Interpolating Rational Normal Curves

Barry H Dayton
https://barryhdayton.space

Note: This is a preliminary version. Check back at my website for updated versions.

This note is an exposition of Theorem 1.18 in Joe Harris' Algebraic Geometry *book which says that given $d+3$ points in general position in projective d-space there is a unique degree d rational normal curve passing through these points. A rational normal curve (RNC) is a curve projectively equivalent to the projective closure of the parametric rational normal curve $\{t^d, t^{d-1}, \ldots, t\}$. Harris' exposition is a bit sparse on implementation details and does not adequately address the uniqueness. I will be able to explain uniqueness heuristically, although will not give a formal proof except for $d = 2$. In addition I will be able to explain why the hypotheses on general position is necessary and sufficient for a set of $d+3$ points to be on a RNC. I will also give insight as to why $d+3$ is the correct number. This is presumably a classical result but other expositions I have seen seem to follow Harris. As always I will use a numerical approach using Mathematica.*

In my Mathematica-Journal article [2] I identify three numbers, d, n, r for a rationally parameterized curve with common denominator. n is the affine dimension of the space, i.e. the number of affine components. d is the maximal degree of the parameter t in the numerator or common denominator and r is the dimension of the space of polynomials in the numerator or common denominator. A affine parametric curve is a rational normal curve if $d = n = r - 1$. In this note we will work in the real or complex setting. A set of points is in *general position* if no $k+1$ of these points line in the same k-dimensional subset of projective n-space.

In this note I will give Mathematica functions to check the general position hypotheses and to construct a parametric rational normal curve through a given general position subset of $d+3$ points. I will start with the two dimensional case where both existence and uniqueness follow easily from [3], give an algorithm to test general position, then give a not so simple description of my simple version of Harris' existence construction with code and finally address, but not give a formal proof, of the uniqueness construction.

One main tool I will use are projective linear transformations which I call in [3,4] fractional linear transformations in the affine case, FLT. On a point basis these are given by Mathematica as *transformation Functions* or just Geometric Transforms. Specifically transformationfunctions are defined from m-space to n-space by transformation matrices M which are any (m+1) × (n+1) matrix. The matrix M is viewed in the form $\begin{pmatrix} A & b \\ c & e \end{pmatrix}$, A is a m×n matrix, b a m×1 matrix, c a 1×m matrix and e a scalar. Then TransformationMatrix[M][x] gives $\frac{Ax+b}{cx+e}$ where the denominator divides each coordinate of the numera-

tor. In my books I use the shorthand

```
flt[x, A] = TransformationMatrix[M][x]
```

Where I digress from Mathematica is that I have companion functions FLT which take not the points to points but curves to curves equation wise. Loosely, if $C_2 = \text{FLT}[C_1, M]$ then each point p in C_1 goes to the point flt[p,M] in C_2, the actual syntax depends on the context, see [3,4]. When the transformation matrix is not a square invertible matrix FLT is a complicated function whose definition takes up much of my Space Curve book [4].

2. Two dimensional Case

We must show that given 5 plane points in general position there is a unique rational normal quadratic curve through these points. We start with the function aCurve in section 2.4 of my plane curve book [3] which gives a unique algebraic curve $f = 0$ through these points. We noticed already in Chapter 1 of that book that a singular quadric curve consists of two, not necessarily distinct, lines. But to place 5 points on 2 lines one of the lines must contain at least 3 points which is not allowed since the points are in general position. So we have a conic. In Section 7.3 of my Plane Curve book I show how to transform any conic to the parabola $y = x^2$. This is easily parameterized by {t,t^2}. Using the inverse transforms we get a parameterization of the curve $f = 0$ which is our RNC (rational normal curve).

Here is an example using functions from my Appendix 2 from the Plane Curves Book [3].

In[ ]:=
```
S2 = {{-1, 1}, {2, 0}, {1, 1}, {-1, -1}, {1, -1}};
```

We first find the unique quadratic plane curve through these points.

In[ ]:=
```
f2eq = Chop[aCurve[S2, x, y]]
```

Out[ ]=
$$1.30742 - 0.326855\ x^2 - 0.980565\ y^2$$

This is an ellipse. Choosing our first point we use the transformation **cTransform** to place this point at the infinite point {0,1,0} with the infinite line as the tangent line at this point.

