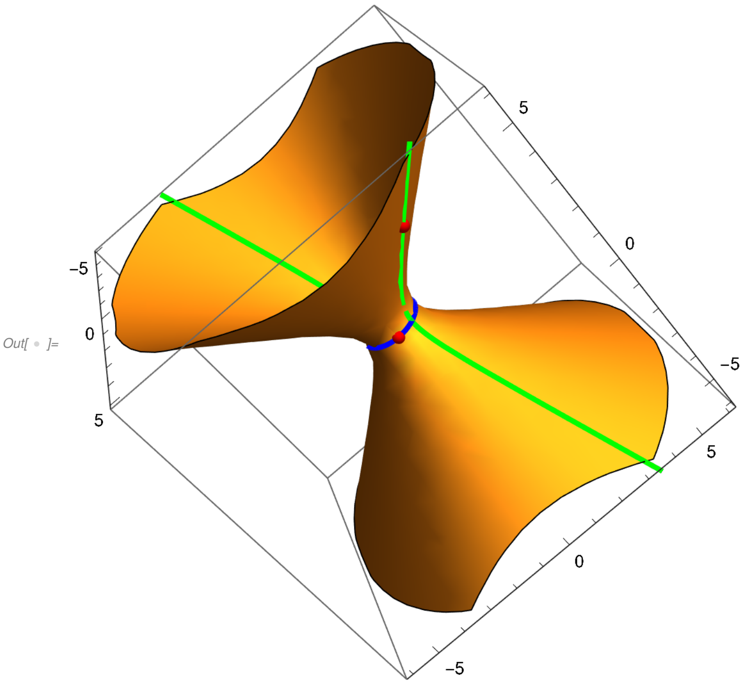
```
In[ ]:= circA = Simplify[fltMD[circ, A2362]]
```

Out[ ]=

$$\left\{\frac{0.230288 + 0.18335\,t + 0.886546\,t^2}{-0.269644 - 0.302778\,t + 1.\,t^2},\right.$$
$$\left.\frac{0.141104 + 0.163638\,t + 0.716208\,t^2}{-0.269644 - 0.302778\,t + 1.\,t^2}, \frac{0.0153259 - 1.06546\,t - 0.546735\,t^2}{0.269644 + 0.302778\,t - 1.\,t^2}\right\}$$

```
In[ ]:= Show[ContourPlot3D [h == 0, {x, -5, 5}, {y, -6, 6}, {z, -6, 6}, Mesh → None],
         ParametricPlot3D [{circ, circA}, {t, -10, 10}, PlotStyle → {Blue, Green}],
         Graphics3D [{Red, Ball[psh, .2], Ball[{1, 0, 0}, .2]}]]
```

Out[ ]=



Instead we get a vertical plane hyperbola through the point **psh**.

Being used to rigid motions of the quadric surfaces it is hard to picture a motion that does this. So we should not think of projective transformations as motions.

2.9 Affine and Projective Symmetries of Quadric Surfaces

The example above shows that our main theorem implies that the symmetry group of a quadric surface is isomorphic to the symmetry group of our standard example even though they have distinct Euclidean symmetries.

There are several ways to find symmetries, one, like above is to construct, using the constructions above but with two different points two different projective linear equivalences A1, A2 from quadric **Q1** to quadric **Q2**. Then **A1.Inverse[A2]** is a symmetry of Q1.

On the other hand if matrix **S** gives a symmetry on one of our standard quadrics in our chart and $A:S\longrightarrow Q$ is a projective equivalence then **S1 = A.S.Inverse[A]** is a symmetry on **Q.** So once we know the symmetry groups of the standard quadrics we know the symmetry groups of all quadric surfaces.

Finally, starting from known isometries, that is linear symmetries of our standard quadrics and perhaps examples as constructed above, we can deduce certain symmetries of the standard quadrics which generate the symmetry groups.

2.9.1 Ellipsoids, Cones and Cylinders

A simple example is an ellipsoid with the coordinate axes as axes so the transform to the sphere is just a homothety.

```
In[ ]:= ell = x^2 + 4 y^2 + 4 z^2 - 16;
        ell2sphere = {{1, 0, 0, 0}, {0, 2, 0, 0}, {0, 0, 2, 0}, {0, 0, 0, 4}};
```

```
In[ ]:= FLTNS[ell, ell2sphere, {x, y, z}]
```

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

An obvious circle on the ellipsoid is the vertical circle is given parametrically by

In[ ]:= $$\text{ecirc} = \left\{0, \frac{4\,t}{1+t^2}, \frac{2\times(1-t^2)}{1+t^2}\right\};$$

Let

```
In[ ]:= R45 = {{0.7071067811865475` , -0.7071067811865475` , 0.` , 0},
          {0.7071067811865475` , 0.7071067811865475` , 0.` , 0}, {0.` , 0.` , 1.` , 0}, {0, 0, 0, 1}}

Out[ ]= {{0.707107 , -0.707107 , 0. , 0}, {0.707107 , 0.707107 , 0. , 0}, {0. , 0. , 1. , 0}, {0, 0, 0, 1}}
```

be a 45^o rotation about the z-axis which is a Euclidean symmetry of the sphere.

Then

```
In[ ]:= R45ell = Inverse[ell2sphere].R45.ell2sphere ;
        R45ell // MatrixForm

Out[ ]//MatrixForm=
        ( 0.707107  -1.41421  0.  0. )
        ( 0.353553  0.707107  0.  0. )
        (    0.        0.     1.  0. )
        (    0.        0.     0.  1. )
```

Note

```
In[ ]:= Det[R45ell]

Out[ ]= 1.
```

so this is a "rotation" .

If the reader has not already figured it out, the symmetries of a surface are given by exactly those invertible 4*4 matrices which fix FLTNS on the surface . Points on the surface remain on the surface, but are not pointwise fixed . Invertibility insures this transformation is 1 - 1 and onto this surface .

```
In[ ]:= FLTNS[ell, R45ell, {x, y, z}]
```

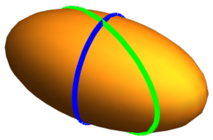
Out[ ]= $-16. + 1.\ x^2 + 4.\ y^2 + 4.\ z^2$

But the action of R45ell on the circle is

```
In[ ∘ ]:= circr45 = Chop[fltMD[ecirc, R45ell]]

            5.65685 t   2.82843 t   2. × (1 - t²)
Out[ ∘ ]= {- ---------, ---------, -------------}
             1 + t²      1 + t²       1 + t²
```

```
In[ ∘ ]:= Show[ContourPlot3D [ell == 0, {x, -4, 4}, {y, -4, 4}, {z, -4, 4}, Mesh → None],
           ParametricPlot3D [{ecirc, circr45}, {t, -10, 10}, PlotStyle → {Blue, Green}],
           Axes → None, Boxed → False, ImageSize → Small]
```

Out[∘]=



This symmetry is not just moving the circle, but the entire ellipsoid. From a Euclidean point of view this ellipsoid would only have 45^o rotations, and other arbitrary rotations, about the major axis. We would have 180^o rotations and reflections about the minor axes but no others. But here we have an affine rotation of arbitrary angle about any line through the origin.

We note that we don’t really need our big theorem. The transform to the circle is just a homothety.

Generalizing from this discussion we see that the symmetry group of any ellipsoid is the orthogonal group $\mathbb{O}(4)$.

In the case of a cone consider the symmetry

```
In[ ∘ ]:= ssCone = {{1.7320508075688772` , 0.` , 1.4142135623730951` , 0.`},
            {2.` , 1.7320508075688772` , 2.449489742783178` , 0.`},
            {2.449489742783178` , 1.4142135623730951` , 3.` , 0.`}, {0.` , 0.` , 0.` , 1.`}}

Out[ ∘ ]= {{1.73205 , 0. , 1.41421 , 0.}, {2. , 1.73205 , 2.44949 , 0.},
           {2.44949 , 1.41421 , 3. , 0.}, {0. , 0. , 0. , 1.}}
```

```
In[ ∘ ]:= ssCone // MatrixForm

Out[ ∘ ]//MatrixForm=
          ( 1.73205     0.      1.41421   0. )
          (   2.     1.73205    2.44949   0. )
          ( 2.44949  1.41421      3.      0. )
          (   0.        0.        0.      1. )
```

```
In[ ∘ ]:= FLTNS[x ^ 2 + y ^ 2 - z ^ 2, ssCone, {x, y, z}]

Out[ ∘ ]= 1. x² + 1. y² - 1. z²
```

Note that this is actually a linear transformation .

In[ ]:=

$$\text{circ1} = \left\{\frac{2\,t}{1+t^2}, \frac{1-t^2}{1+t^2}, 1\right\};$$

```
circ1ss = fltMD[circ1, ssCone]
```

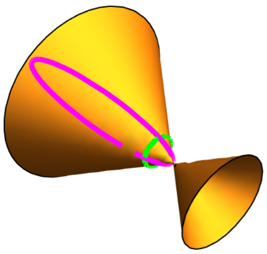
Out[ ]=

$$\left\{1.41421 + \frac{3.4641\ t}{1+t^2},\ 2.44949 + \frac{4.\ t}{1+t^2} + \frac{1.73205 \times (1-t^2)}{1+t^2},\ 3. + \frac{4.89898\ t}{1+t^2} + \frac{1.41421 \times (1-t^2)}{1+t^2}\right\}$$

In[ ]:=

```
Show[ContourPlot3D [x ^ 2 + y ^ 2 == z ^ 2, {x, -6, 6}, {y, -6, 6}, {z, -3, 6}, Mesh → None],
 ParametricPlot3D [{circ1, circ1ss}, {t, -20, 20}, PlotStyle → {Green, Magenta}],
 ImageSize → Small, Axes → False, Boxed → False]
```

Out[ ]=



In the case of a cone consider the symmetry of the cylinder $x^2 + y^2 - z^2$ moving ssCone to the cylinder by **CC3**.

In[ ]:=

```
sscyl = CC3.ssCone.Inverse[CC3]
```

Out[ ]=

```
{{1.73205, 0., 0., 1.41421}, {2., 1.73205, 0., 2.44949},
 {0., 0., 1., 0.}, {2.44949, 1.41421, 0., 3.}}
```

In[ ]:=

```
sscyl // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1.73205 & 0. & 0. & 1.41421 \\ 2. & 1.73205 & 0. & 2.44949 \\ 0. & 0. & 1. & 0. \\ 2.44949 & 1.41421 & 0. & 3. \end{pmatrix}$$

In[ ]:=

```
FLTNS[x ^ 2 + y ^ 2 - 1, sscyl, {x, y, z}]
```

Out[ ]=

$$-1. + 1.\ x^2 + 1.\ y^2$$

Note that unlike the cone, this is a projective transformation .

We can still use circ1 as it is also on the cone

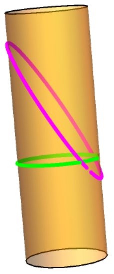
In[ ]:=

```
cyl1ss = Simplify[fltMD[circ1, sscyl]]
```

Out[ ]=

$$\left\{\frac{0.891806 + 2.18447\ t + 0.891806\ t^2}{2.78361 + 3.08931\ t + 1.\ t^2},\ \frac{2.63689 + 2.52241\ t + 0.452418\ t^2}{2.78361 + 3.08931\ t + 1.\ t^2},\ \frac{0.630602 + 0.630602\ t^2}{2.78361 + 3.08931\ t + 1.\ t^2}\right\}$$

```
Show[ContourPlot3D [x ^ 2 + y ^ 2 == 1, {x, -3, 3},
   {y, -3, 3}, {z, -3, 7}, Mesh → None, ContourStyle → Opacity[.5]],
  ParametricPlot3D [{circ1, cyl1ss}, {t, -20, 20}, PlotStyle → {Green, Magenta}],
  ImageSize → Small, Axes → False, Boxed → False]
```



2.9.2 The group of symmetries of the Hyperbola

Given the huge amount of material online about hyperboloids as of this writing I have been unable to find a source giving symmetries of the real hyperboloid. It may be that this is too complicated and possibly not completely known. I don’t know the full story but given the above analysis I can say some things.

We start with the easily described Euclidean geometry . The equation of the standard hyperboloid is

```
h = x ^ 2 + y ^ 2 - z ^ 2 - 1;
```

The obvious symmetries of the standard hyperbola are rotations about the z-axis as well as reflections through planes containing the z-axis. In fact these are all isometries of the circle extended to 3 space. There is also a horizontal reflection in the xy-plane. Finally from the symmetry A2362 found in section 2.3.6 as well as looking at symmetries of the saddle surface we find a simpler example of a rotation of order 2 of the projective hyperboloid I will call the half turn. For the reader’s reference here, and in Global Functions, is a summary. Note we give two versions of the rotations about the z-axis. For more information on the rotations and reflections see the Mathematica documentation. All of these give orthogonal 4×4 matrices, but note that since these will be used in TransformationFunctions they will not all give geometrically orthogonal transformations.

```
thetaR3D := N[m2TM[RotationMatrix [#, {0, 0, 1}]]] &
pRot3D[p_, q_] :=
 If[p⟦3⟧ == 0 && q⟦3⟧ == 0, N[m2TM[RotationMatrix [{p, q}]]], Echo["invalid points"];
  Abort[]]
vReflect3D := N[m2TM[ReflectionMatrix [#]]] &
hReflect3D := {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, -1, 0}, {0, 0, 0, 1}};
halfTurn := {{0, 0, 0, -1}, {0, 0, 1, 0}, {0, 1, 0, 0}, {-1, 0, 0, 0}}
```

Note all of these are symmetries of the standard hyperbola, the first 4 are obvious and the last is verified by

```
In[ ]:= FLTNS[h, halfTurn , {x, y, z}]
```

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

which sends equation h to - h.

2.9.3 The Group ℍ𝕆(4)

Looking at combinations of the above and their inverses we are lead to the following subgroup of the real orthogonal group 𝕆(4) consisting of block matrices

```
In[ ]:= {{{B₁ , 0}, {0 , B₂}} // MatrixForm ,  {{0 , B₁}, {B₂ , 0}} // MatrixForm }
```

Out[ ]= $$\left\{\begin{pmatrix} B_1 & 0 \\ 0 & B_2 \end{pmatrix}, \begin{pmatrix} 0 & B_1 \\ B_2 & 0 \end{pmatrix}\right\}$$

where B_1, B_2 are 2×2 orthogonal matrices. I call the set of all these matrices ℍ𝕆(4) and we will see that these form a subgroup and are all symmetries of the hyperboloid.

A good way of seeing what these symmetries do is to look at their action on the unit circle in the xy plane which lies on the hyperboloid.

$$\text{circ} = \left\{\frac{2\,t}{1+t^2}, \frac{1-t^2}{1+t^2}, 0\right\};$$

Out[ ]= $$\left\{\frac{2\,t}{1+t^2}, \frac{1-t^2}{1+t^2}, 0\right\}$$

The functions, here and in GlobalFunctions,

```
In[ ]:= RHl3D := Module[{rr, R},
          rr = RandomReal [{-1, 1}, 8];
          Orthogonalize [SparseArray [{{1, 1} → rr⟦1⟧, {1, 2} → rr⟦2⟧, {2, 1} → rr⟦3⟧,
              {2, 2} → rr⟦4⟧, {3, 3} → rr⟦5⟧, {3, 4} → rr⟦6⟧, {4, 3} → rr⟦7⟧, {4, 4} → rr⟦8⟧}]]]
        RHr3D := Module[{rr, R},
          rr = RandomReal [{-1, 1}, 8];
          Orthogonalize [SparseArray [{{1, 3} → rr⟦1⟧, {1, 4} → rr⟦2⟧, {2, 3} → rr⟦3⟧,
              {2, 4} → rr⟦4⟧, {3, 1} → rr⟦5⟧, {3, 2} → rr⟦6⟧, {4, 1} → rr⟦7⟧, {4, 2} → rr⟦8⟧}]]]
```

give random examples . Note they will differ each time they run, for example the following are different

```
In[ ]:= MatrixPower[RHl3D, 2] // MatrixForm
        RHl3D.RHl3D // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1. & 0. & 0. & 0. \\ 0. & 1. & 0. & 0. \\ 0. & 0. & 0.119376 & -0.992849 \\ 0. & 0. & 0.992849 & 0.119376 \end{pmatrix}$$

Out[ ]//MatrixForm=

$$\begin{pmatrix} -0.980313 & 0.197449 & 0. & 0. \\ -0.197449 & -0.980313 & 0. & 0. \\ 0. & 0. & -0.711246 & 0.702943 \\ 0. & 0. & 0.702943 & 0.711246 \end{pmatrix}$$

So two random examples are (non evaluative)

```
In[ ]:= L1 = ( 0.2951434768979358`   0.9554529439195828`             0.`                    0.`
               0.9554529439195829`  -0.29514347689793585`            0.`                    0.`
                       0.`                   0.`           0.20263181172652914`   0.979254996860
                       0.`                   0.`          -0.979254996860585`    0.2026318117265

Out[ ]= {{0.295143 , 0.955453 , 0. , 0.}, {0.955453 , -0.295143 , 0. , 0.},
         {0. , 0. , 0.202632 , 0.979255}, {0. , 0. , -0.979255 , 0.202632}}
```

```
In[ ]:= R1 = (          0.`                     0.`             0.7698936188976082`   0.6381722460
                        0.`                     0.`             0.6381722460125827`  -0.7698936188
               0.007857413066204117`   0.9999691300534768`              0.`                   0.`
                0.9999691300534768`  -0.007857413066204117`             0.`                   0.`

Out[ ]= {{0. , 0. , 0.769894 , 0.638172}, {0. , 0. , 0.638172 , -0.769894},
         {0.00785741 , 0.999969 , 0. , 0.}, {0.999969 , -0.00785741 , 0. , 0.}}
```

Note

```
In[ ]:= FLTNS[h, L1, {x, y, z}]
        FLTNS[h, R1, {x, y, z}]
```

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

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

```
In[ ]:= circL = Simplify[fltMD[circ, L1]]
        circR = Simplify[fltMD[circ, R1]]
```

Out[ ]=

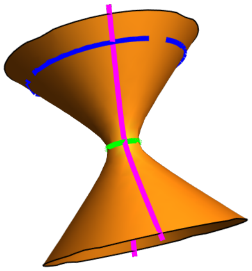
$$\left\{\frac{4.71522 + 2.9131\ t - 4.71522\ t^2}{1. + t^2},\ \frac{-1.45655 + 9.43043\ t + 1.45655\ t^2}{1. + t^2},\ 4.83268\right\}$$

Out[ ]=

$$\left\{\frac{81.2191 + 81.2191\ t^2}{-1. + 254.529\ t + 1.\ t^2},\ \frac{97.9831 + 97.9831\ t^2}{1. - 254.529\ t - 1.\ t^2},\ \frac{127.264 + 2.\ t - 127.264\ t^2}{-1. + 254.529\ t + 1.\ t^2}\right\}$$

```
In[ ]:= Show[ContourPlot3D [h == 0, {x, -6, 6}, {y, -6, 6}, {z, -6, 6}, Mesh → None],
         ParametricPlot3D [{circ, circL, circR},
          {t, -15, 15}, PlotStyle → {Green, Blue, Magenta}],
         ImageSize → Small, Axes → False, Boxed → False, ImageSize → Medium]
```

Out[ ]=



So the left type send the base circle to another horizontal circle while the right type sends it to a vertical hyperbola.

We note that $\mathbb{HO}(4)$ is large enough to already be transitive on points. For example consider our pseudo random point

```
In[ ]:= psh = {3.0408163265306136` , 2.387755102040818` , 3.734693877551021` }
Out[ ]= {3.04082 , 2.38776 , 3.73469}
```

Using circR above

```
In[ ]:= Solve[circR⟦3⟧ == psh⟦3⟧, t]
```

```
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 .
```

```
Out[ ]= {{t → -7.37673}, {t → 0.135561}}
```

we have

```
In[ ]:= q0 = circ /. {t → 0.13556133358205716` }
Out[ ]= {0.26623 , 0.963909 , 0}

In[ ]:= R3 = pRot3D[{1, 0, 0}, q0]
Out[ ]= {{0.26623 , -0.963909 , 0. , 0.},
         {0.963909 , 0.26623 , 0. , 0.}, {0. , 0. , 1. , 0.}, {0. , 0. , 0. , 1.}}

In[ ]:= q1 = fltMD[{1, 0, 0}, R1.R3]
Out[ ]= {2.46734 , -2.97661 , 3.73469}
```

```
In[ ]:= B1 = pRot3D[ReplacePart[q1, 3 → 0], ReplacePart[psh, 3 → 0]]

Out[ ]= {{0.026446 , -0.99965 , 0. , 0.},
         {0.99965 , 0.026446 , 0. , 0.}, {0. , 0. , 1. , 0.}, {0. , 0. , 0. , 1.}}
```

```
In[ ]:= fltMD[q1, B1]

Out[ ]= {3.04082 , 2.38776 , 3.73469}
```

So if

```
In[ ]:= B2 = B1.R1.R3;
        B2 // MatrixForm

Out[ ]//MatrixForm=
        ⎛   0.         0.      -0.617588  0.786501 ⎞
        ⎜   0.         0.       0.786501  0.617588 ⎟
        ⎜0.965972   0.258648      0.         0.     ⎟
        ⎝0.258648  -0.965972      0.         0.     ⎠
```

```
In[ ]:= fltMD[{1, 0, 0}, B2]

Out[ ]= {3.04082 , 2.38776 , 3.73469}
```

which shows a member of ℍ𝕆(4) sending {1,0,0} to our pseudo-random point **psh.**

2.9.4 Another set of Symmetries of the Hyperboloid.

More experimentation with the constructions in 2.3.10 show there are additional symmetries of the hyperboloid, somewhat like our symmetries of the non-spherical ellipsoid. Here is a continuous 1 parameter family sshyp[u] of strange linear symmetries for u ≥ 1.

of linear symmetries. Here we assume u is real, u ≥ 1.

```
In[ ]:= sshyp3D[u_] := {{√u , 0, √(-1 + u) , 0},
           {-1 + u, √u , √(-1 + u) √u , 0}, {√(-1 + u) √u , √(-1 + u) , u, 0}, {0, 0, 0, 1}}
```

```
In[ ]:= sshyp3D[u] // MatrixForm

Out[ ]//MatrixForm=
```

$$\begin{pmatrix} \sqrt{u} & 0 & \sqrt{-1+u} & 0 \\ -1+u & \sqrt{u} & \sqrt{-1+u}\,\sqrt{u} & 0 \\ \sqrt{-1+u}\,\sqrt{u} & \sqrt{-1+u} & u & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Note

```
In[ ]:= Det[sshyp3D[u]]

Out[ ]= 1
```

```
In[ ]:= FLTNS[h, sshyp3D[u], {x, y, z}]
```

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

So these are all symmetries of the hyperbola with determinant 1 but not in ℍ𝕆(4). Note **sshyp[1]** is just the identity symmetry. Here is an example for **u = 3.**

```
In[ • ]:= sshyp3D[3] // MatrixForm
```

Out[•]//MatrixForm=

$$\begin{pmatrix} \sqrt{3} & 0 & \sqrt{2} & 0 \\ 2 & \sqrt{3} & \sqrt{6} & 0 \\ \sqrt{6} & \sqrt{2} & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

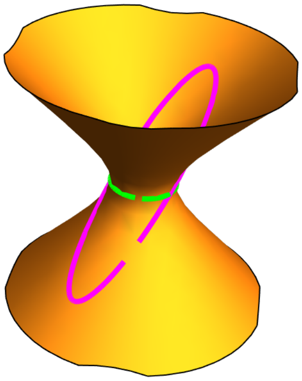
```
In[ • ]:= circss3 = Simplify[fltMD[circ, sshyp3D[3]]]
```

Out[•]=

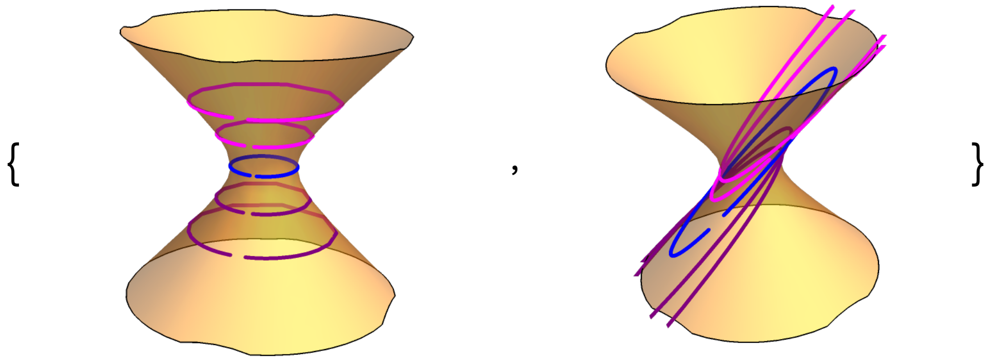
$$\left\{\frac{2\sqrt{3}\,t}{1+t^2}, \frac{\sqrt{3}+4\,t-\sqrt{3}\,t^2}{1+t^2}, \frac{\sqrt{2}\left(1+2\sqrt{3}\,t-t^2\right)}{1+t^2}\right\}$$

```
In[ • ]:= Show[ContourPlot3D[h == 0, {x, -4, 4}, {y, -4, 4}, {z, -4, 4}, Mesh → None],
           ParametricPlot3D[{circ, circss3}, {t, -20, 20}, PlotStyle → {Green, Magenta}],
           ImageSize → Small, Axes → False, Boxed → False]
```

Out[•]=



Alternatively we may show the action of the transformation of the hyperboloid by drawing several curves. I will suppress the code but the transformation sshyp3D[3] takes the curves in the left plot to those in the right, pushing one side of the hyperboloid up and the other down.