In[ ]:=
```
A1 = cTransform[f2eq, S2[[1]], x, y];
A1 // MatrixForm
```

Out[ ]//MatrixForm=
$$\begin{pmatrix} -0.792594 & -0.566139 & -0.226455 \\ -0.57735 & 0.57735 & 0.57735 \\ 0.196116 & -0.588348 & 0.784465 \end{pmatrix}$$

We apply this transform to our curve.

In[ ]:=
```
f2eq2 = FLT[f2eq, A1, x, y]
```

Out[ ]=
$$0.452568 - 1.01613\ x - 0.452568\ x^2 + 1.92447\ y$$

We see this is a parabola with vertical axis. We use the function tangentRealPoints to find the vertex.

In[ ]:=
```
p1 = tangentRealPoints [f2eq2, y, x, y][[1]]
```

Out[ ]=
```
{-1.12263, -0.531541}
```

We translate this point to the origin.

```
In[ ]:= A2 = {{1, 0, -p1[[1]]}, {0, 1, -p1[[2]]}, {0, 0, 1}};
        f2eq3 = FLT[f2eq2, A2, x, y]
```

Out[ ]= $-0.452568\ x^2 + 1.92447\ y$

We clean up the coefficients

```
In[ ]:= A3 = {{-Sqrt[Abs[Coefficient[f2eq3, x^2]]], 0, 0}, {0, Coefficient[f2eq3, y], 0}, {0, 0, 1}};
        A3 // MatrixForm
        f2eq4 = FLT[f2eq3, A3, x, y]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -0.672732 & 0 & 0 \\ 0 & 1.92447 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

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

This is parameterized by

```
In[ ]:= f2a = {t^2, t};
```

So we reverse our steps

```
In[ ]:= A4 = Inverse[A3.A2.A1];
        A4 // MatrixForm
```

Out[ ]//MatrixForm=

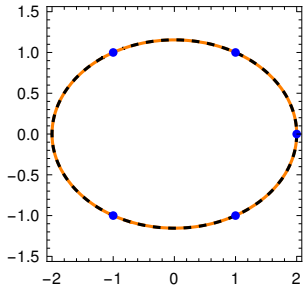
$$\begin{pmatrix} 1.17817 & -0.300005 & 1.39279 \\ 0.841551 & 0.300005 & -0.259672 \\ 0.33662 & 0.300005 & 0.731804 \end{pmatrix}$$

```
In[ ]:= f2p = TransformationFunction [A4][{t, t^2}]
```

Out[ ]= $\left\{\frac{1.39279 + 1.17817\ t - 0.300005\ t^2}{0.731804 + 0.33662\ t + 0.300005\ t^2},\ \frac{-0.259672 + 0.841551\ t + 0.300005\ t^2}{0.731804 + 0.33662\ t + 0.300005\ t^2}\right\}$

```
In[ ]:= Show[ContourPlot[f2eq == 0, {x, -2, 2}, {y, -1.5, 1.5}, ContourStyle → Orange],
         ParametricPlot[f2, {t, -30, 30}, PlotStyle → Directive[Dashed, Black]],
         Graphics[{Blue, PointSize[Medium], Point[S2]}]]
```

Out[ ]=



Our desired parameterization is f2p.

Here is the general function. Initialize Appendix2: GlobalFunctions.nb from the Plane Curve page

before running.

In[ ]:=
```
nrc2[S_] := Module[{l, SS3, tst, i, acurve, ct, T, g1, g2, a, p, par, f},
  SS3 = Subsets[S, {3}];
  tst = False;
  i = 1;
  While[i ≤ 10 && tst == False,
   tst = Abs[(line[SS3[[i, 1]], SS3[[i, 2]], x, y] /. Thread[{x, y} → SS3[[i, 3]])] < .001;
   i++];
  If[i < 11, Echo["Not General Position"]; Abort[]];
  acurve = aCurve[S, x, y];
  ct = cTransform[acurve, S[[1]], x, y];
  g1 = FLT[acurve, ct, x, y];
  g1 = Expand[g1 / Coefficient[-g1, y]];
  p = tangentRealPoints [g1, y, x, y][[1]];
  T = {{1, 0, -p[[1]]}, {0, 1, -p[[2]]}, {0, 0, 1}};
  g2 = FLT[g1, T, x, y];
  a = Coefficient[g2, x^2];
  par = {t, a t^2};
  f = flts[{t, a t^2}, Inverse[T.ct]];
  {Chop[acurve], f}]
```

Using S2 above

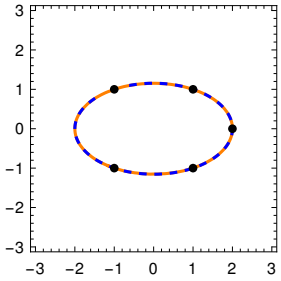
In[ ]:=
```
{g, f} = nrc2[S2]
```

Out[ ]=
$$\left\{1.9649 - 0.491226\ x^2 - 1.47368\ y^2,\ \left\{\frac{1.39279 - 0.792594\ t - 0.135773\ t^2}{0.731804 - 0.226455\ t + 0.135773\ t^2},\ \frac{-0.259672 - 0.566139\ t + 0.135773\ t^2}{0.731804 - 0.226455\ t + 0.135773\ t^2}\right\}\right\}$$

This looks different from the above but