Note the plane containing the conic **circss3** is

```
In[ ∘ ]:= FLTNS[z, sshyp[3], {x, y, z}]
```

Out[∘]= $-\sqrt{2}\ x-\sqrt{6}\ y+3\ z$

which passes through the origin which must happen since we have a linear transformation. To move this away from the origin we can compose this transformation with a left type transformation from ℍ𝕆(4). Here are several examples

```
In[ ∘ ]:= ru = Sort[RandomReal[{1, 12}, 3]]
Out[ ∘ ]= {4.84493, 7.78546, 10.3032}
```

```
In[ ∘ ]:= randho4 = Table[RHl3D, {3}];
          Table[randho4⟦i⟧ // MatrixForm, {i, 3}]
```

```
           ⎛0.960738  -0.277458    0.         0.      ⎞
Out[ ∘ ]= {⎜0.277458   0.960738    0.         0.      ⎟,
           ⎜   0.         0.     0.990014  -0.140968  ⎟
           ⎝   0.         0.     0.140968   0.990014  ⎠

           ⎛-0.967219  0.253942    0.          0.      ⎞
           ⎜ 0.253942  0.967219    0.          0.      ⎟,
           ⎜    0.        0.     0.874658  -0.484741   ⎟
           ⎝    0.        0.    -0.484741  -0.874658   ⎠

           ⎛ 0.500231  -0.865892    0.         0.      ⎞
           ⎜-0.865892  -0.500231    0.         0.      ⎟}
           ⎜    0.         0.     0.179668  -0.983727  ⎟
           ⎝    0.         0.     0.983727   0.179668  ⎠
```

```
In[ ∘ ]:= Table[randho4⟦i⟧.sshyp[ru⟦i⟧] // MatrixForm, {i, 3}]
```

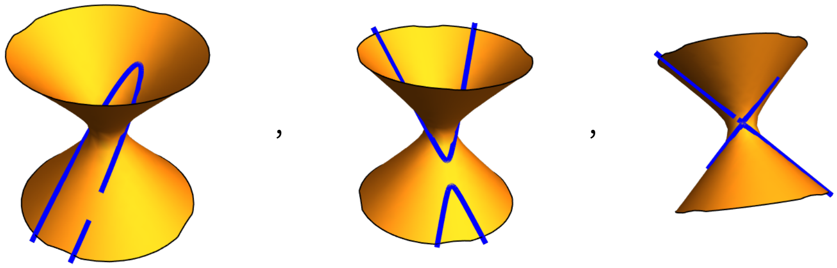
```
           ⎛ 1.04789  -0.61072  0.686333     0.      ⎞  ⎛-0.975666  0.70856  -0.673781     0.      ⎞
Out[ ∘ ]= {⎜ 4.30469   2.1147   4.69066      0.      ⎟, ⎜ 7.27159   2.69878   7.69152      0.      ⎟,
           ⎜ 4.27297   1.94127  4.79655  -0.140968   ⎟  ⎜ 6.35726   2.27839   6.80962  -0.484741   ⎟
           ⎝0.608429  0.276418  0.682983  0.990014   ⎠  ⎝-3.52324  -1.2627   -3.77393  -0.874658   ⎠

           ⎛-6.44992  -2.7794   -6.95173     0.      ⎞
           ⎜-7.43317  -1.60568  -7.53858     0.      ⎟}
           ⎜ 1.75903   0.548008  1.85116  -0.983727  ⎟
           ⎝ 9.63116   3.00049  10.1356    0.179668  ⎠
```

In[ ]:=
```
parss = Table[Together[Simplify[fltMD[circ, randho4[[i]].sshyp[ru[[i]]]]]], {i, 3}]
```

Out[ ]=
$$\left\{\left\{\frac{0.855834 \times (-1. + 3.43166\ t + 1.\ t^2)}{1.77472 + 1.70525\ t + 1.\ t^2},\right.\right.$$
$$\left.-\frac{2.96344 \times (-1. - 4.07121\ t + 1.\ t^2)}{1.77472 + 1.70525\ t + 1.\ t^2}, -\frac{2.91795 \times (-0.864599 - 4.10421\ t + 1.\ t^2)}{1.77472 + 1.70525\ t + 1.\ t^2}\right\},$$
$$\left\{-\frac{1.826 \times (-1. + 2.75394\ t + 1.\ t^2)}{-5.50808 - 18.1591\ t + 1.\ t^2}, -\frac{6.95489 \times (-1. - 5.3888\ t + 1.\ t^2)}{-5.50808 - 18.1591\ t + 1.\ t^2},\right.$$
$$\left.-\frac{7.12073 \times (-0.649136 - 4.60149\ t + 1.\ t^2)}{-5.50808 - 18.1591\ t + 1.\ t^2}\right\}, \left\{-\frac{0.985315 \times (-1. - 4.64124\ t + 1.\ t^2)}{-1.12739 - 6.82863\ t + 1.\ t^2},\right.$$
$$\left.\left.-\frac{0.569223 \times (-1. - 9.25862\ t + 1.\ t^2)}{-1.12739 - 6.82863\ t + 1.\ t^2}, \frac{0.543011 \times (0.284461 - 2.29678\ t + 1.\ t^2)}{-1.12739 - 6.82863\ t + 1.\ t^2}\right\}\right\}$$

In[ ]:=
```
Table[Show[Show[ContourPlot3D [h == 0, {x, -5, 5}, {y, -5, 5}, {z, -5, 5}, Mesh → None],
    ParametricPlot3D [parss[[i]], {t, -30, 30}, PlotStyle → Blue],
    Axes → False, Boxed → False, ImageSize → Small]], {i, 3}]
```

Out[ ]= {



}

Here is another example

```
S4 = RHl3D.sshyp[7.3].RHr3D
```

In[ ]:=
```
S4 = {{5.765471472798233` , -0.2088666170455097` , 2.9114996070358705` ,
    5.080103995549325` }, {4.356696328800707` , -0.15783070438982646` ,
    3.2175957450713106` , 3.10689411694907` }, {-4.769383016337672` ,
    0.9267231581213656` , -2.776345846324982` , -3.8597582709183604` },
   {5.52036682191551` , 0.4579519441664014` , 3.181455025937826` , 4.422953057585654` }}
```

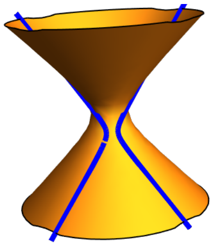
Out[ ]=
```
{{5.76547 , -0.208867 , 2.9115 , 5.0801}, {4.3567 , -0.157831 , 3.2176 , 3.10689},
 {-4.76938 , 0.926723 , -2.77635 , -3.85976}, {5.52037 , 0.457952 , 3.18146 , 4.42295}}
```

In[ ]:=
```
circS4 = Together[FullSimplify [fltMD[circ, S4]]]
```

Out[ ]=
$$\left\{\frac{1.33391 \times (0.921018 + 2.18019\ t + 1.\ t^2)}{1.231 + 2.78455\ t + 1.\ t^2},\right.$$
$$\left.\frac{0.823386 \times (0.903311 + 2.66895\ t + 1.\ t^2)}{1.231 + 2.78455\ t + 1.\ t^2}, -\frac{1.20718 \times (0.612775 + 1.99286\ t + 1.\ t^2)}{1.231 + 2.78455\ t + 1.\ t^2}\right\}$$

```
In[ ]:= Show[Show[ContourPlot3D [h == 0, {x, -5, 5}, {y, -5, 5}, {z, -5, 5}, Mesh → None],
          ParametricPlot3D [circA , {t, -30, 30}, PlotStyle → Blue],
          Axes → False , Boxed → False , ImageSize → Small]]
```

Out[ ]=



I conjecture that all projective symmetries of the standard hyperboloid can be obtained this way but don't have a good argument at this time.

As mentioned above the symmetries of the standard hyperboloid generate symmetries of all hyperboloids. Recall that

```
In[ ]:= sss2h = ss2stdHyperboloid
```

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

is linear projective equivalence from the saddle surface to the standard hyperboloid. Thus

```
In[ ]:= sr3 = N[Inverse[ss2h].sshyp3D[3].ss2h];
        sr3 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 4.29788 & 1.15161 & 2.22474 & 2.22474 \\ 0.116337 & 0.434174 & 0.224745 & 0.224745 \\ 0.707107 & 0.707107 & 1.36603 & 0.366025 \\ 0.707107 & 0.707107 & 0.366025 & 1.36603 \end{pmatrix}$$

is a symmetry of the saddle surface :

```
In[ ]:= FLTNS[z - x y, sr3 , {x, y, z}]
```

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

Of course, here we have a projective symmetry rather than a linear symmetry. Some of the numbers may look familiar, this can be expressed exactly using $\sqrt{2}$, $\sqrt{3}$, and $\sqrt{6}$.

Consider the curve on $z - x\,y$ given by

```
f = {t, -t, -t^2}
```

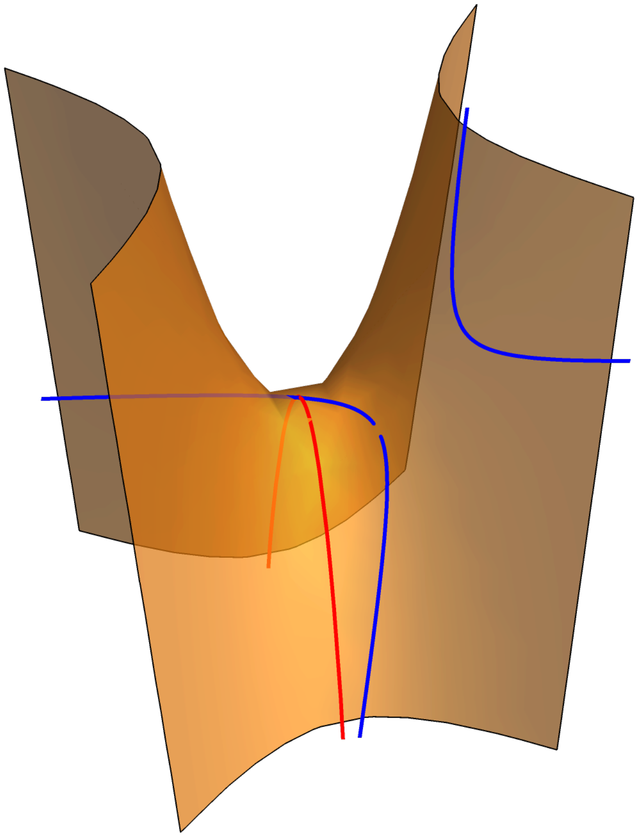
```
In[ ]:= sr3f = Simplify[f, sr3]
```

Out[ ]= $\left\{\frac{-5.03965 - 0.269488\ t + 0.585511\ t^2}{10.0677 - 6.34591\ t + 1.\ t^2}, \frac{5.03965 + 0.269488\ t - 0.585511\ t^2}{10.0677 - 6.34591\ t + 1.\ t^2}, \frac{-2.52274 - 1.85995\ t - 0.342823\ t^2}{10.0677 - 6.34591\ t + 1.\ t^2}\right\}$

So the symmetry sends the red parabola to the blue hyperbola below!

```
Show[ContourPlot3D [{ss == 0}, {x, -10, 10},
   {y, -10, 10}, {z, -10, 10}, ContourStyle → Opacity[.7], Mesh → None],
  ParametricPlot3D [{g, Sr3f}, {t, -30, 30}, PlotStyle → {Red, Blue}],
  Axes → False, Boxed → False, ImageSize → Full]
```

Out[ ]=



3. Cubic Surfaces

3.1 A rational Surface

Joe Harris, in his book [*Algebraic Geometry, A First Course],* claims on p. 157 that a complex cubic surface containing a rational normal curve, eg. a curve equivalent to the twisted cubic must be rational. Based on material I have so far developed I cannot give a proof in the real case. But here is an example of a rational cubic surface containing the twisted cubic curve. This surface is, unsurprisingly, singular.

We will see that this real parameterized surface does not fill up the implicit surface containing it. But I will calculate the singular set both in the surface and parameter space. In the parameter space this is an algebraic set, but in the parameterized surface only a semi-algebraic set.

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

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

The curve

```
In[ ]:= nrat = F /. {t → 0}
```

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

will be in this surface.

```
In[ ]:= f1 = pol2affNS[F, 3, 3, {s, t}, {x, y, z}][[1]]
```

» Number of equations 1

Out[ ]= $0. - 2.\,x^2 + 1.33333\,x^3 + 2.\,y - 5.33333\,x\,y - 2.\,x^2\,y - 3.33333\,y^2 + 0.666667\,y^3 + 4.\,z + 5.33333\,x\,z - 0.666667\,z^2$

```
In[ ]:= f = roundPolyMD[3 f1, {x, y, z}, 1]
```

Out[ ]= $-6\,x^2 + 4\,x^3 + 6\,y - 16\,x\,y - 6\,x^2\,y - 10\,y^2 + 2\,y^3 + 12\,z + 16\,x\,z - 2\,z^2$

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

Out[ ]= 0

So we see that f = 0 is a rational cubic surface containing a normal rational curve. Looking for non-regular points

```
In[ ]:= grd = Grad[f, {x, y, z}]
```

Out[ ]= $\{-12\,x + 12\,x^2 - 16\,y - 12\,x\,y + 16\,z,\ 6 - 16\,x - 6\,x^2 - 20\,y + 6\,y^2,\ 12 + 16\,x - 4\,z\}$

```
In[ ∘ ]:= sol = Solve[f == 0 && grd == 0, {x, y, z}]
```

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

Out[∘]= $\left\{\{y \to 3 + x,\ z \to 3 + 4\,x\},\ \left\{x \to -\frac{4}{3},\ y \to \frac{5}{3},\ z \to -\frac{7}{3}\right\}\right\}$

The first solution to this is a singular line in f = 0, which is easily seen to be parametric .

```
In[ ∘ ]:= pln = {t, 3 + t, 3 + 4 t}
Out[ ∘ ]= {t, 3 + t, 3 + 4 t}
```

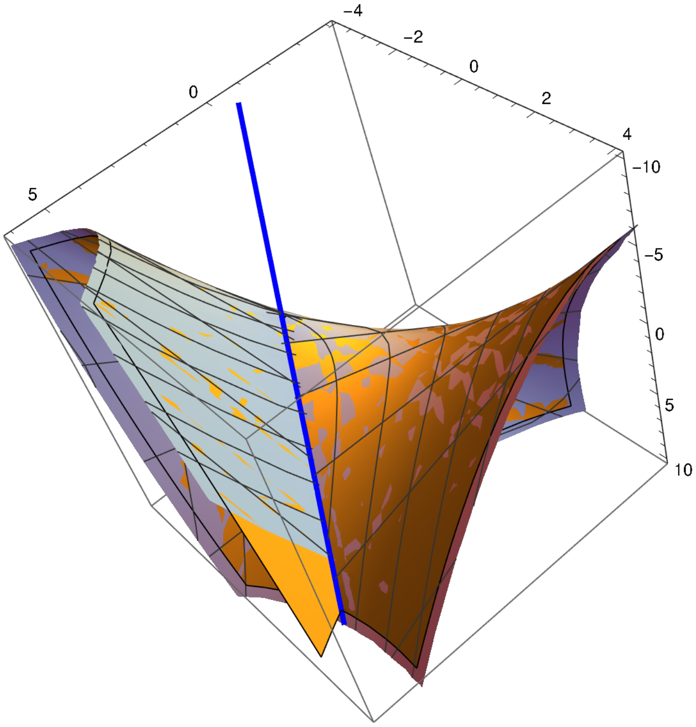
Check

```
In[ ∘ ]:= Expand[f /. Thread[{x, y, z} → pln]]
Out[ ∘ ]= 0
```

Plotting we see there is a bit of a problem here .

```
In[ ∘ ]:= Show[ContourPlot3D [f == 0, {x, -4, 4}, {y, -4, 6}, {z, -10, 10}, Mesh → None,
            MaxRecursion → 5], ParametricPlot3D [F, {s, -5, 5}, {t, -5, 5}, PlotStyle → LightGray],
          ParametricPlot3D [pln, {t, -4, 4}, PlotStyle → Blue]]
```

Out[∘]=



The singular line is in f = 0, but not in the image of F. So here is another case where the implicit surface contains the parametric surface but is larger. We might think of removing this line as we did with blowups in the Space Curve Book, 3.3 but that will not work here because it is not a component.

Note the point {-3, 0, -9} is on this line, hence in surface f = 0. But consider the nearby points

```
In[ ∘ ]:= NSolveValues[{f, y, x + 3.001}, {x, y, z}]
Out[ ∘ ]= {{-3.001, 0., -9.004 - 0.00223652 ⅈ}, {-3.001, 0., -9.004 + 0.00223652 ⅈ}}
```

So this point is close to complex points of f = 0 so this line is not isolated, merely a real set in a complex surface. As mentioned in Section 1.1.2 the non-regular set of this implicit surface is an algebraic set.

But we cannot consider points such as {-3, 0, -9} as singular points of the parametric real surface because they are not on this surface. Fortunately `sol` above gave us another solution, the point

```
In[ ∘ ]:= p = {-4/3, 5/3, -7/3}
            4   5    7
Out[ ∘ ]= {-—, —, -—}
            3   3    3
```

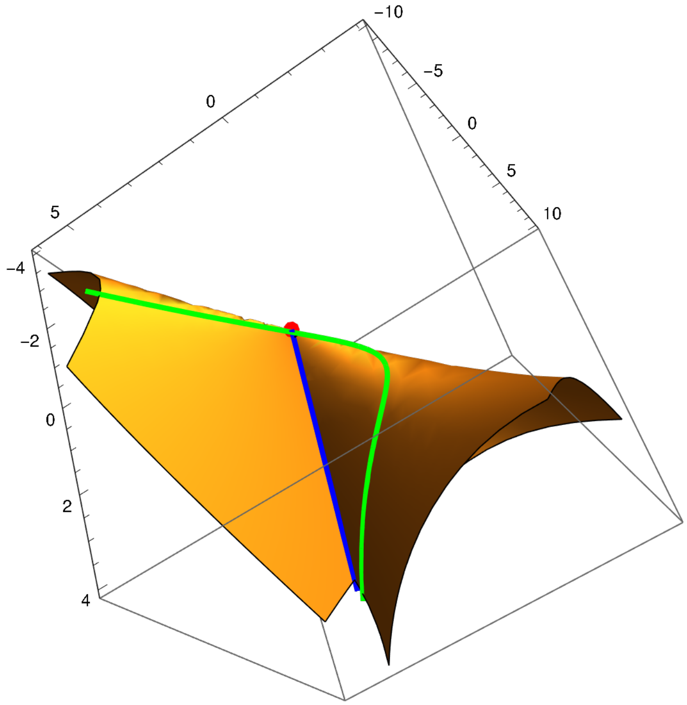
We check that this is on the parametric surface F and also on the parametric line pln

```
In[ ∘ ]:= Solve[F == p, {s, t}]
                            1
Out[ ∘ ]= {{s → -1, t → -—}}
                            3

In[ ∘ ]:= F /. {s → -1, t → -1/3}
            4   5    7
Out[ ∘ ]= {-—, —, -—}
            3   3    3

In[ ∘ ]:= pln /. {t → -4/3}
            4   5    7
Out[ ∘ ]= {-—, —, -—}
            3   3    3
```

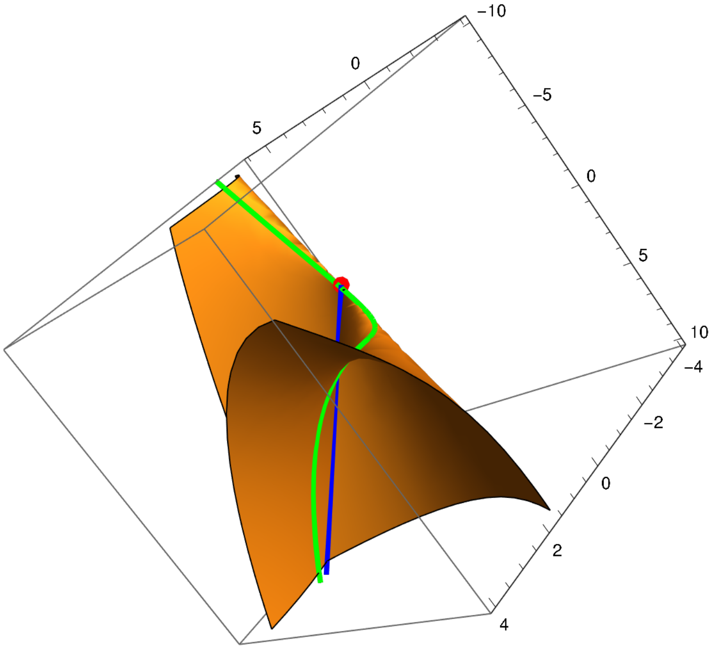
Plotting

In[ ]:=
```
Show[ContourPlot3D[f == 0, {x, -4, 4}, {y, -4, 6}, {z, -10, 10}, Mesh → None,
    MaxRecursion → 5], ParametricPlot3D[pln, {t, -4/3, 4}, PlotStyle → Blue],
  Graphics3D[{Red, PointSize[.025], Point[{-4/3, 5/3, -7/3}]}],
  ParametricPlot3D[{s, s^2, s^3}, {s, -3, 3}, PlotStyle → Green]]
```

Out[ ]=



we see that the singular set for the parametric surface F is the subset of pln with $t \geq \frac{-4}{3}$. In this plot we also show the twisted cubic which goes near, but not through p. Another view is



Thus Abhyankar' s statement quoted in 1.1.2 is not true for parametric surfaces, the regular set is only a semi-algebraic set.

Back in the parametric space we note if we take a point on the line pln with t > -1 we get two real solutions to

```
In[ ∘ ]:= Solve[(pln /. {t → 2}) == F, {s, t}]
Out[ ∘ ]= {{s → -1 - √10 , t → 3 + √10 }, {s → -1 + √10 , t → 3 - √10 }}
```

But if t < -1 then we get two imaginary solutions.

```
In[ ∘ ]:= Solve[(pln /. {t → -2}) == F, {s, t}]
Out[ ∘ ]= {{s → -1 - ⅈ √2 , t → -1 + ⅈ √2 }, {s → -1 + ⅈ √2 , t → -1 - ⅈ √2 }}
```

The fact that exact solutions seem to come with a square root suggest that perhaps the inverse image of the singular part of the parametric surface F is a plane quadric. So using the method above using NSolveValue we obtain the following 5 points in the inverse image

```
In[ ∘ ]:= plnInverseSet = {{-4.162277660168382` , 6.162277660168382` },
                1
            {-1, - ─}, {1, -1}, {1.6457513110645907` , -0.6457513110645907` },
                3
            {2.162277660168379` , -0.16227766016837908` }}
                                          1
Out[ ∘ ]= {{-4.16228 , 6.16228}, {-1, - ─}, {1, -1}, {1.64575 , -0.645751}, {2.16228 , -0.162278}}
                                          3
```

Checking we see all these points map to the singular line

```
In[ ∘ ]:= F /. Thread[{s, t} → #] & /@ plnInverseSet
                                   4   5    7
Out[ ∘ ]= {{2. , 5. , 11.}, {- ─ , ─ , - ─}, {0, 3, 3}, {1. , 4. , 7.}, {2. , 5. , 11.}}
                                   3   3    3
```

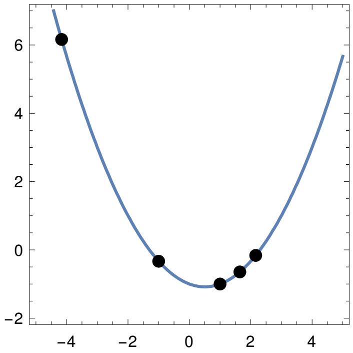
Applying the function `aCurve` from my *Plane Curves Book,* it is in the `GlobalFunctionsS.nb` notebook,

```
In[ ∘ ]:= parb = Chop[aCurve2D[plnInverseSet , x, y]]
Out[ ∘ ]= 6.03157 + 2.01052 x - 2.01052 x^2 + 6.03157 y
```

we get a parabola .

```
In[ ]:= Show[ContourPlot[parb == 0, {x, -5, 5}, {y, -2, 7}, ImageSize → Small],
         Graphics[{Black, PointSize[.04], Point[plnInverseSet]}]]
```

Out[ ]=



Check that a random point on this parabola gives a point on pln with t > -1

```
In[ ]:= x1 = RandomReal[{-4, 4}]
        q2 = NSolveValues[{parb, x - x1}, {x, y}]⟦1⟧

Out[ ]= -3.6543

Out[ ]= {-3.6543, 4.6694}

In[ ]:= q3 = F /. Thread[{s, t} → q2]

Out[ ]= {1.0151, 4.0151, 7.06041}

In[ ]:= pln /. {t → q3⟦1⟧}

Out[ ]= {1.0151, 4.0151, 7.06041}
```

Thus this parabola in the parameter space folds on itself to give the singular set of the parametric surface F.

3.2. 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 $x^3 + y^3 + z^3 + 1 = 0$ 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 α,β be the two cube roots of -1 other than -1 itself, that is α=.5-Sqrt[3]/2 ⅈ, β = .5+Sqrt[3]/2ⅈ.

```
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 α, and or β we can easily construct the remaining 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}
```

3.2.1 The double Six configuration

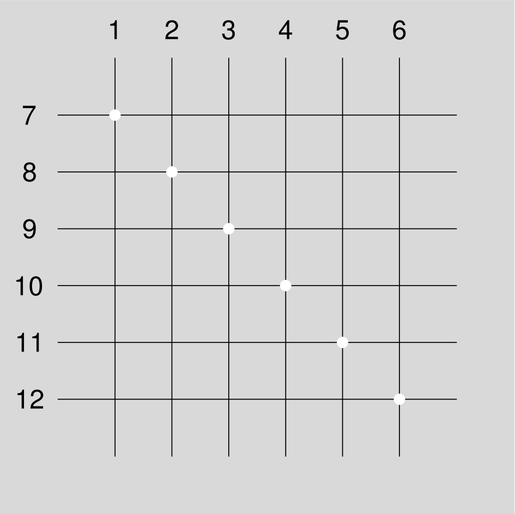
In H . S . M . Coxeter' s review of Volume II of Ludwig Schläfli’s collected works he says that one paper

> . . 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 configuration 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.

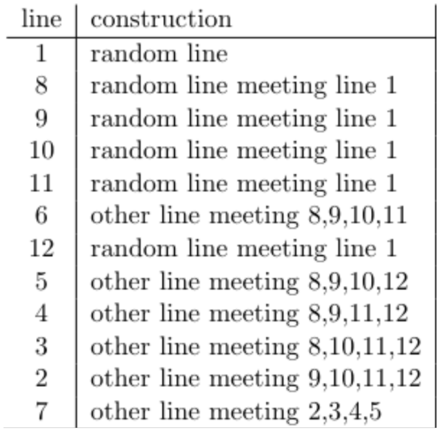
3.3 The theory

[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 configurations 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.

3.3.1 The Problems that must be solved

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 × (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[ ]=
$$\{x \to 1.88744 \times 10^{-23} \times \left(-7.4689 \times 10^{21} + 5.59143 \times 10^{22}\, y - 6332.47 \sqrt{1.39113 \times 10^{36} + 6.65696 \times 10^{36}\, y + 7.96388 \times 10^{36}\, y^2}\right),$$
$$z \to 1.36396 \times 10^{-21} \times \left(-9.33613 \times 10^{20} - 3.6658 \times 10^{20}\, y + 791.559 \sqrt{1.39113 \times 10^{36} + 6.65696 \times 10^{36}\, y + 7.96388 \times 10^{36}\, y^2}\right)\}$$

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[ ]= $\{0., 2.77556 \times 10^{-17}\}$

Out[ ]= $5.64271 \times 10^{-13} + 2.04947 \times 10^{-12}\ t + 1.75637 \times 10^{-12}\ t^{2}$

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[ ]:= $p = \left\{-\frac{14}{15}, -\frac{17}{15}, 0\right\};$

$q = \left\{\frac{1}{13}, -\frac{11}{13}, \frac{11}{13}\right\};$

```
In[ ]:= l = ratLine3D[p, q]
```

Out[ ]= $\left\{-\frac{171}{56} + x - \frac{197\, y}{56}, \frac{14}{15} + x - \frac{197\, z}{165}\right\}$

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[ ]= $\left\{-\frac{162}{455}, -\frac{63}{65}, \frac{44}{91}\right\}$

```
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}
```

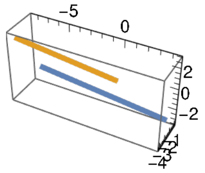
Note that the function returns a list of length 4 with the last component 0, this means infinite point. Now let

```
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 Chapter 2 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

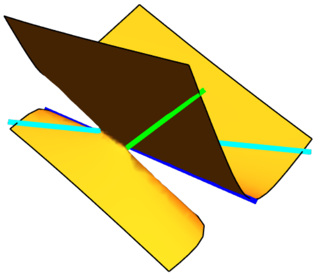
```
In[ ]:= rl1eq = pl2eqMD[rline1, t, {x, y, z}]

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[ ]=
```

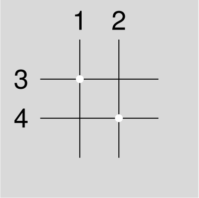


E. Finding two lines intersecting 4 skew lines the last intersecting the hyperboloid generated by the first 3 in two points. Actually Hilbert stated this more generally, but if a line not in, or tangent to, a hyperboloid intersects a hyperboloid in one point then since the equation of the hyperboloid 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.

G . Material from the Space Curve Book . We have already seen this in Section 1.4 The Torus Story but just as a reminder these numerical linear algebra techniques will also be needed here. Thankfully these are all given by functions in my **GlobalFunctionsNS.nb** so if one is willing to accept these functions as given there is no need to review this information. Specifically the functions needed are **`vectorSpaceIntersection3, sylvesterMD, hBasisMD.`**

3.4 Example of 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. In order to give the construction note that lines L1, L8, L9, L10, L11, and L12 can be given arbitrarily as long as L8, L9, L10, L11 and L12 met L1 and are mutually skew.

Recall the hyperboloid and its equation are given by

In[ ]:=

$$\text{hyp1} = \left\{\frac{t - s^2\,t}{1 - s^2}, \frac{1 + s^2 - 2\,s\,t}{1 - s^2}, \frac{2\,s - t - s^2\,t}{1 - s^2}\right\};$$

```
hypEq = 1 - x^2 - y^2 + z^2;
```

Some rational lines can be calculated directly from the parameterization hyp1. Given a value s0 ≠ 1

In[ ]:=

```
r1[s0_] := Expand[hyp1 /. {s → s0}]
r2[s0_] := Expand[{r1[s0]⟦1⟧, r1[s0]⟦2⟧, -r1[s0]⟦3⟧}]
```

```
In[ ]:= L1 = r2[-1/2]
        L8 = r1[-2/3]
        L9 = r1[-1/4]
        L10 = r1[1/4]
```

Out[ ]= $\left\{t, \frac{5}{3} + \frac{4\,t}{3}, \frac{4}{3} + \frac{5\,t}{3}\right\}$

Out[ ]= $\left\{t, \frac{13}{5} + \frac{12\,t}{5}, -\frac{12}{5} - \frac{13\,t}{5}\right\}$

Out[ ]= $\left\{t, \frac{17}{15} + \frac{8\,t}{15}, -\frac{8}{15} - \frac{17\,t}{15}\right\}$

Out[ ]= $\left\{t, \frac{17}{15} - \frac{8\,t}{15}, \frac{8}{15} - \frac{17\,t}{15}\right\}$

One can check using plineIntersectionMD that these meet the criteria. The following points are also on L1

```
In[ ]:= l1a = L1 /. {t → -5/4}
        l1b = L1 /. {t → 1/2}
```

Out[ ]= $\left\{-\frac{5}{4}, 0, -\frac{3}{4}\right\}$

Out[ ]= $\left\{\frac{1}{2}, \frac{7}{3}, \frac{13}{6}\right\}$

Line L11 will be chosen arbitrarily, but not actually randomly

$$\mathbf{L11} = \left\{t, \frac{5}{13} + \frac{4\,t}{13}, \frac{4}{13} + \frac{11\,t}{13}\right\};$$

It can be checked that l1a is on L11 these lines are skew to each other and the lines L8, L9, L10.

Now we need to find L6 which meets L8, L9, L10 and L11. Now L8, L9 and L10 were chosen inside the hyperbola hyp1 so we don't need to do problem D here. But then the line we need is the line in the opposite ruling to L8, L9, L10 through the second point of intersection of hyp1 with L11. To do this it helps to find the implicit equation of L11, using problem B above. L11 goes through **l1a** above, a second point is

```
In[ ]:= l11a = L11 /. {t → 1}
```

Out[ ]= $\left\{1, \frac{9}{13}, \frac{15}{13}\right\}$

```
In[ ]:= L11eq = ratLine3D[l1a, l11a]
```

Out[ ]= $\left\{\frac{5}{4} + x - \frac{13\,y}{4}, \frac{4}{11} + x - \frac{13\,z}{11}\right\}$

We then solve, using just Solve to get rational solutions

```
In[ ]:= SolveValues[hypEq == 0 && L11eq == 0, {x, y, z}]
```

Out[ ]= $$\left\{\left\{-\frac{5}{4}, 0, -\frac{3}{4}\right\}, \{2, 1, 2\}\right\}$$

But the first solution is just `l1a` so the desired point is

```
In[ ]:= l11b = {2, 1, 2}

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

So using problem A

```
In[ ]:= tp11 = tangentPlaneNS[hypEq, l11b, {x, y, z}]

Out[ ]= 2 - 4 x - 2 y + 4 z
```

```
In[ ]:= Solve[hypEq == 0 && tp11 == 0, {x, y, z}]

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

Out[ ]= $$\left\{\{y \to 1, z \to x\}, \left\{y \to \frac{1}{3} \times (-5 + 4 x), z \to \frac{1}{3} \times (-4 + 5 x)\right\}\right\}$$

Our first solution gives the parametric line $\{t, 1, t\}$ in the hyperboloid which will be a candidate for L6

```
In[ ]:= hypEq /. Thread[{x, y, z} → {t, 1, t}]

Out[ ]= 0
```

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

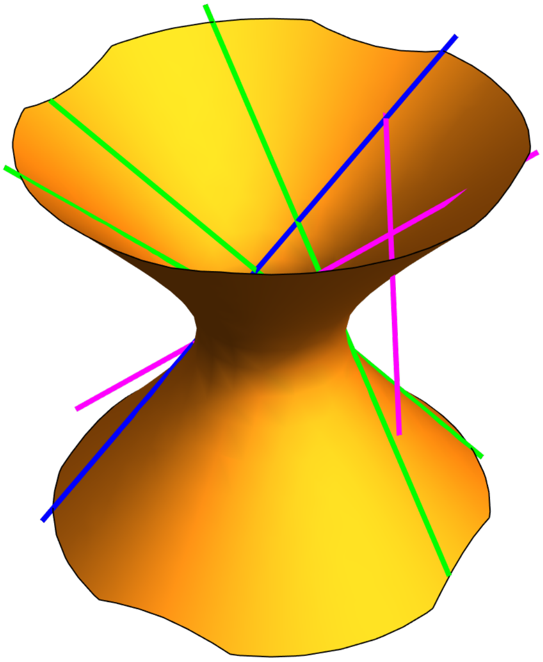
```
In[ ]:= pLineIntersectionMD[L6, L8, t, {x, y, z}, dTol]
        pLineIntersectionMD[L6, L9, t, {x, y, z}, dTol]
        pLineIntersectionMD[L6, L10, t, {x, y, z}, dTol]
        pLineIntersectionMD[L6, L11, t, {x, y, z}, dTol]

Out[ ]= {-0.666667, 1., -0.666667}

Out[ ]= {-0.25, 1., -0.25}

Out[ ]= {0.25, 1., 0.25}

Out[ ]= {2., 1., 2.}
```

Finally we choose L12, this must meet L1 but be skew to L8, L9 ,L10, L11 and L6. We leave the check to the reader.

$$\text{L12} = \left\{t, \frac{599}{180} - \frac{179\,t}{90}, \frac{409}{180} - \frac{19\,t}{90}\right\};$$

```
In[ ]:= Show[ContourPlot3D [hypEq == 0, {x, -3, 3}, {y, -3, 3}, {z, -3, 3}, Mesh → None],
         ParametricPlot3D [{L1, L8, L9, L10, L11, L12}, {t, -3, 3},
          PlotStyle → {Blue, Green, Green, Green, Magenta, Magenta}],
         Axes → None, Boxed → False, ImageSize → Medium]
```

Out[ ]=



One case not obvious by the picture is whether L11 meets L12, but it doesn’t.

```
In[ ]:= pLineIntersectionMD [L11, L12, t, {x, y, z}, dTol]
Out[ ]= {}
```

This works . A similar method to the one finding L6 will give the other lines in the double 6 although in the other cases Problem D will be needed to find a hyperboloid containing 3 of the lines. Here are the 12 lines.

In[ ]:=

$$\text{L1} = \left\{t, \frac{5}{3} + \frac{4\,t}{3}, \frac{4}{3} + \frac{5\,t}{3}\right\};$$

```
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[∘]:=

$$L5 = \left\{t, \frac{29}{20} - \frac{21\,t}{20}, -\frac{21}{20} + \frac{29\,t}{20}\right\};$$

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[∘]:=

$$L8 = \left\{t, \frac{13}{5} + \frac{12\,t}{5}, -\frac{12}{5} - \frac{13\,t}{5}\right\};$$

In[∘]:=

$$L9 = \left\{t, \frac{17}{15} + \frac{8\,t}{15}, -\frac{8}{15} - \frac{17\,t}{15}\right\};$$

In[∘]:=

$$L10 = \left\{t, \frac{17}{15} - \frac{8\,t}{15}, \frac{8}{15} - \frac{17\,t}{15}\right\};$$

In[∘]:=

$$L11 = \left\{t, \frac{5}{13} + \frac{4\,t}{13}, \frac{4}{13} + \frac{11\,t}{13}\right\};$$

In[∘]:=

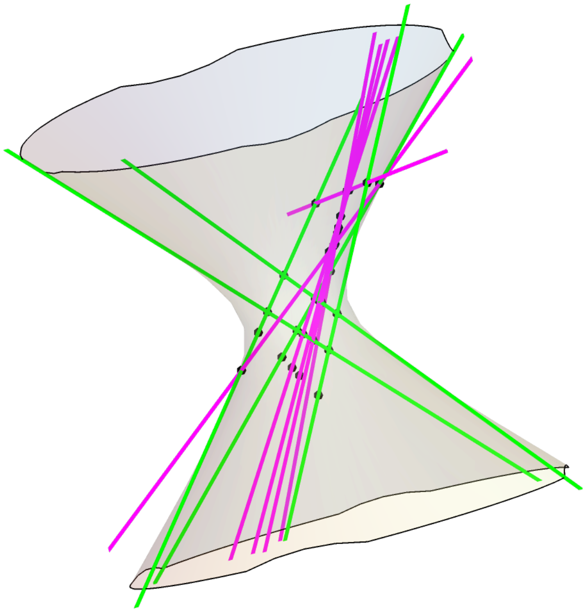
$$L12 = \left\{t, \frac{599}{180} - \frac{179\,t}{90}, \frac{409}{180} - \frac{19\,t}{90}\right\};$$

The reader with Mathematica can use pLineIntersectionMD to check that lines that should intersect should and those that shouldn’t don’t. For example

In[∘]:=
```
pLineIntersectionMD [L1, L7, t, {x, y, z}, dTol]
```

Out[∘]=
```
{}
```

Here is the plot of the full double 6 with the intersection points



3.5 the 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.

As before these lines have the following implicit equations

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

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[ ]=
$$\left\{\frac{13}{12}+x-\frac{5\,y}{12},\ \frac{12}{13}+x+\frac{5\,z}{13}\right\}$$

In[ ]:=
```
L9eq = ratLine3D[L9 /. {t → 0}, L9 /. {t → 4}]
```

Out[ ]=

$$\left\{\frac{17}{8}+x-\frac{15\,y}{8},\ \frac{8}{17}+x+\frac{15\,z}{17}\right\}$$

In[ ]:=
```
L10eq = ratLine3D[L10 /. {t → 0}, L10 /. {t → 4}]
```

Out[ ]=

$$\left\{-\frac{17}{8}+x+\frac{15\,y}{8},\ -\frac{8}{17}+x+\frac{15\,z}{17}\right\}$$

In[ ]:=
```
L11eq = ratLine3D[L11 /. {t → 0}, L11 /. {t → 4}]
```

Out[ ]=

$$\left\{\frac{5}{4}+x-\frac{13\,y}{4},\ \frac{4}{11}+x-\frac{13\,z}{11}\right\}$$

In[ ]:=
```
L12eq = ratLine3D[L12 /. {t → 0}, L12 /. {t → 4}]
```

Out[ ]=

$$\left\{-\frac{599}{358}+x+\frac{90\,y}{179},\ -\frac{409}{38}+x+\frac{90\,z}{19}\right\}$$

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}
```

3.6 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 mutually 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.

There is one disclaimer. As long as all the lines are chosen randomly there should be no problem, but if the lines are arbitrarily chosen then one must check that L12 is also skew to the constructed L6 which depends on the first 5 lines.

Here is our construction, new to this edition of the book.

The first step is to find the implicit equations for the 6 lines. We presumably did this in the previous section using the method used above in constructing L6. We call these L1eq, L2eq ...

We do the following 4 calculations

```
In[ ]:= X4 = mExpsMD[4, {x, y, z}];
        sylL1 = sylvesterMD[L1eq, 4, {x, y, z}];
        sylL8 = sylvesterMD[L8eq, 4, {x, y, z}];
        int18 = vectorSpaceIntersection [sylL1, sylL8, dTol].X4;
        Basis18 = hBasisMD[int18, 4, {x, y, z}, dTol]
        tDegMD[#, {x, y, z}] & /@ Basis18

     » Initial Hilbert Function  {1, 2, 2, 2, 2}

     » Final Hilbert Function  {1, 2, 2, 2, 2}

Out[ ]= {-8. - 7. x + 4. y + 1. z, 4.33333 + 7.46667 x + 3.2 x^2 - 4.26667 y - 3.73333 x y + 1. y^2}

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

```
In[ ]:= sylL11 = sylvesterMD[L11eq, 4, {x, y, z}];
        sylL9 = sylvesterMD[L9eq, 4, {x, y, z}];
        int911 = vectorSpaceIntersection [sylL11, sylL9, dTol].X4;
        Basis911 = hBasisMD[int911, 4, {x, y, z}, dTol]
        tDegMD[#, {x, y, z}] & /@ Basis911

     » Initial Hilbert Function  {1, 3, 2, 2, 2}

     » Final Hilbert Function  {1, 3, 2, 2, 2}

Out[ ]= {-2.45455 - 7.63636 x - 3.54545 x^2 + 3.72727 y + 8.77273 x y + 3.31818 z + 1. x z,
          0.435897 + 0.553846 x + 0.164103 x^2 - 1.51795 y - 0.841026 x y + 1. y^2,
          -0.960373 - 2.94965 x - 1.43963 x^2 + 1.68019 y + 3.83263 x y + 0.636364 z + 1. y z,
          0.540793 + 1.39301 x + 0.0592075 x^2 - 1.0704 y - 2.51935 x y - 0.727273 z + 1. z^2}

Out[ ]= {2, 2, 2, 2}
```

```
In[ ]:= sylL10 = sylvesterMD[L10eq, 4, {x, y, z}];
        sylL12 = sylvesterMD[L12eq, 4, {x, y, z}];
        int1012 = vectorSpaceIntersection [sylL10, sylL12, dTol].X4;
        Basis1012 = hBasisMD[int1012, 4, {x, y, z}, dTol]
        tDegMD[#, {x, y, z}] & /@ Basis1012

     » Initial Hilbert Function  {1, 3, 2, 2, 2}

     » Final Hilbert Function  {1, 3, 2, 2, 2}

Out[ ]= {-0.549873 - 2.3229 x + 1.47125 x^2 + 1.19466 y + 0.633588 x y - 1.50763 z + 1. x z,
          3.77148 - 4.02889 x + 1.06074 x^2 - 4.46111 y + 2.52222 x y + 1. y^2,
          2.86845 - 0.212231 x - 0.672072 x^2 - 2.90937 y - 0.126802 x y - 0.329262 z + 1. y z,
          1.95113 + 0.435233 x - 1.73875 x^2 - 1.60615 y - 0.851824 x y - 0.778626 z + 1. z^2}

Out[ ]= {2, 2, 2, 2}
```

```
In[ ]:= sylB18 = sylvesterMD[Basis18, 4, {x, y, z}];
        sylB911 = sylvesterMD[Basis911, 4, {x, y, z}];
        sylB1012 = sylvesterMD[Basis1012, 4, {x, y, z}];
```

```
In[ ]:= intAll = vectorSpaceIntersection3[sylB18, sylB911, sylB1012, 10^(-11)].X4;
        BasisAll = hBasisMD[intAll, 4, {x, y, z}, dTol];
        tDegMD[#, {x, y, z}] & /@ BasisAll

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

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

Out[ ]= {3, 4, 4, 4, 4, 4, 4}
```

We notice that there is one cubic and 6 4th degree equations. This cubic must be the unique cubic through the six lines. Further, since each of the other 21 lines intersects three of the six lines then they must also lie in this cubic. The skeptical reader who has calculated all 21 of these lines can easily check these last assertions directly. Here is a graphic showing the 6 lines and the cubic

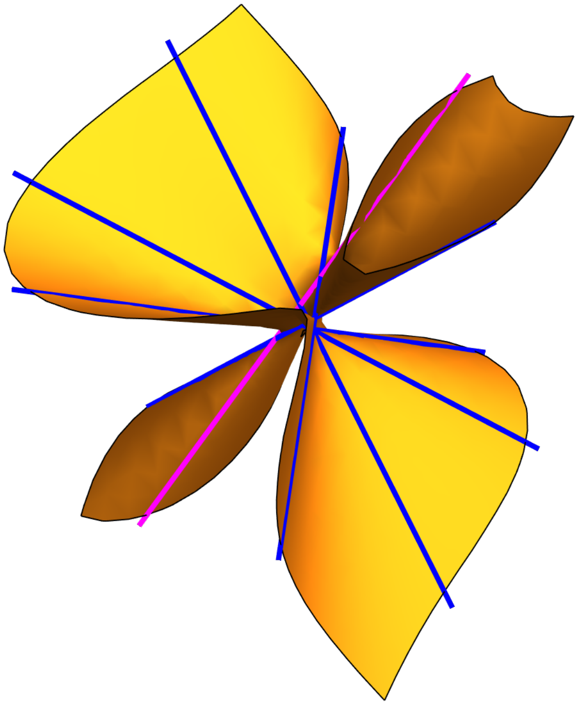
```
In[ ]:= SScubic = BasisAll[[1]]
```

Out[ ]=

$$-1.9593 - 3.01427\, x + 0.746586\, x^2 + 2.14804\, x^3 + 5.29948\, y + 2.26374\, x\, y - 1.01454\, x^2\, y - 4.24981\, y^2 + 0.695088\, x\, y^2 + 0.909641\, y^3 + 2.21734\, z + 0.461988\, x\, z - 1.25871\, x^2\, z - 1.82023\, y\, z - 0.480467\, x\, y\, z - 0.341667\, y^2\, z + 0.121951\, z^2 - 1.88933\, x\, z^2 + 0.164481\, y\, z^2 + 1.\, z^3$$

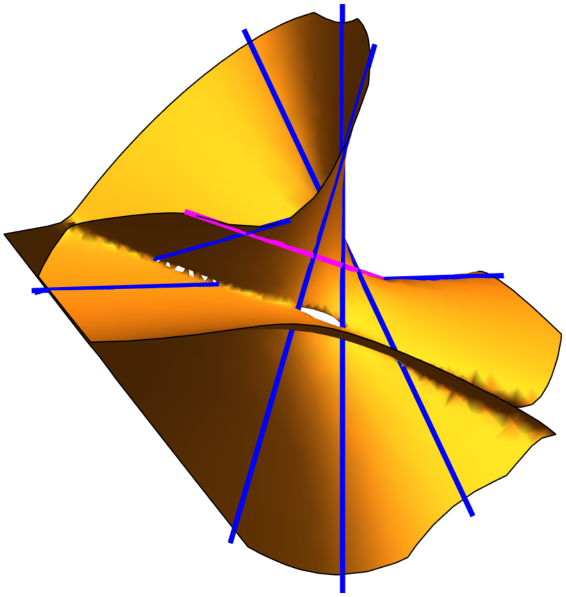
```
In[ ]:= Show[ContourPlot3D[SScubic == 0, {x, -10, 10}, {y, -10, 10}, {z, -10, 10}, Mesh → None],
          ParametricPlot3D[{L1, L8, L9, L10, L11, L12}, {t, -10, 10},
           PlotStyle → {Magenta, Blue, Blue, Blue, Blue, Blue}], Axes → None, Boxed → False]
```

Out[ ]=



You should recognize this as the cover illustration of the Surface Story.

Here is a closeup view of the middle of the graphic showing first the magenta line intersecting all 5 blue lines and the blue lines not intercepting each other. There are actually 2 small holes in the surface not visible in the large view.



3.7 Finding lines on a given smooth cubic, Example 1

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 illustrate with 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[ ]=
$$16\,a1^3 - 72\,a2 + 24\,a1^2\,a2 - 93.5307\,a2^2 - 31\,a2^3 +$$
$$\left(-48\,a1^2 + 48\,a1^2\,b1 + 48\,a1\,a2\,b1 - 72\,b2 + 24\,a1^2\,b2 - 187.061\,a2\,b2 - 93\,a2^2\,b2\right) t +$$
$$\left(-48\,a1 + 24\,a2 - 96\,a1\,b1 + 48\,a1\,b1^2 + 24\,a2\,b1^2 + 48\,a1\,b1\,b2 - 93.5307\,b2^2 - 93\,a2\,b2^2\right) t^2 +$$
$$\left(16 - 48\,b1 - 48\,b1^2 + 16\,b1^3 + 24\,b2 + 24\,b1^2\,b2 - 31\,b2^3\right) t^3$$

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 in human time. 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 complicated 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.

We re-arrange the lines to find a double 6, we do not show the work since it is tedious. Remember that this is one example of a double 6 in this cubic, but not the only one.

In[ ]:=

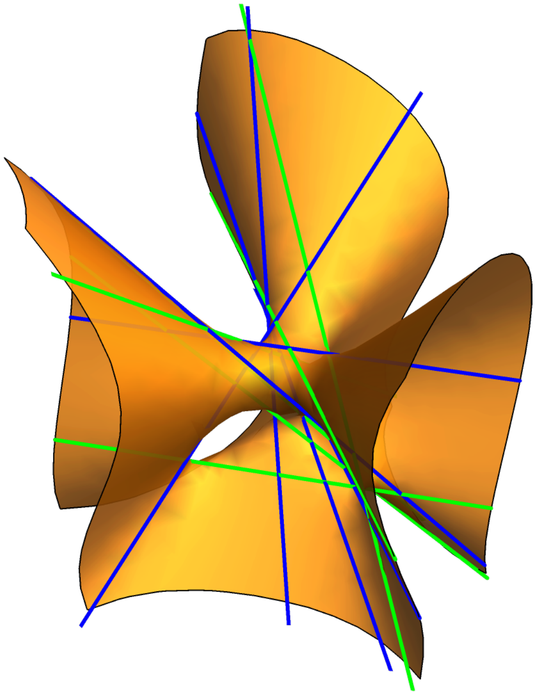
0	5	3	4	7	25	26	2	16	14	9	20	10
5	1	0	0	0	0	0	0	3	3	3	4	3
3	0	1	0	0	0	0	3	0	3	3	3	3
4	0	0	1	0	0	0	3	3	0	3	3	3
7	0	0	0	1	0	0	3	3	3	0	3	3
25	0	0	0	0	1	0	3	3	3	3	0	3
26	0	0	0	0	0	1	3	4	3	3	3	0
2	0	3	3	3	3	3	1	0	0	0	0	0
16	3	0	3	3	3	4	0	1	0	0	0	0
14	3	3	0	3	3	3	0	0	1	0	0	0
9	3	3	3	0	3	3	0	0	0	1	0	0
20	4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

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

```
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**.

```
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 &]
```

```
{{{0. , 0. , 0.}, 3}}
```

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

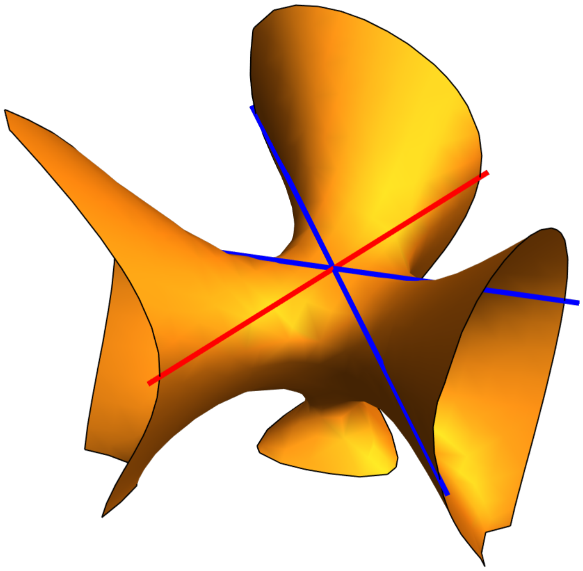
```
KeySelect [otherAssoc , otherAssoc [#] == {0, 0, 0} &]
```

```
<|{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[ ]=



3.8 Finding lines, Clebsch Diagonal Cubic

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.

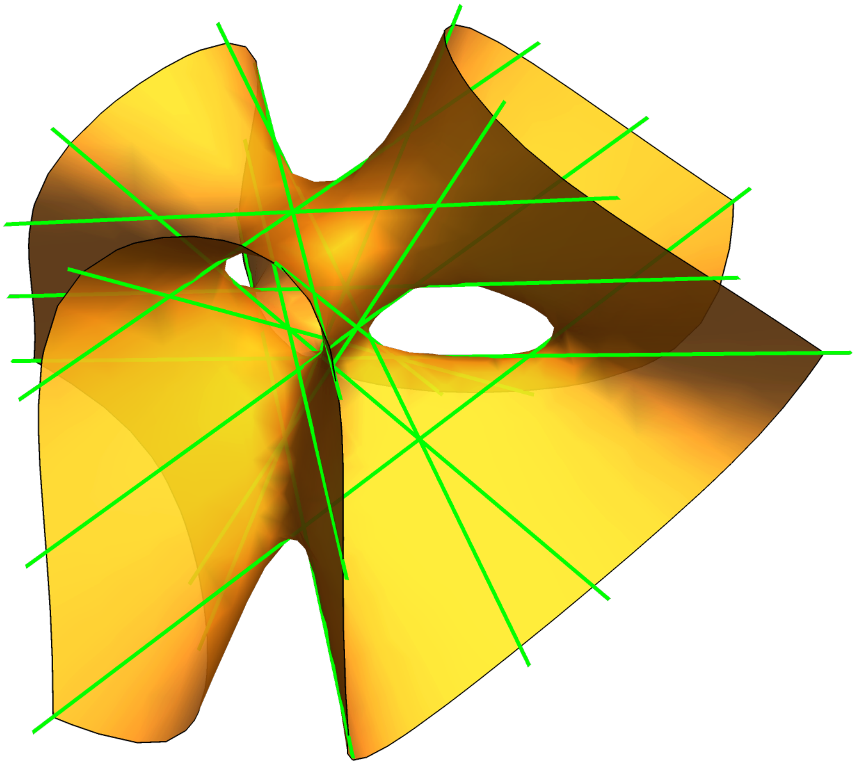
```
In[ ]:= epoints = {{1/6, 1/6, 0}, {0, -1/3, 0}, {0, 0, -1/3},
           {1/6, 1/6, 0}, {1/3, 1/3, 1/3}, {-1/3, 0, 0}, {0, 1/6, 1/6, 0}};
```

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[ ]=



4. Fourth Degree and Related Surfaces

We already saw in Chapter 1 some surfaces related to the torus. Here we will consider these again as well as some other 4 degree surfaces. But while there were large continuous groups of symmetries in degree 2 the symmetry group of higher degree surfaces will generally be finite. So we start our discussion with the geometric point groups. As a comment one source is the book *Geometry and Symmetry* by Paul B. Yale (Holden Day, 1968). He also was one of my undergraduate professors, he made me get excited about abstract algebra while he was writing this book. Unfortunately it is written in 1960’s algebra speak with hardly any matrix representations so this book is not very relevant to this discussion.

4.1 Geometric Point groups and applications

We remind our readers that in mathematics a *matrix group* is a set of $n \times n$ matrices for some fixed n which satisfy the two rules: 1) the product of any two matrices in the group is in the group and 2) the inverse of any matrix in the group is in the group. In particular the set of projective linear symmetries of a surfaces forms a group. We caution that multiplication of matrices is not commutative, possibly $A.B \neq B.A$, so these groups do not satisfy a commutative law. To an algebraist this makes them more interesting. They do, because of matrix multiplication in general, satisfy the associative law $(A.B).C = A.(B.C)$ however. The identity n×n matrix with ones down the main diagonal and 0 elsewhere is automatically in every matrix group from rules 1) and 2) above.

As an example, a well known type of matrix groups are the *crystallographic point groups.* There is a brief discussion in *Wolfram Math World* but I recommend instead the article by this name in Wikipedia. We will loosely follow the standard notation in these sources but must worry about compatibility with Mathematica variables.

4.1.1 Tetrahedral groups

The so called *tetrahedral groups*, generally denoted T, are perhaps better called cubic groups as in [Yale]. These will give a building base for the other groups. We give an inductive construction, the reader should notice the pattern. Tet2 and Tet3 are given as 2×2, 3×3 matrices to facilitate the inductive construction but must be expanded to 4×4 matrices using 2 or 1 applications of **m2TM** before using **fltMD** or **FLTNS.**

In[ ]:=

```
Tet2 = {{{1, 0}, {0, 1}}, {{0, 1}, {1, 0}}};
MatrixForm[#] & /@ Tet2
```

Out[ ]= $\left\{\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}, \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\right\}$

```
In[ ]:= Tet3 = Join[{{1, 0, 0}, Prepend[#⟦1⟧, 0], Prepend[#⟦2⟧, 0]} & /@ Tet2,
           {Prepend[#⟦1⟧, 0], {1, 0, 0}, Prepend[#⟦2⟧, 0]} & /@ Tet2,
           {Prepend[#⟦1⟧, 0], Prepend[#⟦2⟧, 0], {1, 0, 0}} & /@ Tet2];
        MatrixForm[#] & /@
         Tet3
```

Out[ ]=

$$\left\{\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix},\begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix},\begin{pmatrix}0&1&0\\1&0&0\\0&0&1\end{pmatrix},\begin{pmatrix}0&0&1\\1&0&0\\0&1&0\end{pmatrix},\begin{pmatrix}0&1&0\\0&0&1\\1&0&0\end{pmatrix},\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}\right\}$$

```
In[ ]:= Tet4 = Join[{{1, 0, 0, 0}, Prepend[#⟦1⟧, 0], Prepend[#⟦2⟧, 0], Prepend[#⟦3⟧, 0]} & /@ Tet3,
           {Prepend[#⟦1⟧, 0], {1, 0, 0, 0}, Prepend[#⟦2⟧, 0], Prepend[#⟦3⟧, 0]} & /@ Tet3,
           {Prepend[#⟦1⟧, 0], Prepend[#⟦2⟧, 0], {1, 0, 0, 0}, Prepend[#⟦3⟧, 0]} & /@ Tet3,
           {Prepend[#⟦1⟧, 0], Prepend[#⟦2⟧, 0], Prepend[#⟦3⟧, 0], {1, 0, 0, 0}} & /@ Tet3];
        MatrixForm[#] & /@
         Tet4
```

Out[ ]=

$$\left\{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&1&0&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{pmatrix},\right.$$

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

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

$$\left.\begin{pmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&0&0&1\\0&0&1&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&1&0&0\\0&0&0&1\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&0&0&1\\0&1&0&0\\0&0&1&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&0&1\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&0&0&1\\0&0&1&0\\0&1&0&0\\1&0&0&0\end{pmatrix}\right\}$$

These have exactly one 1 in each row and column. Note there are 2 symmetry matrices in **Tet2**, 6 in **Tet3** and 24 in **Tet4.**

As examples, Tet3 provides symmetries of several cubic surfaces. The first example is the Fermat surface

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

Out[ ]= $1 + x^3 + y^3 + z^3$

Note that

```
In[ ]:= FLTNS[fermat, m2TM[Tet3[[RandomInteger[{1, 6}]]]], {x, y, z}]
```

Out[ ]= $1 + x^3 + y^3 + z^3$

We saw that this surface had 3 real lines given parametrically

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

Permuting, say **lf1**, with the 6 symmetries in **Tet3**

```
In[ ]:= Column[Table[fltMD[lf1, m2TM[Tet3[[n]]]], {n, 6}]]

        {t, -t, -1}
        {t, -1, -t}
        {-t, t, -1}
Out[ ]= {-1, t, -t}
        {-t, -1, t}
        {-1, -t, t}
```

sends this line to one of the three . You do need to notice that , say for $\{-t, t, -1\}$, changing the sign on both t's merely changes the direction of the parameterization, so $\{-t, t, -1\}$ is the same line as $\{t, -t, -1\}$. The example also works if **lf1** is replaced by **lf2** or **lf3.**

Another cubic example is the Clebsch Diagonal Cubic. The equation below makes it clear that permuting variables makes no difference.

```
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;
```

so for example

```
In[ ]:= cdc2 = FLTNS[cdc, m2TM[Tet3[[3]]], {x, y, z}]
```

Out[ ]=
$$1 - 9 x - 9 x^2 + 81 x^3 - 9 y + 126 x y - 189 x^2 y - 9 y^2 - 189 x y^2 + 81 y^3 - 9 z + 126 x z - 189 x^2 z + 126 y z + 54 x y z - 189 y^2 z - 9 z^2 - 189 x z^2 - 189 y z^2 + 81 z^3$$

```
In[ ]:= Expand[cdc2 - cdc]

Out[ ]= 0
```

Here the real lines must also be permuted, but this is more complicated as there are 27 of these . In these cubic cases we do not claim that these are the only affine symmetries, but your author does not know of any others .

4.1.2 Octahedral Groups

These also are considered cubic groups as we will see in our example . Essentially we now let the entries in our matrices take values in $\{1, -1\}$ instead of just 1. Although the tetrahedral groups were given as lists the rest of our groups will be given as functions so we don't need to list them all.

The following function gives all 3×3 diagonal matrices with elements in {1,-1}. We are representing integers in a reverse binary form. In order for this to work correctly this function needs the domain to be the set of integers 8 through 15.

```
In[ ∘ ]:=  unitDiag3[m_] := Module[{j}, If[m < 8 || m > 15, Echo["Need  8  ≤  m  ≤  15"];
             Abort[],
             j = Reverse[IntegerDigits[m, 2]];
             {{(-1)^j⟦1⟧, 0, 0}, {0, (-1)^j⟦2⟧, 0}, {0, 0, (-1)^j⟦3⟧}}]]
```

Then the group I will call Oh, or Oh(3), note regular capital O here, is the set of matrices

```
In[ ∘ ]:=  Oh[k_, m_] := m2TM[unitDiag3[m + 7].Tet3⟦k⟧]
```

Here, to get each symmetry once, k=1…6 and m =1…8 . Thus we will get 48 symmetries. Although they would generally be presented as 3×3 matrices we will give them already as 4×4 transformation matrices.

For example

```
In[ ∘ ]:=  Oh[3, 6] // MatrixForm
Out[ ∘ ]//MatrixForm=
```

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

This group of symmetries may be familiar as a discrete subgroup of symmetries of the sphere, that is the well known orthogonal group 𝕆(3).

```
In[ ∘ ]:=  sphere = x^2 + y^2 + z^2 - 1
```

Out[∘]= $-1 + x^2 + y^2 + z^2$

```
In[ ∘ ]:=  FLTNS[sphere, Oh[3, 6], {x, y, z}]
```

Out[∘]= $-1 + x^2 + y^2 + z^2$

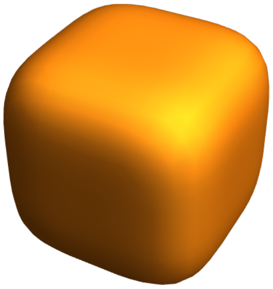
Of course the orthogonal group is a continuous group and infinite. However when working with the rounded cube one needs to use a point group.

```
In[ ∘ ]:=  rcube = x^4 + y^4 + z^4 - 1
```

Out[∘]= $-1 + x^4 + y^4 + z^4$

```
In[ ]:= ContourPlot3D[x^4 + y^4 + z^4 == 1, {x, -1, 1}, {y, -1, 1},
         {z, -1, 1}, Mesh → None, ImageSize → Small, Axes → False, Boxed → False]
```

Out[ ]=



```
In[ ]:= FLTNS[rcube, Oh[4, 7], {x, y, z}]
```

Out[ ]= $-1+x^4+y^4+z^4$

```
In[ ]:= M = m2TM[Orthogonalize[RandomReal[{-2, 2}, {3, 3}]]]

Out[ ]= {{-0.802743, -0.570305, 0.174226, 0}, {-0.596262, 0.763416, -0.248328, 0},
         {-0.00861577, 0.303228, 0.952879, 0}, {0, 0, 0, 1}}

In[ ]:= FLTNS[rcube, M, {x, y, z}]
```

Out[ ]=

$$-1.+0.521955\ x^4+0.662071\ x^3\ y+2.52318\ x^2\ y^2-0.344949\ x\ y^3+0.469864\ y^4-0.186999\ x^3\ z+0.857031\ x^2\ y\ z-1.05707\ x\ y^2\ z+0.48859\ y^3\ z+0.34509\ x^2\ z^2-0.951367\ x\ y\ z^2+0.657636\ y^2\ z^2+0.539357\ x\ z^3-0.774267\ y\ z^3+0.832879\ z^4$$

which is not a symmetry . So it appears that one can only have finitely many symmetries.

The **rcube** reminds one of some dice which have rounded edges to roll more smoothly. The reader is reminded that these transformations do not all apply to physical objects as they contain reflections which turn your right hand into your left hand which does not happen in the physical world. In some cases one may wish to work with a smaller group, the symmetries of determinant one.

```
In[ ]:= Clear[k, m]

In[ ]:= SO = Reap[Do[If[ Det[Oh[k, m]] == 1, Sow[{k, m}]], {k, 6}, {m, 8}]][[2, 1]]

Out[ ]= {{1, 1}, {1, 4}, {1, 6}, {1, 7}, {2, 2}, {2, 3}, {2, 5}, {2, 8}, {3, 2}, {3, 3}, {3, 5}, {3, 8},
         {4, 1}, {4, 4}, {4, 6}, {4, 7}, {5, 1}, {5, 4}, {5, 6}, {5, 7}, {6, 2}, {6, 3}, {6, 5}, {6, 8}}
```

There are 24 physical symmetries . For example

```
In[ ]:= MatrixForm[Oh[2, 8]]
```

Out[ ]//MatrixForm=

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

Note that as in 4.2 any symmetry of the **rcube** transfers to a symmetry of any surface projectively equivalent to the rcube. An example motivated by my breakfast today is

```
In[ ]:= K = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, .5, 0}, {0, 0, 0, 1}}

Out[ ]= {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 0.5, 0}, {0, 0, 0, 1}}
```

```
In[ ]:= jellyDonut = FLTNS[rcube, K, {x, y, z}]

Out[ ]= -1. + 1. x^4 + 1. y^4 + 16. z^4
```

```
In[ ]:= jrot = K.Oh[2, 3].Inverse[K]

Out[ ]= {{1., 0., 0., 0.}, {0., 0., -2., 0.}, {0., 0.5, 0., 0.}, {0., 0., 0., 1.}}
```

To see this as a rotation of the jelly Donut

```
In[ ]:= FLTNS[jellyDonut, jrot, {x, y, z}]

Out[ ]= -1. + 1. x^4 + 1. y^4 + 16. z^4
```

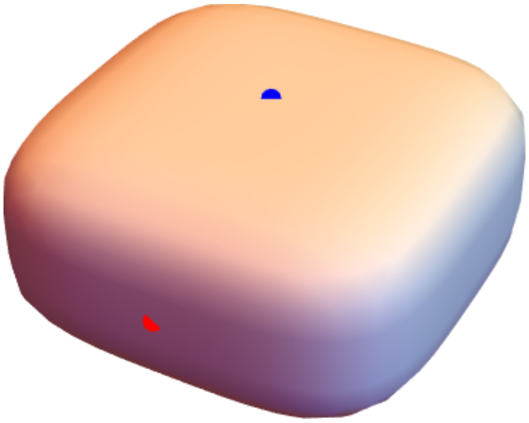
Note that point

```
In[ ]:= pjd = {0, 0, .5};
        jellyDonut /. Thread[{x, y, z} → pjd]

Out[ ]= 0.
```

is on my jellyDonut . Rotating

```
In[ ]:= qjd = fltMD[pjd, jrot]

Out[ ]= {0., -1., 0.}
```

```
In[ ]:= ImageCrop[Show[ContourPlot3D[jellyDonut == 0, {x, -1, 1},
           {y, -1, 1}, {z, -1, 1}, Mesh → None, ContourStyle → LightPink],
          Graphics3D[{PointSize[.03], {Blue, Point[pjd]}, {Red, Point[qjd]}}],
          Axes → False, Boxed → False]]

Out[ ]=
```



Once again, this is a theoretical rotation, do not try this on your own jelly donut.

4.1.2 The quartic hyperboloid.

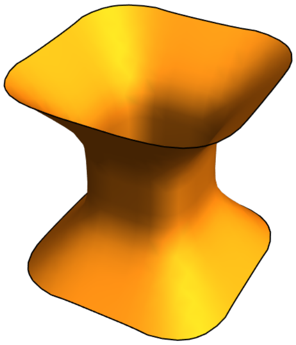
The quartic hyperboloid is the surface with equation

```
In[ ]:= qhyp = x^4 + y^4 - z^4 - 1
```

Out[ ]= $-1 + x^4 + y^4 - z^4$

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

Out[ ]=



We can think of this as an opened box, so affine symmetries are those symmetries of Oh that don't move the upper and lower faces of the **rcube**. In fact we can start with the symmetries of the quadric hyperboloid which are in Oh.

```
In[ ]:= ohyp =
         Reap[Do[If[FLTNS[x^2 + y^2 - z^2 - 1, Oh[k, m], {x, y, z}] == x^2 + y^2 - z^2 - 1, Sow[{k, m}]],
            {k, 6}, {m, 8}]]⟦2, 1⟧
```

```
Out[ ]= {{1, 1}, {1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6}, {1, 7},
          {1, 8}, {3, 1}, {3, 2}, {3, 3}, {3, 4}, {3, 5}, {3, 6}, {3, 7}, {3, 8}}
```

```
In[ ]:= MatrixForm[Oh @@ #] & /@ ohyp
```

Out[ ]=

$$\left\{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}-1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&-1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}-1&0&0&0\\0&-1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&-1&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}-1&0&0&0\\0&1&0&0\\0&0&-1&0\\0&0&0&1\end{pmatrix},\right.$$

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

$$\left.\begin{pmatrix}0&-1&0&0\\-1&0&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\0&0&-1&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&-1&0&0\\1&0&0&0\\0&0&-1&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&1&0&0\\-1&0&0&0\\0&0&-1&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&-1&0&0\\-1&0&0&0\\0&0&-1&0\\0&0&0&1\end{pmatrix}\right\}$$

Note that these all have the form of matrices in $\mathbb{HO}(4)$ of Chapter 2 of the left type, except with entries restricted to {0,-1,1}.

The interesting thing is these discrete symmetries also work for the quartic hyperboloid

```
In[ ]:= FLTNS[qhyp, Oh @@ ohyp⟦RandomInteger[{1, 16}]⟧, {x, y, z}]
```

Out[ ]= $-1 + x^4 + y^4 - z^4$

But a general symmetry in ℍ𝕆(4) of that type does not work

```
In[ ]:= A1 = ( 0.5814431788612586`   0.8135870142496832`             0.`                     0.`
               0.8135870142496833`  -0.5814431788612587`             0.`                     0.`
                       0.`                    0.`            0.47455284783002816`    0.8802270131144
                       0.`                    0.`             0.8802270131144637`   -0.4745528478300
```

```
Out[ ]= {{0.581443, 0.813587, 0., 0.}, {0.813587, -0.581443, 0., 0.},
         {0., 0., 0.474553, 0.880227}, {0., 0., 0.880227, -0.474553}}
```

```
In[ ]:= FLTNS[qhyp, A1, {x, y, z}]
```

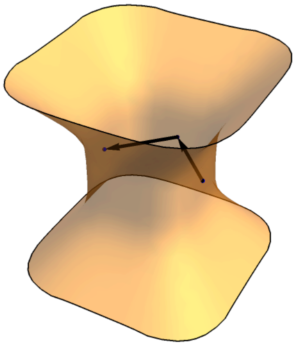
Out[ ]= $-0.65103 + 0.552439\, x^4 - 0.612791\, x^3\, y + 2.68537\, x^2\, y^2 + 0.612791\, x\, y^3 + 0.552439\, y^4 - 0.918302\, z - 2.09382\, z^2 + 0.918302\, z^3 - 0.65103\, z^4$

For qhyp we don’t have circles on the surface to work with be we can see the results of a symmetry by looking at arrows from the point to the image point.

For example start with point {1,0,0} on qhyp and apply twice

```
In[ ]:= A2 = Oh[3, 6]
Out[ ]= {{0, -1, 0, 0}, {1, 0, 0, 0}, {0, 0, -1, 0}, {0, 0, 0, 1}}

In[ ]:= fltMD[{1, 0, 0}, A2]
Out[ ]= {0, 1, 0}

In[ ]:= fltMD[{0, 1, 0}, A2]
Out[ ]= {-1, 0, 0}

In[ ]:= Show[ContourPlot3D[qhyp == 0, {x, -2, 2}, {y, -2, 2}, {z, -2, 2},
          Mesh → None, ContourStyle → Opacity[.55], Axes → False, Boxed → False],
         Graphics3D[{{Blue, Ball[{1, 0, 0}, .04], Ball[{0, 1, 0}, .04], Ball[{-1, 0, 0}, .04]},
           {Black, Thickness[.01], Arrow[{{1, 0, 0}, {0, 1, 0}}],
            Arrow[{{0, 1, 0}, {-1, 0, 0}}]}}], ImageSize → Small]
```

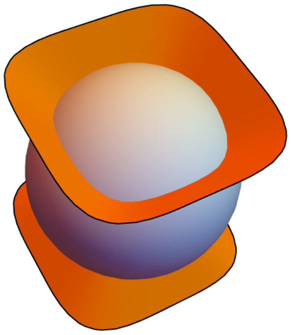
Out[ ]=



Unlike the **rcube** which is bounded the hyperboloids have infinite curves so there can be projective symmetries. The maximal form for the quartic hyperboloid is the cone $x^4 + y^4 - z^4$.

In[ ]:=
```
ContourPlot3D [{x ^ 4 + y ^ 4 - z ^ 4 == 0,  x ^ 2 + y ^ 2 + z ^ 2 == 1},
 {x, -1, 1}, {y, -1, 1}, {z, -1, 1}, ContourStyle → {Orange, LightGray},
 Mesh → None, Axes → False, Boxed → False, ImageSize → Small]
```

Out[ ]=



To induce a point symmetry of the quadric hyperboloid $x^2 + y^2 - z^2 - 1$ a 4×4 real matrix it appears that the matrix must be in ℍ𝕆(4) and Oh(4). There are two types, we called left and right of these (see Section 2.9 or paragraph 85 in **GlobalFunctions.nb).** We can construct these as follows :

In[ ]:=
```
unitDiag2 [m_] := Module[{j}, If[m < 2, Abort[], j = Reverse[IntegerDigits [m, 2]];
   {{(-1) ^ j⟦1⟧, 0}, {0, (-1) ^ j⟦2⟧}}]]
```

In[ ]:=
```
Hyp4[t1_, t2_, b1_, b2_, p_] := Switch[p, 1,
  Join[Partition [Flatten [Riffle[unitDiag2 [b1 + 3].Tet2⟦t1⟧, {{0, 0}, {0, 0}}]], 4],
   Partition [Flatten [Riffle[{{0, 0}, {0, 0}}, unitDiag2 [b2 + 3].Tet2⟦t2⟧]], 4]], 2,
  Join[Partition [Flatten [Riffle[{{0, 0}, {0, 0}}, unitDiag2 [b1 + 3].Tet2⟦t1⟧]], 4],
   Partition [Flatten [Riffle[unitDiag2 [b2 + 3].Tet2⟦t2⟧, {{0, 0}, {0, 0}}]], 4]]]
```

For this function t1,t2 come from 1,2, while b1,b2 go from 1 to 4, and p goes from 1,2. So we have 128 projective symmetries of the quartic hyperboloid. These all lie in ℍ𝕆(4) so are also symmetries of the quadric hyperboloid, but not vice versa. Note also if t2 = 1 and b2 = 1, 2 then this is in m2TM[Oh].

One technicality is that transformation matrices are homogeneous, that is a constant multiple of a transformation gives the same results. For example, consider the IdentityMatrix[4] and

In[ ]:=
```
Id = IdentityMatrix [4];
MI = -Id;
MI // MatrixForm
```

Out[ ]//MatrixForm=

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

For the point

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

In[ ]:= fltMD[p, Id]
        fltMD[p, MI]

Out[ ]= {-1, 0, 0}

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

give the same result . This is actually true for all points on the quartic Hyperboloid

For this reason the function **Hyp4** above gives duplicate results. So to count actual symmetries we can normalize by

```
In[ ]:= Hyp4N[t1_, t2_, b1_, b2_, p_] :=
          If[Total[Hyp4[t1, t2, b1, b2, p][[4]]] > 0, Hyp4[t1, t2, b1, b2, p], -Hyp4[t1, t2, b1, b2, p]]

In[ ]:= Now

Out[ ]= Sun 21 Aug 2022 10:05:11 GMT-4

In[ ]:= Length[DeleteDuplicates [
           Flatten[Table[Hyp4N[t1, t2, b1, b2, p], {t1, 2}, {t2, 2}, {b1, 4}, {b2, 4}, {p, 2}], 4]]]

Out[ ]= 64
```

So this group has only 64 distinct symmetries .

Here are some examples, we must be a bit careful as this second type sends the z-plane to infinity. We could use fltiMD but that doesn’t help with plotting.

```
In[ ]:= A1 = Hyp4[2, 2, 2, 4, 2];
        A1 // MatrixForm

Out[ ]//MatrixForm=
```

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

```
In[ ]:= FLTNS[x^4 + y^4 - z^4 - 1, A1, {x, y, z}]
```

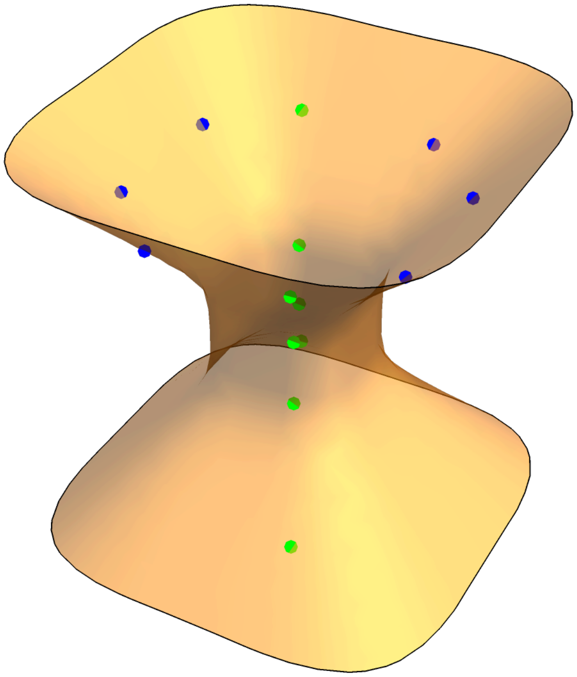
Out[ ]= $1 - x^4 - y^4 + z^4$

```
In[ ]:= P = {x, y, z} /. FindInstance [x^4 + y^4 - z^4 == 1 && z == 2, {x, y, z}, Integers , 8]

Out[ ]= {{-2, -1, 2}, {-2, 1, 2}, {-1, -2, 2}, {-1, 2, 2}, {1, -2, 2}, {1, 2, 2}, {2, -1, 2}, {2, 1, 2}}

In[ ]:= Q = fltMD[#, A1] & /@ P
```

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

```
In[ ]:= Show[ContourPlot3D [x ^ 4 + y ^ 4 - z ^ 4 == 1, {x, -3, 3}, {y, -3, 3}, {z, -3, 3}, Mesh → None,
          Axes → False, Boxed → False, ContourStyle → Opacity[0.55]], Graphics3D[
          {{Blue, PointSize[.02], Point[P]}, {Green, PointSize[.02], Point[Q]}}, ImageSize → Small]]
```

Out[ ]=



So this reminds us of the symmetries of the quadric hyperboloid in Chapter 2 where the images of points of z-height 2 lie on a vertical curve. I will make this more precise in the next sub-section.

4.1.3 An orbit in the quartic hyperboloid

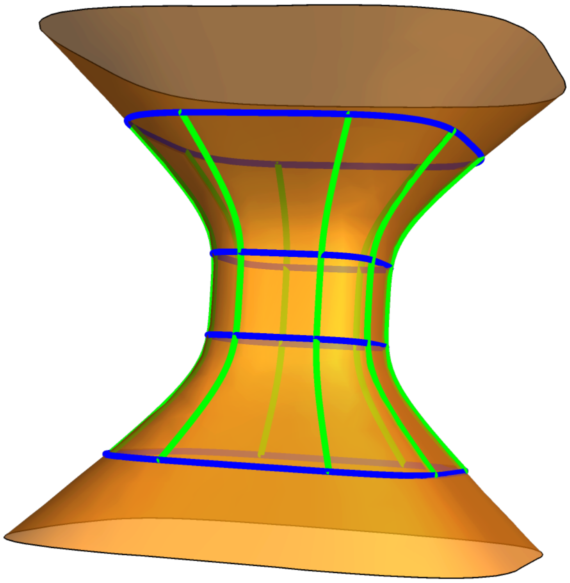
The *orbit* of a point *p* under a group of symmetries is the set of points obtained by applying all the symmetries in the group to this point. In Chapter 2 the set of projective symmetries of the quadric hyperboloid was shown, in two ways, to be transitive, that is, every point of the hyperboloid is in the orbit of any given point. This concept of orbit is more interesting when we look at finite symmetry groups such as our group **Hyp4**. We start with our point p1 = *P*⟦1⟧ = {−2, −1, 2}. We can calculate our 32 point orbit by

```
In[ ∘ ]:= p1 = {-2, -1, 2};
          orbitp1 = DeleteDuplicates [Flatten[Transpose[
               Table[fltMD[p1, Hyp4[t1, t2, b1, b2, p]], {t1, 2}, {t2, 2}, {b1, 4}, {b2, 4}, {p, 2}]], 4]]
```

```
Out[ ∘ ]= {{-2, -1, 2}, {-2, -1, -2}, {2, 1, -2}, {2, 1, 2}, {2, -1, 2}, {2, -1, -2}, {-2, 1, -2},
           {-2, 1, 2}, {-1, -2, 2}, {-1, -2, -2}, {1, 2, -2}, {1, 2, 2}, {1, -2, 2}, {1, -2, -2},
           {-1, 2, -2}, {-1, 2, 2}, {-1, -1/2, 1/2}, {-1, -1/2, -1/2}, {1, 1/2, -1/2}, {1, 1/2, 1/2},
           {1, -1/2, 1/2}, {1, -1/2, -1/2}, {-1, 1/2, -1/2}, {-1, 1/2, 1/2}, {-1/2, -1, 1/2}, {-1/2, -1, -1/2},
           {1/2, 1, -1/2}, {1/2, 1, 1/2}, {1/2, -1, 1/2}, {1/2, -1, -1/2}, {-1/2, 1, -1/2}, {-1/2, 1, 1/2}}
```

We can easily check that each of these points lies on at least one of the 8 planes $z-2$, $z-1/2$, $z+1/2$, $z+2$, $x-2y$, $x+2y$, $2x-y$, $2x+y$. Using **pathFinder3D** from Chapter 1 of my *Space Curves Book* one can trace the entire curve obtained by intersecting the quartic hyperbola by the first plane $z=2$. The intersections of the last 4 planes with the quartic hyperbola contain an infinite point and hence have two affine components. But it is easy to trace the part of the first curve cut out by $x=2y$ that lies between the points $\{-2,-1,2\}$ and $\{-2,-1,-2\}$. The rest of the curves or segments in the following plot can be calculated using **fltMD** and the symmetries in **Hyp4.**

In[∘]:=



Then it can be checked that orbitp1 consists of all the intersection points in the above plot. Unlike Chapter 2 we cannot construct more points higher or lower by taking powers of our symmetries because Hyp4 is a group and all powers already are contained there. The rounded square shape of the horizontal curves seems to be an obstruction to the existence of more rotational symmetries and the vertical curves are just projective linear images of the horizontal ones. So I am comfortable in claiming that Hyp4 contains all symmetries of the quartic hyperbola, but may be wrong as we saw with the strange symmetries of the quadric hyperbola.

4.1.4 More on the quartic hyperbola.

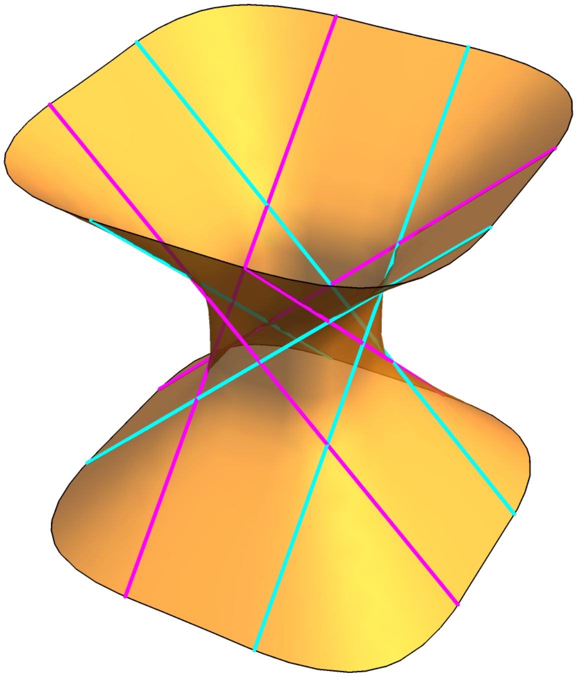
Unlike the quadric hyperbola which has 2 lines through every point, I know of only 8 lines. These are given by

In[ ]:=
```
hl1a = {1, t, t};
hl1b = {1, t, -t};
hl2a = {-1, t, -t};
hl2b = {-1, t, t};
hl3a = {t, 1, -t};
hl3b = {t, 1, t};
hl4a = {t, -1, t};
hl4b = {t, -1, -t};
```

These lines form two mutually skew sets of lines which intersect each line of the other set. This is somewhat analogous two the two rulings for the quadric hyperbola.

In[ ]:=
```
Show[ContourPlot3D[x^4 + y^4 - z^4 == 1, {x, -3, 3},
   {y, -3, 3}, {z, -3, 3}, Mesh → None, ContourStyle → Opacity[.75]],
 ParametricPlot3D[{hl1a, hl2a, hl3a, hl4a}, {t, -3, 3}, PlotStyle → Cyan],
 ParametricPlot3D[{hl1b, hl2b, hl3b, hl4b}, {t, -3, 3}, PlotStyle → Magenta],
 Axes → False, Boxed → False]
```

Out[ ]=

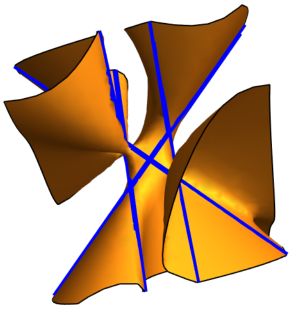


Note, however that some meet in infinite points.

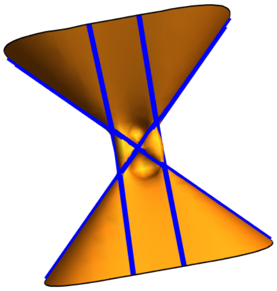
```
In[ ]:= pLineIntersectionMD [hl1a, hl2b, t, {x, y, z}, dTol]
Out[ ]= {0, -0.707107 , -0.707107 , 0}
```

Unfortunately, unlike smooth cubics, these lines are not enough to determine the quadric hyperboloid, here are three other quadric surfaces, there are actually many containing these 8 lines.

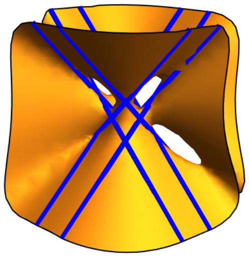
In[ ]:= {



,



,

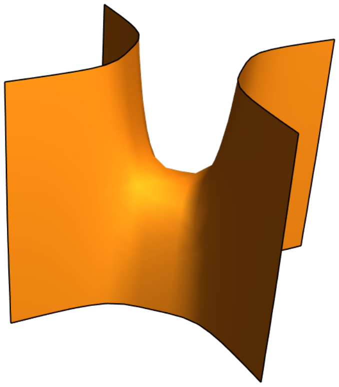


}

I also mention that the theory of quartic hyperboloids is much more complicated than the quartic, it appears there are many non-projectively equivalent projective quartic surfaces, such as the ones above. It also appears that the quartic saddle surface

```
In[ ]:= ContourPlot3D[z == x^4 - y^4, {x, -3, 3}, {y, -3, 3}, {z, -3, 3},
         Mesh → None, Axes → False, Boxed → False, ImageSize → Small]
```

Out[ ]=



may not be equivalent to the quartic hyperboloid. Here is a similar surface that is equivalent to **qhyp** along with a two of the lines. All 8, of course, can transform to **ss4** although some may go to the new infinite plane .

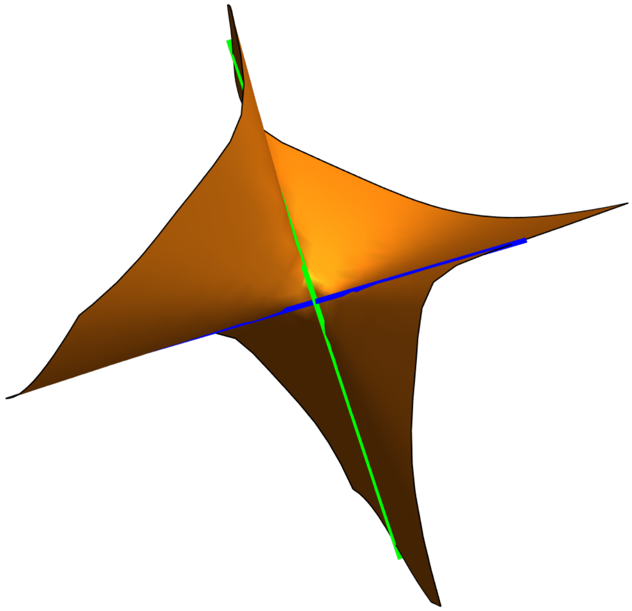
```
In[ ]:= J = iTransform3D[y - 1]

Out[ ]= {{0.788675, 0.57735, -0.211325, -0.366025}, {-0.57735, 0.57735, -0.57735, 1.},
         {-0.211325, 0.57735, 0.788675, -0.366025}, {0, 1.73205, 0, -1.73205}}
```

```
In[ ]:= ss4 = FLTNS[qhyp, J, {x, y, z}]

Out[ ]= -0.0624642 + 0.00564428 x + 0.908494 x^2 - 2.31319 x^3 + 1.95753 x^4 + 0.250926 y -
         1.07356 x y + 2.04904 x^2 y - 1.20753 x^3 y - 0.366025 y^2 + 0.707532 x y^2 +
         0.274519 x^2 y^2 + 0.288675 y^3 - 0.0245191 x y^3 + 0.496207 z - 0.732051 x z -
         0.158494 x^2 z + 1.18301 x^3 z - 0.390544 y z + 1.73205 x y z - 0.475481 x^2 y z +
         1.02452 y^2 z + 0.0245191 y^3 z - 1.64054 z^2 + 1.89054 x z^2 - 0.316987 y z^2 +
         0.475481 x y z^2 - 0.274519 y^2 z^2 + 2.89054 z^3 - 1.18301 x z^3 + 1.20753 y z^3 - 1.95753 z^4
```

```
In[ ]:= lss4a = fltMD[hl4a, J]
        lss4b = fltMD[hl4b, J]

Out[ ]= {-0.288675 × (-0.943376 + 0.57735 t),
         -0.288675 × (0.42265 - 1.1547 t), -0.288675 × (-0.943376 + 0.57735 t)}

Out[ ]= {-0.288675 × (-0.943376 + 1. t), -0.122008, -0.288675 × (-0.943376 - 1. t)}
```

In[ ]:=
```
Show[ContourPlot3D [ss4 == 0, {x, -3, 3}, {y, -3, 3}, {z, -3, 3}, Mesh → None],
  ParametricPlot3D [lss4a, {t, -10, 10}, PlotStyle → Blue],
  ParametricPlot3D [lss4b, {t, -10, 10}, PlotStyle → Green], Axes → False, Boxed → False]
```

Out[ ]=



Also, because of the equivalence J, all 128 symmetries of **qhyp** become symmetries of **ss4.** For example

In[ ]:=
```
Ω = J.Hyp4[2, 1, 3, 4, 2].Inverse[J]
```

Out[ ]=
```
{{0.721688 , 0.644338 , -0.0669873 , -0.221688},
 {-0.211325 , -0.788675 , -1.36603 , 0.211325},
 {-0.961325 , 0.32735 , -0.75 , 0.67265}, {0.549038 , 1.18301 , -1.18301 , 0.816987}}
```

In[ ]:=
```
Chop[FLTNS[ss4, Ω, {x, y, z}] + ss4, dTol]
```

Out[ ]=
```
0
```

so ss4 is equivalent to its image up to the constant -1.

In[ ]:=
```
lss4aJ = Chop[Simplify[fltMD[lss4a, Ω]], dTol]
lss4bJ = Chop[Simplify[fltMD[lss4b, Ω]], dTol]
```

Out[ ]=

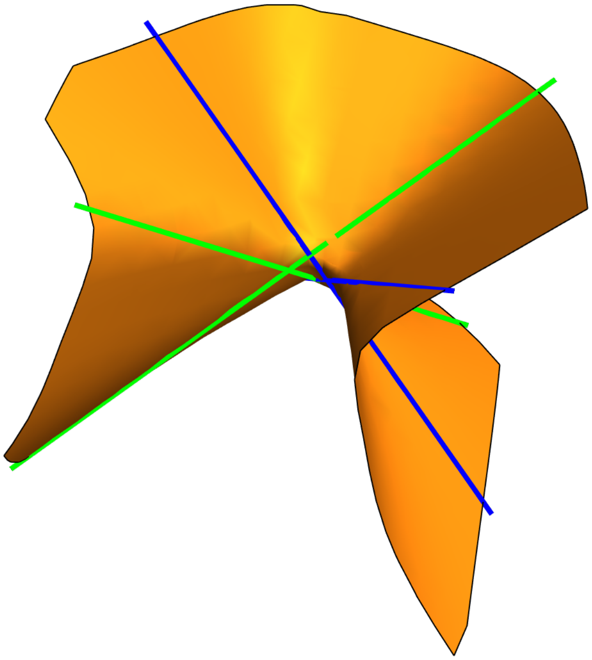
$$\left\{\frac{-0.244017+0.211325\,t}{1.+1.\,t},\ -\frac{0.244017}{1.+1.\,t},\ \frac{0.333333+0.788675\,t}{1.+1.\,t}\right\}$$

Out[ ]=

$$\left\{\frac{0.244017+0.455342\,t}{-1.+1.\,t},\ \frac{0.244017+0.666667\,t}{-1.+1.\,t},\ \frac{0.333333+0.122008\,t}{1.-1.\,t}\right\}$$

In[ ]:=
```
Show[ContourPlot3D [ss4 == 0, {x, -3, 3}, {y, -3, 3}, {z, -3, 3}, Mesh → None],
 ParametricPlot3D [{lss4a , lss4b}, {t, -10, 10}, PlotStyle → Blue],
 ParametricPlot3D [{lss4aJ , lss4bJ}, {t, -10, 10}, PlotStyle → Green],
 Axes → False , Boxed → False]
```

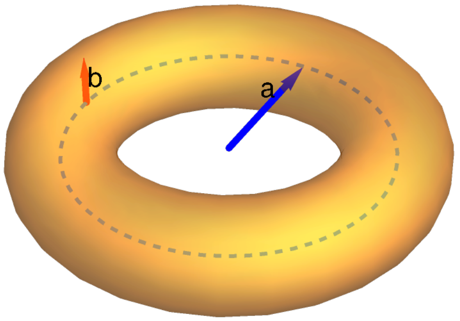
Out[ ]=



4.2 More on the Torus

As we have seen, working with quartics that there are many different types of seemingly similar surfaces which are not projectively equivalent. In particular tori come in different shapes, the ratio between the outer radius and inner radius must be the same in projectively equivalent tori. These are often given by parameters a,b as $a > b > 0$ shown below.

Out[ ]=



The trigonometric parameterization is

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

The parameters range from -π to π. From this we get the rational parametrization as in Chapter 1.4 Now the parameters range from $-\infty < u, v < \infty$.

In[ ]:=
$$\text{torabrat} = \text{Together}\left[\text{Expand}\left[\left\{\left(a+\frac{2\,b\,v}{1+v^2}\right)\left(\frac{2\,u}{1+u^2}\right),\ \left(a+\frac{2\,b\,v}{1+v^2}\right)\left(\frac{1-u^2}{1+u^2}\right),\ b\,\frac{1-v^2}{1+v^2}\right\}\right]\right]$$

Out[ ]=
$$\left\{\frac{2\,u\,(a+2\,b\,v+a\,v^2)}{(1+u^2)\times(1+v^2)},\ \frac{a-a\,u^2+2\,b\,v-2\,b\,u^2\,v+a\,v^2-a\,u^2\,v^2}{(1+u^2)\times(1+v^2)},\ \frac{b-b\,v^2}{1+v^2}\right\}$$

The implicit equation can be given by the equation

```
In[ ]:= torusEqab = Collect[a^4 + b^4 + x^4 + 2 x^2 y^2 + y^4 + 2 x^2 z^2 +
           2 y^2 z^2 + z^4 + b^2 (-2 a^2 - 2 x^2 - 2 y^2 - 2 z^2) + a^2 (-2 x^2 - 2 y^2 + 2 z^2), {a, b}]
```

Out[ ]=
$$a^4+b^4+x^4+2\,x^2\,y^2+y^4+2\,x^2\,z^2+2\,y^2\,z^2+z^4+b^2\,(-2\,x^2-2\,y^2-2\,z^2)+a^2\,(-2\,b^2-2\,x^2-2\,y^2+2\,z^2)$$

Since a, b are not linear factors we can not expect tori with different a,b to be projectively equivalent but if the ratio of a,b are the same then these will be equivalent

```
In[ ]:= torus31 = Expand[torusEqab /. {a → 3, b → 1}]
```

Out[ ]=
$$64-20\,x^2+x^4-20\,y^2+2\,x^2\,y^2+y^4+16\,z^2+2\,x^2\,z^2+2\,y^2\,z^2+z^4$$

```
In[ ]:= torus62 = Expand[torusEqab /. {a → 6, b → 2}]
```

Out[ ]=
$$1024-80\,x^2+x^4-80\,y^2+2\,x^2\,y^2+y^4+64\,z^2+2\,x^2\,z^2+2\,y^2\,z^2+z^4$$

Then

In[ ]:=
```
Expand[16 * FLTNS[torus31 , {{2, 0, 0, 0}, {0, 2, 0, 0}, {0, 0, 2, 0}, {0, 0, 0, 1}}, {x, y, z}]]
```

Out[ ]= $1024 - 80 x^2 + x^4 - 80 y^2 + 2 x^2 y^2 + y^4 + 64 z^2 + 2 x^2 z^2 + 2 y^2 z^2 + z^4$

so torus31 is projectively equivalent to torus62.

4.2.1 Symmetries of the torus

All these standard tori, using these equations, have the same symmetry group. Any rotation on the z-axis will preserve the torus. In addition we can reflect the torus in the xy-plane. Thus the obvious symmetries will have transformation matrices of the form

In[ ]:=
```
torusSym[ang_, refh_, refv_] := If[refh ^ 2 + refv ^ 2 == 2,
  Join[Join[RotationMatrix [ang].{{1, 0}, {0, refv}}, {{0, 0}, {0, 0}}, 2],
   {{0, 0, refv , 0}, {0, 0, 0, 1}}], Echo["Must have refh,refv =±1"];
  Abort[]]
```

Where **ang** is any real number, the angle, while **refh**=1 gives horizontal rotation and **refh** = -1 gives horizontal reflection, **refv** = -1 gives vertical reflextion.

For example

In[ ]:=
```
Clear[α];
{torusSym[α, 1, -1] // MatrixForm , torusSym[Pi / 3, -1, 1] // MatrixForm}
```

Out[ ]= $\left\{\begin{pmatrix} \mathrm{Cos}[\alpha] & \mathrm{Sin}[\alpha] & 0 & 0 \\ \mathrm{Sin}[\alpha] & -\mathrm{Cos}[\alpha] & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\right\}$

Another way to get the rotation symmetry is to take two planes through the z-axis, that is planes defined by linear equations involving only x, y, then `planeRotate3D` will give such a symmetry.

In[ ]:=
```
planeRotate3D [x - 3 y, x]
```

Out[ ]=
```
{{0.316228 , -0.948683 , 0. , 0.},
 {0.948683 , 0.316228 , 0. , 0.}, {0. , 0. , 1. , 0.}, {0. , 0. , 0. , 1.}}
```

also reflections can be obtained by

In[ ]:=
```
ReflectionMatrix [{3, 4, 0, 0}]
ReflectionMatrix [{0, 0, 1, 0}]
```

Out[ ]= $\left\{\left\{\frac{7}{25}, -\frac{24}{25}, 0, 0\right\}, \left\{-\frac{24}{25}, -\frac{7}{25}, 0, 0\right\}, \{0, 0, 1, 0\}, \{0, 0, 0, 1\}\right\}$

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

where in the first case one reflects in the plane $3 x + 4 y = 0$ and in the second gives reflection through the xy plane.

As in other situations any projective equivalent to these tori will have equivalent transformation groups. For example consider the affine equivalent surface from the transformation

```
In[ ]:= A = {{.7, .8, 0, -1}, {-.1, 2, .4, 2}, {0, 0, 2, 0}, {0, 0, 0, 1}};
        A // MatrixForm
```

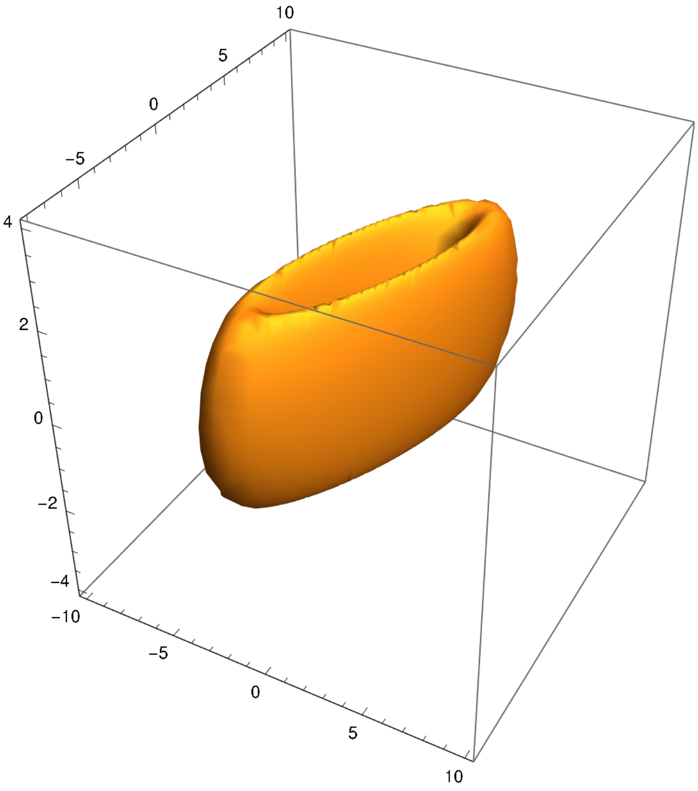
```
Out[ ]//MatrixForm=
 ( 0.7   0.8   0    -1 )
 (-0.1   2     0.4   2 )
 ( 0     0     2     0 )
 ( 0     0     0     1 )
```

```
In[ ]:= g = FLTNS[torus31, A, {x, y, z}]
```

```
Out[ ]= -25.0325 - 42.7573 x + 29.5471 x^2 + 23.6362 x^3 + 3.35152 x^4 + 22.9208 y - 35.4258 x y -
         30.7072 x^2 y - 5.11504 x^3 y + 8.55849 y^2 + 16.3294 x y^2 + 3.84051 x^2 y^2 - 3.57051 y^3 -
         1.44139 x y^3 + 0.26614 y^4 - 4.58416 z + 7.08517 x z + 6.14143 x^2 z + 1.02301 x^3 z -
         3.4234 y z - 6.53176 x y z - 1.5362 x^2 y z + 2.14231 y^2 z + 0.864837 x y^2 z -
         0.212912 y^3 z + 7.68648 z^2 + 3.8809 x z^2 + 1.06898 x^2 z^2 - 2.15874 y z^2 - 0.87147 x y z^2 +
         0.321817 y^2 z^2 + 0.37462 z^3 + 0.151232 x z^3 - 0.111694 y z^3 + 0.0732436 z^4
```

```
In[ ]:= ContourPlot3D[g == 0, {x, -10, 10}, {y, -8, 10}, {z, -4, 4}, Mesh → None, MaxRecursion → 4]
```

Out[ ]=



Consider the symmetry

```
In[ ]:= Ω = torusSym[Pi / 4, -1, 1]
        N[Ω] // MatrixForm
```

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

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0.707107 & -0.707107 & 0. & 0. \\ 0.707107 & 0.707107 & 0. & 0. \\ 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 1. \end{pmatrix}$$

Recall torus13 was the torus of Section 1.4. Using the parametric definition we can make a circle on this

```
In[ ]:= circ1 = torabrat /. {a → 3, b → 1, u → 0}
```

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

At v -> 0 we have the point on circ1

```
In[ ]:= p1 = circ1 /. {v → 0}
```

Out[ ]= $\{0, 3, 1\}$

```
In[ ]:= circ2 = fltMD[circ1, Ω]
```

Out[ ]= $\left\{-\frac{3 + 2 v + 3 v^2}{\sqrt{2} \left(1 + v^2\right)}, \frac{3 + 2 v + 3 v^2}{\sqrt{2} \left(1 + v^2\right)}, \frac{1 - v^2}{1 + v^2}\right\}$

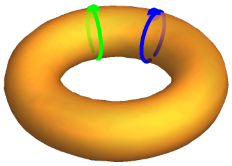
Now we get point

```
In[ ]:= p2 = fltMD[p1, Ω]
```

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

```
In[ ]:= Show[ContourPlot3D[torus31 == 0, {x, -4, 4}, {y, -4, 4}, {z, -4, 4}, Mesh → None,
          ContourStyle → Opacity[.7]], ParametricPlot3D[circ1, {v, -10, 10}, PlotStyle → Blue],
         ParametricPlot3D[circ2, {v, -10, 10}, PlotStyle → Green],
         Graphics3D[{{Blue, PointSize[.04], Point[p1]}, {Green, PointSize[.04], Point[p2]}}],
         Axes → False, Boxed → False, ImageSize → Small]
```

Out[ ]=



On g we have

```
In[ ]:= circA1 = fltMD[circ1, A]
```

Out[ ]=

$$\left\{-1. + \frac{0.8 \times (3 + 2\,v + 3\,v^2)}{1 + v^2},\ 2. + \frac{0.4 \times (1 - v^2)}{1 + v^2} + \frac{2. \times (3 + 2\,v + 3\,v^2)}{1 + v^2},\ \frac{2. \times (1 - v^2)}{1 + v^2}\right\}$$

```
In[ ]:= q1 = fltMD[p1, A]
Out[ ]= {1.4, 8.4, 2.}
```

The symmetry is then ψ given by

```
In[ ]:= ψ = A.Ω.Inverse[A]
Out[ ]= {{1.4381, -0.539886, 0.107977, 1.51787},
         {1.91588, -0.0238887, 0.204778, 3.96365}, {0., 0., 1., 0.}, {0., 0., 0., 1.}}
```

Checking that we have a symmetry

```
In[ ]:= Chop[FLTNS[g, ψ, {x, y, z}] - g, dTol]
Out[ ]= 0
```

```
In[ ]:= circA2 = Simplify[fltMD[circA1, ψ]]
```

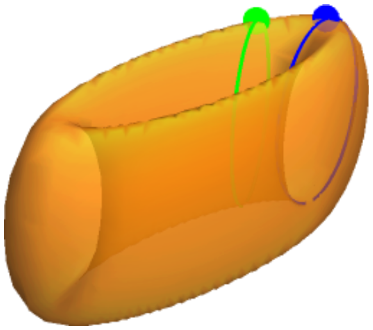
Out[ ]=

$$\left\{\frac{-0.787868 + 0.141421\,v - 0.787868\,v^2}{1. + v^2},\ \frac{6.85477 + 2.96985\,v + 6.05477\,v^2}{1. + v^2},\ \frac{2. - 2.\,v^2}{1 + v^2}\right\}$$

```
In[ ]:= q2 = fltMD[q1, ψ]
Out[ ]= {-0.787868, 6.85477, 2.}
```

```
In[ ]:= ImageCrop[Show[ContourPlot3D[g == 0, {x, -8, 8},
           {y, -8, 10}, {z, -4, 4}, Mesh → None, ContourStyle → Opacity[.7]],
         ParametricPlot3D[circA1, {v, -20, 20}, PlotStyle → Blue],
         ParametricPlot3D[circA2, {v, -20, 20}, PlotStyle → Green],
         Graphics3D[{{Blue, PointSize[.04], Point[q1]}, {Green, PointSize[.04], Point[q2]}}],
         Axes → False, Boxed → False, ImageSize → Medium]]
```

Out[ ]=



So in both cases we see we have something like a rotation and reflection even though this is not exactly correct.

4.2.2 Intersecting surface by plane

A parametric conic will always be taken to a parametric conic by a projective linear transformation. In particular the conic is always planar. In Chapter 2 the intersection of a quadric surface and plane is always a conic and this is why we construct many strange symmetries. But with quartic surfaces this is rare. A torus or equivalent surface does have at least 2 conics through any point, but perhaps no more.

The following function will help us decide if a plane through a surface can support a conic. This first tries to find 5 points in the intersection then lifts them to a plane, tests for general position, if that checks then the function attempts to produce a parametric conic . The surface may be given as a semi-algebraic set consisting as surface given as an equation and one or more inequalities, the plane is given as an equation. This is a probabilistic algorithm and may not work for any given run. You should try several times before giving up. To check that the parametric conic does lie in the surface write the surface in form $f = 0$ and use `Simplify[f/.Thread[{x,y,z}->conic]` where conic is the parametric output of the function .

In[ ]:=
```
planeIntersectSurfaceNS [plane_ , surfaceEq_ , V_] :=
 Module[{k, sol, tbl, a, u, v, par, points, rnc, s, t},
  k = 1;
  n = 1;
  tbl = Reap[While[k < 6 && n < 25, n++;
      sol = NSolveValues [plane && surfaceEq && RandomReal [{0, 4}, 3].V == 3, V, Reals];
      If[Length[sol] > 0, k++; Sow[sol⟦1⟧]]]]⟦2, 1⟧;
  If[Length[tbl] < 5, Echo["5 points not found, try again or change equation"];
   Abort[]];
  a = tbl⟦1⟧;
  u = tbl⟦2⟧ - a;
  v = tbl⟦3⟧ - a;
  par = a + s * u + t * v;
  points = Table[First[SolveValues [tbl⟦i⟧ == par, {s, t}]], {i, 5}];
  If[! gpTestMD [points, 2, .003], Echo["Fails,try again"]; Abort[]];
  rnc = rncInterpolate [points, 1, 2];
  Simplify [par /. Thread[{s, t} → rnc⟦2⟧]]]
```

This is listed as paragraph 86 in **GlobalFunctions.nb.**

If we apply this to our example torus31 and a horizontal plane $z = c, -1 < c < 1$ we should get success.

In[ ]:=
```
conic1 = planeIntersectSurfaceNS [z == .7, torus31 == 0, {x, y, z}]
```

Out[ ]=
$$\left\{\frac{3.14809 + 6.23321\ t + 2.80623\ t^2}{0.858926 + 1.78349\ t + 1.\ t^2},\ \frac{-0.516455 - 2.9227\ t - 2.43309\ t^2}{0.858926 + 1.78349\ t + 1.\ t^2},\ 0.7\right\}$$

In[ ]:=
```
Simplify [torus31 /. Thread[{x, y, z} → conic1]]
```

Out[ ]=
$$\left(-9.9476 \times 10^{-14} - 6.82121 \times 10^{-13}\ t + 3.63798 \times 10^{-12}\ t^2 + 1.45519 \times 10^{-11}\ t^3 + 1.81899 \times 10^{-11}\ t^4 + 1.00044 \times 10^{-11}\ t^5 - 9.09495 \times 10^{-13}\ t^6 - 2.27374 \times 10^{-13}\ t^7 + 7.10543 \times 10^{-13}\ t^8\right) / \left(0.858926 + 1.78349\ t + 1.\ t^2\right)^4$$

On the other hand if we take a vertical plane through the line $\{x = 0,\ y = 0\}$ we should also get success (perhaps after several tries)

In[ ]:=
```
conic2 = planeIntersectSurfaceNS [2 x == 3 y, torus31 == 0, {x, y, z}]
```

Out[ ]=
$$\left\{\frac{0.142124 - 0.212296\ t + 1.69074\ t^2}{0.0427706 - 0.0753224\ t + 1.\ t^2},\ \frac{0.0947495 - 0.141531\ t + 1.12716\ t^2}{0.0427706 - 0.0753224\ t + 1.\ t^2},\ \frac{0.00480237 - 0.412591\ t + 0.25102\ t^2}{0.0427706 - 0.0753224\ t + 1.\ t^2}\right\}$$

In[ ]:=
```
Simplify [torus31 /. Thread[{x, y, z} → conic2]]
```

Out[ ]=
$$64 - 24\ \text{conic2}^2 + 9\ \text{conic2}^4$$

But if we take a random plane intersecting the torus we get

```
In[ ]:= plane3 = RandomReal[{-4, 4}, 3].{x, y, z} - .2
Out[ ]= -0.2 + 1.57775 x - 2.55377 y + 3.18539 z
```

```
In[ ]:= conic3 = planeIntersectSurfaceNS[plane3 == 0, torus31 == 0, {x, y, z}]
```

Out[ ]=
$$\left\{\frac{0.586069 - 1.46416\ t + 0.631606\ t^2}{0.320132 - 1.13073\ t + 1.\ t^2},\ \frac{0.901848 - 3.28888\ t + 2.90138\ t^2}{0.320132 - 1.13073\ t + 1.\ t^2},\ \frac{-0.298704 + 1.10884\ t - 0.99953\ t^2}{0.320132 - 1.13073\ t + 1.\ t^2}\right\}$$

```
In[ ]:= Chop[Simplify[torus31 /. Thread[{x, y, z} → conic3]]]
```

Out[ ]= $64 - 24\ \text{conic3}^2 + 9\ \text{conic3}^4$

So this function fails , indicating that there may be no such conic. One can try this last experiment many times and it will generally fail. There are few, if any, planes other than the horizontal and vertical ones which support conics. Thus the odds of finding a symmetry other than the ones in 4.2.1 are very small.

4.2.3 A lateral rotation

For the standard torus given by **torusEqab** each plane through the z-axis does intersect the torus in two circles. T hese have a rotation symmetry. It may appear that rotating each of these circles in the angle σ will give a symmetry of the torus. This is true but it is not a projective linear transformation. However we can give a function attaining this rotation using Mathematica. In fact each point is rotated by an affine transformation but for different points we must use a different affine transformation.

```
torusLateralRotNS [p_, σ_, a_, b_] :=
 Module[{rΘ, J, K, RΘ, SΘ, p1, p2, q1, Θ, p0, plane, planeRo, teqab, A},
  teqab = a^4 - 2 a^2 b^2 + b^4 - 2 a^2 x^2 - 2 b^2 x^2 + x^4 - 2 a^2 y^2 -
    2 b^2 y^2 + 2 x^2 y^2 + y^4 + 2 a^2 z^2 - 2 b^2 z^2 + 2 x^2 z^2 + 2 y^2 z^2 + z^4;
  If[Abs[teqab /. Thread[{x, y, z} → p]] > 1*^-6,
   Echo["p not on torus with parameters a,b"];
   Abort[]];
  rΘ = {{Cos[Θ], 0, Sin[Θ], 0}, {0, 1, 0, 0}, {-Sin[Θ], 0, Cos[Θ], 0}, {0, 0, 0, 1}};
  J = {{1, 0, 0, -a}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}};
  K = {{-1, 0, 0, -a}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}};
  RΘ = (Inverse[J].rΘ.J) /. {Θ → σ};
  SΘ = (Inverse[K].rΘ.K) /. {Θ → σ};
  plane = Chop[linearSetMD[{p, {0, 0, 0}, {0, 0, 2}}, {x, y, z}]][[1]];
  planeRo = planeRotate3D[plane, y];
  p0 = NSolveValues[plane == 0 && teqab == 0 && z == b && x ≥ 0, {x, y, z}, Reals][[1]];
  If[p0[[1]] * p0[[2]] > 0,
   A = Inverse[planeRo].RΘ.planeRo, A = Inverse[planeRo].SΘ.planeRo];
  q = fltMD[p, A];
  If[Abs[teqab /. Thread[{x, y, z} → q]] < 1.*^-9, Return[q], Return[{}]]]
```

Here **p** is a point on the torus with parameters a,b and *σ* is the lateral angle of rotation. In rare instances due perhaps numerical issues the point calculated is not close enough to the torus and the empty set is returned. So, for example, we could start with

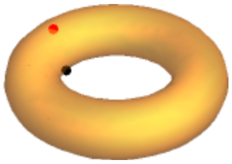
```
p = {-2.2554393726474875` , 0.2032078198223429` , -0.6776061916189526` };
```

Then its image is

```
q = torusLateralRotNS [p, 2 Pi / 3, 3, 1]
```

```
{-2.76967 , 0.249538 , 0.9757}
```

```
ImageCrop[Show[ContourPlot3D[torus31 == 0, {x, -4, 4},
    {y, -4, 4}, {z, -4, 4}, Mesh → None, ContourStyle → Opacity[.5]],
  Graphics3D[{{Black, PointSize[.03], Point[p]}, {Red, PointSize[.03], Point[q]}}],
  ImageSize → Small, Axes → False, Boxed → False]]
```



Here the black point *p* rotates to the red point *q*.

With this function we can't enter a parameterized curve for the argument but using the standard Mathematica formulation we can enter a list of points

```
torusLateralRotNS [♯, σ, a, b] & /@ L
```

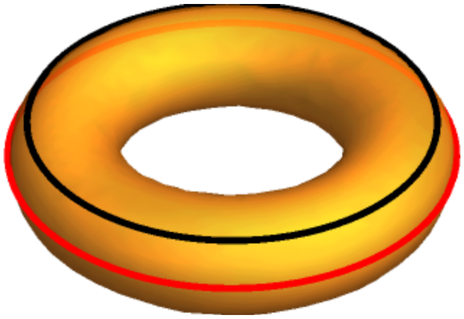
For example we may trace the horizontal circle of say height $z = .8$ on the **torus31 and then rotate by Pi/4.**

```
In[ ]:= FindInstance [torus31 == 0 && z == .8, {x, y, z}]
Out[ ]= {{x → -3.6, y → 0, z → 0.8}}
```

```
In[ ]:= L1 = pathFinder3D [{torus31 , z - .8}, {0, -3.6, .8}, {0, 3.6, .8}, .25, {x, y, z}, maxit → 60];
        L2 = pathFinder3D [{torus31 , z - .8}, {0, 3.6, .8}, {0, -3.6, .8}, .25, {x, y, z}, maxit → 60];
        L = Join[L1, L2];
        Ha = torusLateralRotNS [♯, Pi / 4, 3, 1] & /@ L;
        H = Reap[Do[If[Length[q] == 3, Sow[q]], {q, Ha}]][[2, 1]];
```

```
» not a point  {3.55255 , 0.582543 , 0.8}
» not a point  {-3.58438 , -0.33503 , 0.8}
```

```
In[ ]:= ImageCrop [Show[ContourPlot3D [torus31 == 0, {x, -4, 4},
           {y, -4, 4}, {z, -4, 4}, Mesh → None, ContourStyle → Opacity[.75]],
          Graphics3D [{{Black, Thickness [.01], Line[L]}, {Red, Thickness [.01], Line[H]}}],
          ImageSize → Medium, Axes → False, Boxed → False]]
Out[ ]=
```



While these lateral rotations are not projective linear transformations they can be composed with projective linear transformations as in section 4.2.1 to get a lateral symmetry of any projective torus, that is surface projectively equivalent to a torus. I will not illustrate that here. One application is that the group of symmetries including our projective linear symmetries and our lateral symmetries is now transitive.

4.2.4 A characterization of projective tori?

It follows from the above, from 4.3.2 or directly from our equations parametric or implicit that a property of our standard tori is that each point in contained in two circles lying on the torus. So each projective surface equivalent to a standard surface is contained in two plane conics. This is because plane conics are preserved by projective linear transformations.

An interesting question for further thought is whether the converse is true? Is a 4th degree surface in $\mathbb{R}^3$ such that each point is contained in two distinct conics necessarily a projective torus? A difficulty is that unlike the case of a quadric surface we don’t have a good handle on defining projective transformation to a torus. A second problem is that unlike the quadric surfaces there are infinitely many equivalence classes of the standard torus. But as in 1.4 three pairs of these conics may give an equation of these surfaces.

4.2.5 Some Variants on the torus idea

4.2.5.1 Elliptic Torus

The center line of the standard torus is a circle in the xy-plane about the origin. We can extend this to a torus-like surface with centerline an ellipse by transforming via a simple homothety that works only in the y-direction. This actually gives us some interesting examples. The following function takes positive numbers a1,a2 and b. Generally we think of $a1 > a2 > b$ but we will see that that is not necessary. In that case, however, the central ellipse will be

$$\frac{x^2}{a_1^2} + \frac{y^2}{a_2^2} = 1$$

We will see below that vertical planes through the z-axis do not necessarily cut out circles.

In[ ]:=

```
ellipticTorus[a1_, a2_, b1_] := Module[{toreq, tor1, A, tor, c, d},
  If[a1 ≤ 0 || a2 ≤ 0 || b1 ≤ 0, Echo["all values must be positive"];
   Abort[]];
  toreq = a^4 - 2 a^2 b^2 + b^4 - 2 a^2 x^2 - 2 b^2 x^2 + x^4 - 2 a^2 y^2 - 2 b^2 y^2 +
    2 x^2 y^2 + y^4 + 2 a^2 z^2 - 2 b^2 z^2 + 2 x^2 z^2 + 2 y^2 z^2 + z^4;
  tor1 = toreq /. {a → a1, b → b1};
  A = {{1, 0, 0, 0}, {0, a2 / a1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}};
  tor = FLTNS[tor1, A, {x, y, z}];
  c = Max[Norm /@ NSolveValues[{tor, x, z}, {x, y, z}, Reals]];
  d = Max[Norm /@ NSolveValues[{tor, y, z}, {x, y, z}, Reals]];
  Echo[{{d, 0, 0}, {0, c, 0}}, "Points on outer ellipse"];
  tor]
```

The picture is as follows, since we are interested in ratios it is best to plot this with all three ranges equal and the function echos a suggested value, typically, but not always, $a1 + b$.

In[ ]:=

```
etor753 = ellipticTorus[7, 5, 3]
```

» Points on outer ellipse {{10., 0, 0}, {0, 7.14286, 0}}

Out[ ]=

$$1600 - 116\, x^2 + x^4 - \frac{5684\, y^2}{25} + \frac{98\, x^2 y^2}{25} + \frac{2401\, y^4}{625} + 80\, z^2 + 2\, x^2 z^2 + \frac{98\, y^2 z^2}{25} + z^4$$

The important thing is that we have points $\{a1, 0, b\}$ and $\{a2, 0, b\}$ on this torus which gives the correct center ellipse.

```
In[ ∘ ]:= etor753 /. Thread[{x, y, z} → {7, 0, 3}]
          etor753 /. Thread[{x, y, z} → {0, 5, 3}]

Out[ ∘ ]= 0

Out[ ∘ ]= 0
```

The function ellipticTorus also echo’s two additional points on the torus, in this case

```
In[ ∘ ]:= etor753 /. Thread[{x, y, z} → {10, 0, 0}]
          etor753 /. Thread[{x, y, z} → {0, 7.142857142857143` , 0}]

Out[ ∘ ]= 0

Out[ ∘ ]= 0.
```

In this case the y value of the second point is $\frac{a1+b}{a1/a2}$ $= \frac{50}{7}$. But that may not always be the case, although it should be near a2+b which is 7 in this case.

The simplest way to plot the central ellipse $\frac{x^2}{7^2} + \frac{y^2}{5^2} = 1$ is to note that

{7, 0, 0}, {−7, 0, 0}, {0, 5, 0} and {0, −5, 0} lie on this ellipse as well as the solutions to

```
In[ ∘ ]:= sol = NSolveValues [{x ^ 2 / 49 + y ^ 2 / 25 - 1, x - y}, {x, y}, Reals]

Out[ ∘ ]= {{-4.06867 , -4.06867}, {4.06867 , 4.06867}}
```

so we can get a parametric function by

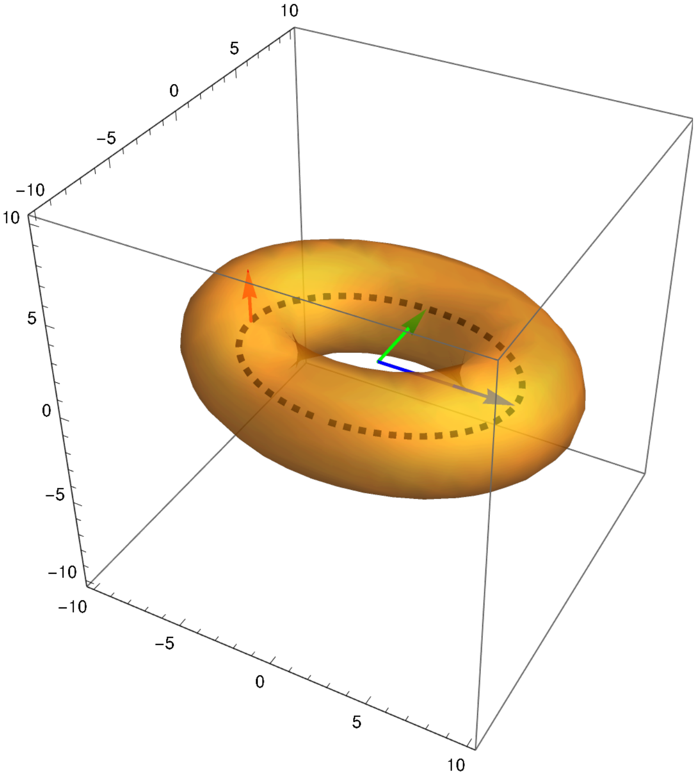
```
In[ ∘ ]:= cec = Chop[rncInterpolate [{{7, 0}, {-7, 0}, {0, 5}, {0, -5}, sol⟦1⟧}, 1, 2]⟦2⟧]
```

Out[∘]=
$$\left\{\frac{4.48651\ t}{-0.102698 - 1.\ t^2},\ \frac{-0.513489 + 5.\ t^2}{-0.102698 - 1.\ t^2}\right\}$$

We should use the maximum norm of the two points on the xy-plane and ellicptic torus echoed by the ellipticToris function to get the max and min for all components of the plot. These should be the same because we are interested in the ratios.

```
In[ ]:= Show[ContourPlot3D [etor753 == 0, {x, -10, 10}, {y, -10, 10}, {z, -10, 10},
          Mesh → None, ContourStyle → Opacity[.65]], ParametricPlot3D [{Append[cec, 0]},
          {t, -20, 20}, PlotStyle → Directive[{Black, Thickness[.01], Dashed}]],
        Graphics3D [{{Blue, Thickness[.005], Arrow[{{0, 0, 0}, {7, 0, 0}}]}, {Green, Thickness[.005],
            Arrow[{{0, 0, 0}, {0, 5, 0}}]}, {Red, Thickness[.005], Arrow[{{-7, 0, 0}, {-7, 0, 3}}]}}]]
```

Out[ ]=



To find the transverse conics we can use

```
In[ ]:= ce1 = planeIntersectSurfaceNS [y == 0, etor753 == 0 && x > 0, {x, y, z}]
```

Out[ ]=

$$\left\{\frac{17.6969 + 20.9185\ t + 6.88412\ t^2}{1.94042 + 2.55314\ t + 1.\ t^2},\ 0,\ \frac{-4.11851 - 7.7829\ t - 2.99776\ t^2}{1.94042 + 2.55314\ t + 1.\ t^2}\right\}$$

In[ ]:=

$$\text{ce1} = \left\{\frac{1.068604887036968` - 6.295225510389116`\ t + 9.53963066266547`\ t^2}{0.10688305225352232` - 0.6315609392728924`\ t + 1.`\ t^2},\ 0,\ \frac{-0.01202700100288967` + 0.5786133264878908`\ t - 1.596958389329299`\ t^2}{0.10688305225352232` - 0.6315609392728924`\ t + 1.`\ t^2}\right\};$$

```
ce2 = planeIntersectSurfaceNS [x == 0, etor753 == 0 && y > 0, {x, y, z}];
```

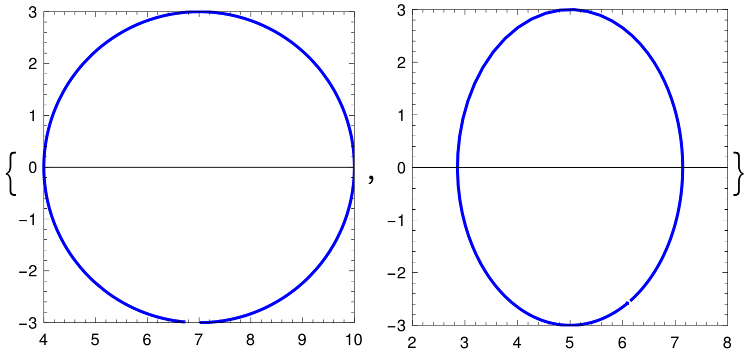
In[ ]:=

$$\text{ce2} = \left\{0,\ \frac{5.997745301899537` - 11.916147176615913`\ t + 6.122412060454718`\ t^2}{0.8702399822561371` - 1.8004006722223846`\ t + 1.`\ t^2},\ \frac{-1.2255935960394186` + 3.831952553651778`\ t - 2.555538042454029`\ t^2}{0.8702399822561371` - 1.8004006722223846`\ t + 1.`\ t^2}\right\};$$

We can plot this in the plane

```
In[ ]:= {ParametricPlot[Delete[ce1, {2}], {t, -20, 20}, PlotStyle → Blue,
    PlotRange → {{4, 10}, {-3, 3}}, ImageSize → Small, Frame → True],
  ParametricPlot[Take[ce2, -2], {t, -20, 30}, PlotStyle → Blue,
    PlotRange → {{2, 8}, {-3, 3}}, ImageSize → Small, Frame → True]}
```

Out[ ]=



The first is a circle but the second is clearly not a circle as the horizontal and vertical ranges of the plot have equal length. So these conics are not uniform around the elliptic torus.

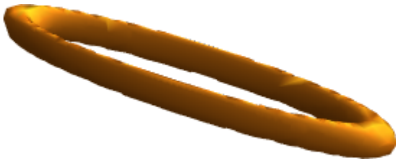
As an amusing application, it has been pointed out to me that the King James Bible describes Noah’s Ark as having dimensions 300, 50 and 30 cubits. Some scholars see as the source of this a Babylonian flood story which some have interpreted as being a raft rather than a boat, the shape then being a torus. This is unlikely as the the torus was not likely a Babylonian shape, much less an elliptic torus. But plotting a 30,5,3 elliptic torus we get

```
In[ ]:= NoahTorus = ellipticTorus[30, 5, 3]

  » Points on outer ellipse {{33., 0, 0}, {0, 5.5, 0}}
```

Out[ ]=
$$793\,881 - 1818\,x^2 + x^4 - 65\,448\,y^2 + 72\,x^2\,y^2 + 1296\,y^4 + 1782\,z^2 + 2\,x^2\,z^2 + 72\,y^2\,z^2 + z^4$$

```
In[ ]:= ImageCrop[ContourPlot3D[NoahTorus == 0, {x, -33, 33}, {y, -33, 33},
    {z, -33, 33}, Mesh → None, MaxRecursion → 4, Axes → False, Boxed → False]]
```

Out[ ]=



which is not a completely improbable shape for a raft, especially since Noah and the animals were sealed inside.

4.2.5.2 Octic Torus

Since all the monomials in a torus or ellipticTorus are even we can double the exponents of *x*, *y*, *z* to get a slightly different shape.

In[ ]:=
```
octicTorus = Expand[torusEqab /. {x → x^2, y → y^2, z → z^2}]
```

Out[ ]=

$$a^4 - 2 a^2 b^2 + b^4 - 2 a^2 x^4 - 2 b^2 x^4 + x^8 - 2 a^2 y^4 - 2 b^2 y^4 + 2 x^4 y^4 + y^8 + 2 a^2 z^4 - 2 b^2 z^4 + 2 x^4 z^4 + 2 y^4 z^4 + z^8$$

This gives us a squared off torus, not unlike a donut the author was served recently.

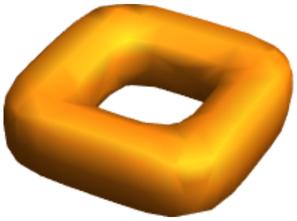
In[ ]:=
```
otorus32 = octicTorus /. {a → 3, b → 2}
```

Out[ ]=

$$25 - 26 x^4 + x^8 - 26 y^4 + 2 x^4 y^4 + y^8 + 10 z^4 + 2 x^4 z^4 + 2 y^4 z^4 + z^8$$

In[ ]:=
```
ImageCrop[ContourPlot3D[otorus32 == 0, {x, -4, 4},
  {y, -4, 4}, {z, -8, 8}, Mesh → None, Axes → False, Boxed → False]]
```

Out[ ]=



One can play this game on a specific Elliptic Torus. As example we can square the **NoahTorus** by

In[ ]:=
```
NoahRaft = ellipticTorus[3, .5, .3] /. {x → x^2, y → y^2, z → z^2}
```

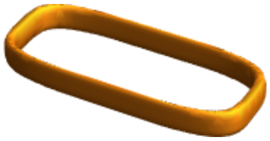
» Points on outer ellipse {{3.3 , 0, 0}, {0, 0.55 , 0}}

Out[ ]=

$$79.3881 - 18.18 x^4 + 1. x^8 - 654.48 y^4 + 72. x^4 y^4 + 1296. y^8 + 17.82 z^4 + 2. x^4 z^4 + 72. y^4 z^4 + 1. z^8$$

In[ ]:=
```
ImageCrop[ContourPlot3D[NoahRaft == 0, {x, -3, 3}, {y, -3, 3},
  {z, -8, 8}, Mesh → None, Axes → False, Boxed → False, MaxRecursion → 4]]
```

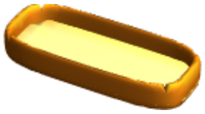
Out[ ]=



To make this more like a raft we could add a bottom

In[ ]:=
```
ImageCrop[ Show[ContourPlot3D[NoahRaft == 0,
   {x, -4, 4}, {y, -4, 4}, {z, -8, 8}, Mesh → None, MaxRecursion → 4],
  RegionPlot3D[x^4/3.3^2 + y^4/.55^2 ≤ 1 && -.55 < z < -.1, {x, -2, 2}, {y, -2, 2},
   {z, -2, 2}, Mesh → None, ColorFunction → Blue], Axes → None, Boxed → False]]
```

Out[ ]=



4.2.5.3 Symmetries of elliptic and octic tori

The elliptic tori are projective linear transforms of standard tori so any projective symmetry of the latter will transport to them. Note that the transform from the standard torus torusEqab to the elliptic torus ellipicTorus[a1,a2,b1] is just

```
J := {{1, 0, 0, 0}, {0, a2 / a1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}}
```

So, for example, consider the elliptic torus above

```
etor753 = ellipticTorus[7, 5, 3]
```

» Points on outer ellipse {{10., 0, 0}, {0, 7.14286, 0}}

$$1600-116\,x^2+x^4-\frac{5684\,y^2}{25}+\frac{98\,x^2\,y^2}{25}+\frac{2401\,y^4}{625}+80\,z^2+2\,x^2\,z^2+\frac{98\,y^2\,z^2}{25}+z^4$$

Here

```
J753 = J /. {a1 → 7, a2 → 5}
```

$$\left\{\{1,0,0,0\},\left\{0,\frac{5}{7},0,0\right\},\{0,0,1,0\},\{0,0,0,1\}\right\}$$

Let

```
sym1 = N[torusSym[-6 Pi / 11, 1, 1]]
```

```
{{-0.142315, 0.989821, 0., 0.},
 {-0.989821, -0.142315, 0., 0.}, {0., 0., 1., 0.}, {0., 0., 0., 1.}}
```

```
esym1 = J753.sym1.Inverse[J753]
```

```
{{-0.142315, 1.38575, 0., 0.},
 {-0.707015, -0.142315, 0., 0.}, {0., 0., 1., 0.}, {0., 0., 0., 1.}}
```

```
Chop[etor753 - FLTNS[etor753, esym1, {x, y, z}]]
```

```
0
```

so we have a symmetry.

Consider our curve

$$\text{ce1}=\left\{\frac{7.456123843292054\grave{}\;-16.93950360896598\grave{}\;t+9.642705601053876\grave{}\;t^2}{0.7919997073644772\grave{}\;-1.7753725463339702\grave{}\;t+1.\grave{}\;t^2},\,0,\,\frac{-1.4103710158407021\grave{}\;+2.8556585740233564\grave{}\;t-1.4198968646202492\grave{}\;t^2}{0.7919997073644772\grave{}\;-1.7753725463339702\grave{}\;t+1.\grave{}\;t^2}\right\};$$

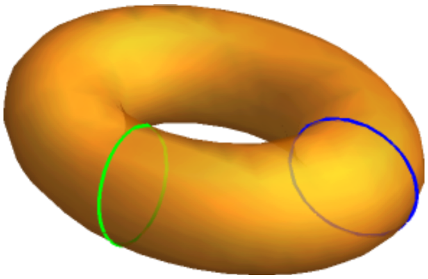
we have

```
ece1 = fltMD[ce1, esym1]
```

$$\left\{0.-\frac{0.142315\times(7.45612-16.9395\,t+9.64271\,t^2)}{0.792-1.77537\,t+1.\,t^2},\;0.-\frac{0.707015\times(7.45612-16.9395\,t+9.64271\,t^2)}{0.792-1.77537\,t+1.\,t^2},\;\frac{1.\times(-1.41037+2.85566\,t-1.4199\,t^2)}{0.792-1.77537\,t+1.\,t^2}\right\}$$

```
In[ ∘ ]:= ImageCrop[Show[ContourPlot3D[etor753 == 0, {x, -10, 10},
            {y, -10, 10}, {z, -10, 10}, ContourStyle → Opacity[.75], Mesh → None],
          ParametricPlot3D[ce1, {t, -20, 20}, PlotStyle → Blue],
          ParametricPlot3D[ece1, {t, -20, 20}, PlotStyle → Green], Axes → False, Boxed → False]]
```

Out[∘]=



In[∘]:=

As in 4.2.3 we can port the lateral rotations to elliptic tori since they are projectively equivalent to standard tori. But again remember that these symmetries are not projective linear symmetries.

In[∘]:=

The symmetries above are not symmetries of octic tori but one can check that the symmetry group containing the symmetries `Hyp4[i,1,j,k,1]` for $i = 1, 2,\ j, k = 1, 2, 3, 4$ are symmetries, for example

```
In[ ∘ ]:= otorusS = FLTNS[otorus32, Hyp4[2, 1, 3, 4, 1], {x, y, z}] - otorus32
Out[ ∘ ]= 0
```

Note by normalizing

```
In[ ∘ ]:= OtSym = DeleteDuplicates[Flatten[Table[Hyp4N[i, 1, j, k, 1], {i, 2}, {j, 4}, {k, 4}], 2]]
Out[ ∘ ]= {{{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}},
           {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, -1, 0}, {0, 0, 0, 1}},
           {{-1, 0, 0, 0}, {0, -1, 0, 0}, {0, 0, -1, 0}, {0, 0, 0, 1}},
           {{-1, 0, 0, 0}, {0, -1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}},
           {{-1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}},
           {{-1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, -1, 0}, {0, 0, 0, 1}},
           {{1, 0, 0, 0}, {0, -1, 0, 0}, {0, 0, -1, 0}, {0, 0, 0, 1}},
           {{1, 0, 0, 0}, {0, -1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}},
           {{0, 1, 0, 0}, {1, 0, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}},
           {{0, 1, 0, 0}, {1, 0, 0, 0}, {0, 0, -1, 0}, {0, 0, 0, 1}},
           {{0, -1, 0, 0}, {-1, 0, 0, 0}, {0, 0, -1, 0}, {0, 0, 0, 1}},
           {{0, -1, 0, 0}, {-1, 0, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}},
           {{0, -1, 0, 0}, {1, 0, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}},
           {{0, -1, 0, 0}, {1, 0, 0, 0}, {0, 0, -1, 0}, {0, 0, 0, 1}},
           {{0, 1, 0, 0}, {-1, 0, 0, 0}, {0, 0, -1, 0}, {0, 0, 0, 1}},
           {{0, 1, 0, 0}, {-1, 0, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}}}
```

```
In[ ]:= Table[FLTNS[otorus32, OtSym⟦i⟧, {x, y, z}] - otorus32, {i, 16}]
Out[ ]= {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

So we have 16 distinct symmetries of the otic torus.

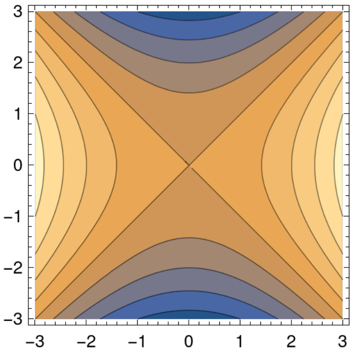
4.3 Gluing surfaces

I already noticed in my *Plane Curve Book* that singularities of curves could be removed by adding or subtracting a constant to the equation. One way to see this is to look at the actual contour plot. A simple example is the curve $x^2 - y^2 = 0$.

```
In[ ]:= Clear[x, y]
```

```
In[ ]:= ContourPlot[x^2 - y^2, {x, -3, 3}, {y, -3, 3}, ImageSize → Small]
```

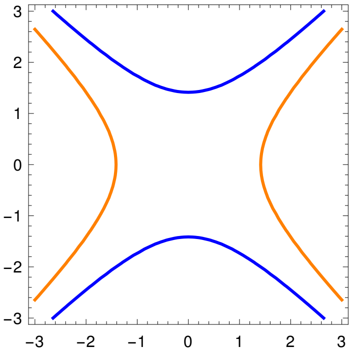
Out[ ]=



The default is $x^2 - y^2 == 0$, adding or subtracting a constant moves the curve to a different contour which should not contain a singularity.

```
In[ ]:= ContourPlot[{x^2 - y^2 == 2, x^2 - y^2 == -2}, {x, -3, 3},
         {y, -3, 3}, ContourStyle → {Orange, Blue}, ImageSize → Small]
```

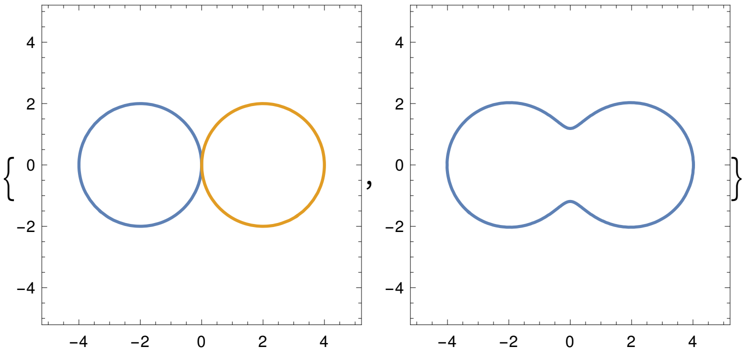
Out[ ]=



Another example is we can glue two curves together to make a non-singular curve . Here are two circles

```
In[ ]:= {ContourPlot[{(x + 2) ^ 2 + y ^ 2 == 4, (x - 2) ^ 2 + y ^ 2 == 4},
           {x, -5, 5}, {y, -5, 5}, ImageSize → Small], ContourPlot[
           ((x + 2) ^ 2 + y ^ 2 - 4) ((x - 2) ^ 2 + y ^ 2 - 4) - 2 == 0, {x, -5, 5}, {y, -5, 5}, ImageSize → Small]}
```

Out[ ]=



The same holds for surfaces but contour plots of surfaces showing different contours are usually ugly.

4.3.1 The double Torus

Likewise we can glue two tori to make a double torus.

```
In[ ]:= tor = Expand[torusEqab /. {a → 2, b → 1}]
```

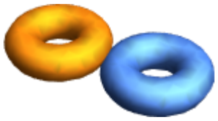
Out[ ]= $9 - 10 x^2 + x^4 - 10 y^2 + 2 x^2 y^2 + y^4 + 6 z^2 + 2 x^2 z^2 + 2 y^2 z^2 + z^4$

Note this torus contains the points {-3, 0, 0} and {3, 0, 0} . So translating by ±3 in the x-direction gives us two two tori touching the origin

```
In[ ]:= torp = FLTNS[tor, {{1, 0, 0, -3}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}}, {x, y, z}];
        torm = FLTNS[tor, {{1, 0, 0, 3}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}}, {x, y, z}];
```

```
In[ ]:= ImageCrop[ContourPlot3D[{torp == 0, torm == 0}, {x, -6, 6}, {y, -6, 6},
           {z, -6, 6}, Mesh → None, Axes → False, Boxed → False, ImageSize → Small]]
```

Out[ ]=



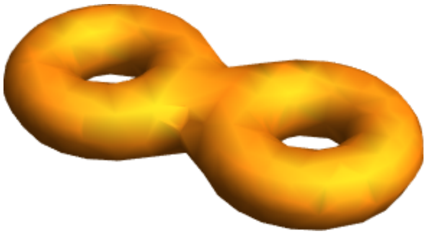
So we get the double torus

```
In[ ]:= doublTorus = torp * torm - 600
```

Out[ ]= $-600 + \left(-48 x + 44 x^2 - 12 x^3 + x^4 + 8 y^2 - 12 x y^2 + 2 x^2 y^2 + y^4 + 24 z^2 - 12 x z^2 + 2 x^2 z^2 + 2 y^2 z^2 + z^4\right) \times \left(48 x + 44 x^2 + 12 x^3 + x^4 + 8 y^2 + 12 x y^2 + 2 x^2 y^2 + y^4 + 24 z^2 + 12 x z^2 + 2 x^2 z^2 + 2 y^2 z^2 + z^4\right)$

In[ ]:=
```
ImageCrop[ContourPlot3D[doublTorus == 0, {x, -6, 6},
  {y, -6, 6}, {z, -6, 6}, Mesh → None, Axes → False, Boxed → False]]
```

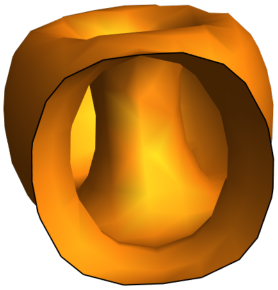
Out[ ]=



We can look inside

In[ ]:=
```
ContourPlot3D[doublTorus == 0, {x, -6, 1}, {y, -6, 6},
 {z, -3, 3}, Mesh → None, Axes → False, Boxed → False]
```

Out[ ]=



Thus the interior is connected but not simply connected. I note that we actually get a nice plane curve of degree 8 out of this, an oval with two nested ovals.

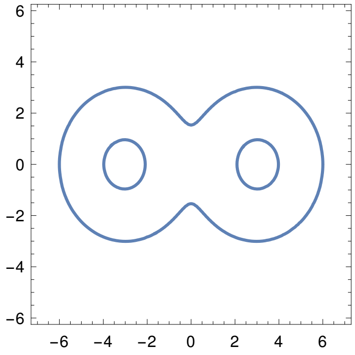
In[ ]:=
```
curveDT = Expand[doublTorus /. {z → 0}]
```

Out[ ]=

$$-600 - 2304\,x^2 + 784\,x^4 - 56\,x^6 + x^8 - 448\,x^2\,y^2 - 96\,x^4\,y^2 + 4\,x^6\,y^2 + 64\,y^4 - 24\,x^2\,y^4 + 6\,x^4\,y^4 + 16\,y^6 + 4\,x^2\,y^6 + y^8$$

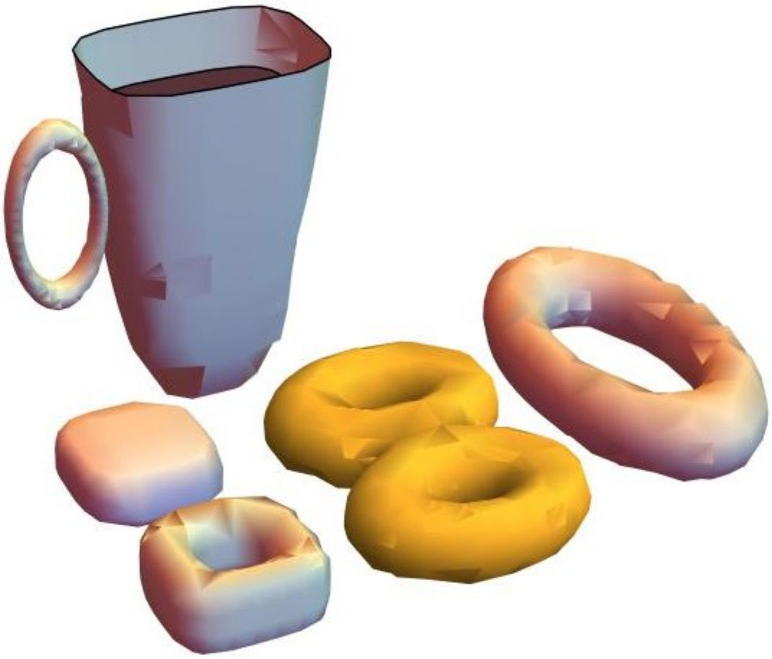
```
In[ ]:= ContourPlot[curveDT == 0, {x, -7, 7}, {y, -6, 6}, ImageSize → Small]
```

Out[ ]=



4.4 Breakfast with Barry

In[ ]:=



My selection of Donuts is a jelly donut (back left), octic donut, (front left) a double donut and an elliptic donut. All objects in this graphic are made from 100% fourth and eighth degree surfaces. Note however that the jelly donut is of smaller degree and has more symmetry than the octic donut.

Here is the code for the graphic.

```
In[ ]:= CoffeeCup = (0.0625` x^4 + 11.24682650380698` y +
          4.743416490252569` y^2 + 0.8891397050194614` y^3 + 0.0625` y^4 - 1.25` z) ×
       (262.5` x^2 + 2.44140625` x^4 - 1344.` y - 175.` x^2 y + 3520.` y^2 + 50.` x^2 y^2 -
          1792.` y^3 + 256.` y^4 + 96.` z^2 + 12.5` x^2 z^2 - 448.` y z^2 + 128.` y^2 z^2 + 16.` z^4);
```

```
In[ ]:= DoubleDonut =
     3.423774655931445`*^8 + 1.3655528886821947`*^8 x + 2.6047110685191058`*^7 x^2 +
3.02080968449735`*^6 x^3 + 230293.87029144645` x^4 + 11722.824430539758` x^5 +
387.4700098060629` x^6 + 7.589160493137579` x^7 + 0.06776036154587124` x^8 +
7.745211966466844`*^7 y + 2.307104403777339`*^7 x y + 3.1919077874500793`*^6 x^2 y +
252437.3347839762` x^3 y + 11945.18683298869` x^4 y + 318.7447407117784` x^5 y +
3.79458024656879` x^6 y + 1.1841404708939653`*^7 y^2 + 2.8843808155736877`*^6 x y^2 +
334626.2789470098` x^2 y^2 + 22493.664568820335` x^3 y^2 + 932.9138161057081` x^4 y^2 +
22.76748147941274` x^5 y^2 + 0.27104144618348497` x^6 y^2 + 1.1161226979040564`*^6 y^3 +
197460.24190600865` x y^3 + 15977.00423657296` x^2 y^3 + 637.4894814235568` x^3 y^3 +
11.383740739706369` x^4 y^3 + 80825.53651895358` y^4 + 10770.84013828058` x y^4 +
703.4176027932275` x^2 y^4 + 22.767481479412737` x^3 y^4 + 0.4065621692752274` x^4 y^4 +
4031.817403584271` y^5 + 318.7447407117784` x y^5 + 11.383740739706369` x^2 y^5 +
157.97379649358237` y^6 + 7.589160493137581` x y^6 + 0.27104144618348497` x^2 y^6 +
3.7945802465687892` y^7 + 0.06776036154587124` y^8 + 7.140206420134889`*^7 z +
2.0700852050226893`*^7 x z + 2.808963995461423`*^6 x^2 z + 219230.81126304282` x^3 z +
10289.730729647046` x^4 z + 273.20977775295285` x^5 z + 3.25249735420182` x^6 z +
1.1228074623911565`*^7 y z + 2.1596103664289545`*^6 x y z + 184227.17453733447` x^2 y z +
7649.87377708268` x^3 y z + 136.60488887647642` x^4 y z + 1.3643985507912594`*^6 y^2 z +
207806.99975593266` x y^2 z + 15071.552339794564` x^2 y^2 z + 546.4195555059057` x^3 y^2 z +
9.757492062605456` x^4 y^2 z + 89266.74138448146` y^3 z + 7649.873777082679` x y^3 z +
273.20977775295285` x^2 y^3 z + 4781.8216101475155` y^4 z + 273.2097777529528` x y^4 z +
9.757492062605458` x^2 y^4 z + 136.60488887647642` y^5 z + 3.25249735420182` y^6 z +
1.0948567163463235`*^7 z^2 + 2.6677540213443628`*^6 x z^2 + 313646.82623042853` x^2 z^2 +
21547.7516049 5567` x^3 z^2 + 916.0225131795534` x^4 z^2 + 22.76748147941274` x^5 z^2 +
0.27104144618348497` x^6 z^2 + 1.4262920758236062`*^6 y z^2 + 225866.77319824233` x y z^2 +
16991.523211295593` x^2 y z^2 + 637.4894814235568` x^3 y z^2 + 11.383740739706369` x^4 y z^2 +
160218.91779091774` y^2 z^2 + 20595.76731269649` x y^2 z^2 + 1373.0525997341456` x^2 y^2 z^2 +
45.53496295882548` x^3 y^2 z^2 + 0.8131243385504549` x^4 y^2 z^2 + 9078.153781891173` y^3 z^2 +
637.4894814235566` x y^3 z^2 + 22.767481479412737` x^2 y^3 z^2 + 457.03008655459234` y^4 z^2 +
22.767481479412737` x y^4 z^2 + 0.813124338550455` x^2 y^4 z^2 + 11.383740739706369` y^5 z^2 +
0.27104144618348497` y^6 z^2 + 975690.7289601153` z^3 + 170227.9055252731 8` x z^3 +
13729.44183155672` x^2 z^3 + 546.4195555059056` x^3 z^3 + 9.757492062605456` x^4 z^3 +
88326.8997490113` y z^3 + 7649.873777082679` x y z^3 + 273.2097777529528` x^2 y z^3 +
8221.532712057191` y^2 z^3 + 546.4195555059056` x y^2 z^3 + 19.514984125210912` x^2 y^2 z^3 +
273.2097777529528` y^3 z^3 + 9.757492062605456` y^4 z^3 + 71772.37405939946` z^4 +
9824.92717441591` x z^4 + 669.6349969409181` x^2 z^4 + 22.767481479412737` x^3 z^4 +
0.4065621692752274` x^4 z^4 + 5046.336378306903` y z^4 + 318.7447407117784` x y z^4 +
11.383740739706369` x^2 y z^4 + 440.13878362843764` y^2 z^4 + 22.767481479412737` x y^2 z^4 +
0.8131243385504549` x^2 y^2 z^4 + 11.383740739706369` y^3 z^4 + 0.4065621692752274` y^4 z^4 +
3439.711101909677` z^5 + 273.20977775295285` x z^5 + 9.757492062605458` x^2 z^5 +
136.60488887647642` y z^5 + 9.757492062605458` y^2 z^5 + 141.0824935674276` z^6 +
7.589160493137581` x z^6 + 0.27104144618348497` x^2 z^6 + 3.79458024656879` y z^6 +
0.27104144618348497` y^2 z^6 + 3.252497354201819` z^7 + 0.06776036154587124` z^8;
```

```
In[•]:= Coffee = 0.0625 x^4 + 11.24682650380698` y +
    4.743416490252569` y^2 + 0.8891397050194614` y^3 + 0.0625` y^4 - 1.25` z;
```

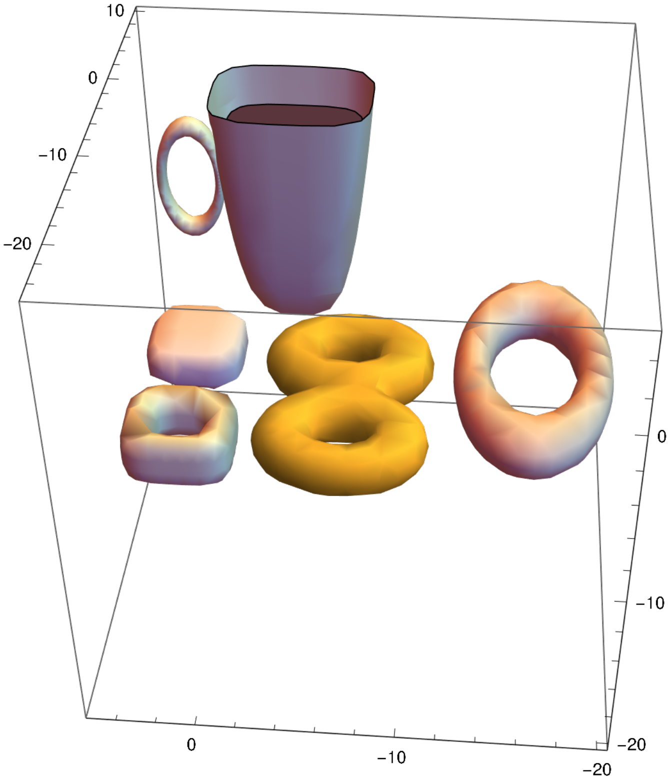
```
In[•]:= EllipticDonut = 9.466790363338271`*^6 + 233512.40375308643` x + 12359.174320987653` x^2 +
    135.26913580246912` x^3 + 3.1604938271604937` x^4 + 2.1823589135802467`*^6 y +
    27053.827160493827` x y + 1264.1975308641975` x^2 y + 194618.4691358025` y^2 +
    845.4320987654321` x y^2 + 39.50617283950617` x^2 y^2 + 7901.234567901234` y^3 +
    123.45679012345678` y^4 + 298127.53066666663` z + 3469.653333333333` x z +
    162.13333333333333` x^2 z + 32426.66666666666` y z + 1013.3333333333331` y^2 z +
    28230.897777777776` z^2 + 304.3555555555555` x z^2 + 14.222222222222221` x^2 z^2 +
    2844.4444444444443` y z^2 + 88.88888888888889` y^2 z^2 + 364.8` z^3 + 16.` z^4;
```

```
In[•]:= OcticDonut = 3.469045206798153`*^8 + 1.4458678390406924`*^8 x +
    2.782662290002328`*^7 x^2 + 3.1595294956295667`*^6 x^3 + 228959.79098600807` x^4 +
    10735.065996037181` x^5 + 315.7372351775643` x^6 + 5.306508154244778` x^7 +
    0.039018442310623375` x^8 - 36786.78605103493` y - 7667.904282897711` x y -
    676.5797896655276` x^2 y - 26.53254077123711` x^3 y - 0.3901844231054383` x^4 y +
    68976.19931089878` y^2 + 14377.320530408528` x y^2 + 1268.587105624203` x^2 y^2 +
    49.748513946042294` x^3 y^2 + 0.7315957933242032` x^4 y^2 - 57483.01118730754` y^3 -
    11981.10044200579` x y^3 - 1057.1559213534624` x^2 y^3 - 41.45709495503738` x^3 y^3 -
    0.6096631611034908` x^4 y^3 + 17968.77554869617` y^4 + 3744.093888126808` x y^4 +
    330.36122542295374` x^2 y^4 + 12.955342173449164` x^3 y^4 + 0.1905197378448407` x^4 y^4 -
    6.668190824544581` y^5 + 4.167619265355597` y^6 - 1.4884354519128173` y^7 +
    0.2325680393613778` y^8 + 4.6891396192918494`*^7 z + 9.76468964187654`*^6 x z +
    861590.2625185181` x^2 z + 33787.85343209876` x^3 z + 496.88019753086417` x^4 z -
    2484.400987662375` y z + 4658.251851852518` y^2 z - 3881.8765432098953` y^3 z +
    1213.0864197530864` y^4 z + 1.192557005865718`*^7 z^2 + 2.1539756562962956`*^6 x z^2 +
    190056.67555555553` x^2 z^2 + 7453.2029629629615` x^3 z^2 + 109.60592592592592` x^4 z^2 -
    548.0296296291053` y z^2 + 1027.555555555562` y^2 z^2 - 856.2962962962993` y^3 z^2 +
    267.59259259259255` y^4 z^2 + 1.7119754005491352`*^6 z^3 + 211174.0839506172` x z^3 +
    18633.007407407404` x^2 z^3 + 730.706172839506` x^3 z^3 + 10.74567901234568` x^4 z^3 -
    53.72839506203309` y z^3 + 100.74074074074815` y^2 z^3 - 83.95061728395103` y^3 z^3 +
    26.23456790123457` y^4 z^3 + 182675.9776790124` z^4 + 7763.75308641975` x z^4 +
    685.0370370370368` x^2 z^4 + 26.864197530864192` x^3 z^4 + 0.3950617283950617` x^4 z^4 -
    1.9753086419768806` y z^4 + 3.7037037037036953` y^2 z^4 - 3.0864197530864246` y^3 z^4 +
    0.9645061728395061` y^4 z^4 + 17608.192` z^5 + 1294.72` z^6 + 54.4` z^7 + 1.` z^8;
```

```
In[•]:= jellyDonut = 998.4833359433574` + 9.142857142857139` x +
    1.959183673469388` x^2 + 0.18658892128279878` x^3 + 0.006663890045814243` x^4 -
    0.12851787945498896` y + 0.18359697064998423` y^2 -
    0.11656950517459316` y^3 + 0.02775464408918885` y^4 + 561.9712000000002` z +
    120.42240000000002` z^2 + 11.468800000000003` z^3 + 0.40960000000000002` z^4;
```

In[ ]:=
```
Show[ContourPlot3D[{CoffeeCup == 2}, {x, -25, 10}, {y, -20, 5},
   {z, -20, 5}, Mesh → None, MaxRecursion → 5, ContourStyle → LightYellow],
  ContourPlot3D[DoubleDonut == 0, {x, -23, 5}, {y, -20, 5}, {z, -15, 5},
   Mesh → None, ColorFunction → ColorData[1, "ColorList"]],
  RegionPlot3D[Coffee < -5, {x, -5, 5}, {y, -10, 4}, {z, -15, 3},
   Mesh → None, ColorFunction → ColorData[12, "ColorList"]],
  ContourPlot3D[{EllipticDonut == 0, OcticDonut == 0}, {x, -25, 10}, {y, -20, 5}, {z, -18, 5},
   Mesh → None, MaxRecursion → 6, ContourStyle → {LightOrange, LightYellow}],
  ContourPlot3D[jellyDonut == 0, {x, -25, 10}, {y, -20, 5}, {z, -20, 5},
   Mesh → None, ContourStyle → LightPink, MaxRecursion → 6]]
```

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.

Joe Harris , Algebraic Geometry, A First Course, Springer Graduate Texts in Mathematics 133, 2010 edition.

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/**

Paul B . Yale, *Geometry and Symmetry,* Holden Day, 1968, (Also Dover, 1988)