```
Show[ContourPlot [g == 0, {x, -3, 3}, {y, -3, 3}, ContourStyle → Orange],
 ParametricPlot [f, {t, -20, 20}, PlotStyle → Directive [Dashed, Blue]],
 Graphics [{Black, PointSize [Medium], Point[S2]}]]
```

Out[ ]=



The reader may note that the first part of this function is devoted to testing for general position, if this fails then the remainder of the function may fail or create errors, possibly an infinite loop.

The main take-a-way from this discussion is that it puts into focus the meaning of *unique*. Here there is a curve with implicit equation totally independent of the particular 5 points and independent of the parametric function. In this case the 5 points already determine an algebraic curve for $d = 2$, something that does not happen for higher d. In general rational curves of degree d are only a small subset of all curves of degree d, in particular rational curves have genus 0 whereas most non-singular curves of degree greater than 2 have positive genus. And it takes far more than $d + 3$ points to determine a curve. *So in general uniqueness will be more difficult but as in this case it will mean that the algebraic curve produced will be independent of which $d + 3$ points on a RNC are used and independent of the algorithm we develop for finding this curve.*

3. General Position

For higher dimensions checking general positions will be more difficult and time consuming so I will give a stand alone algorithm for checking this. But it will still be important to run that before running our algorithm for finding the RNC.

General position for S means that no $k + 1$ of points in S lie in a $k - 1$ dimensional algebraic set. For example one point gives a 0-dimensional set, two points determine a line, 1 dimensional, 3 points determine a plane-3 dimensional. But when we apply this in d-space we will be considering $d + 3$ points and any such set must lie in at most a d dimensional set. So the requirement on k above is limited to $k \leq d$.

But t it is enough to test the largest subsets, either those of length $d + 1$ or S itself if S has fewer than d+1 points. That is because if our requirement fails for a subset $T \subset S$ then it fails for each subset containing T including S.

Here is a case where it is easier to work projectively rather than affinely since in projective space a linear set with the irrelevant point 0 added is a sub-vector space. The dimension of the linear set is one less than the vector dimension of this space. So we can simply use a matrix rank calculation. Additionally we can now easily allow infinite points.

In my Space Curve GlobalFunctionsMD [4] I have a numerical rank finder matrixrankMD which takes a matrix, that is in Mathematica, a list of vectors of the same size, and a tolerance and calculates the rank to this tolerance via singular values. For this purpose we use a very loose tolerance, such as (0.0003) to eliminate sets that are “almost” not in general position, in other words we are checking *numerical general position.*

Here is our Mathematica code. It is available in our GlobalFunctionsMD.nb [4].

In this function S is a finite list of points either of length d, considered affine, or of length $d + 1$ for infinite points. Note that no more than d infinite points are allowed in a general position set. Unlike some other functions the first element may be affine or projective, this is why the user must enter d. A typical tolerance here is .003

In[ ]:=
```
matrixrank[M_, tol_] := Module[{s, k, l},
    s = SingularValueList[N[M], Tolerance → 0];
    If[s[[1]] < tol, Return[0]]; l = Length[s]; s = s / s[[1]];
    k = 1;
    While[k ≤ l, If[s[[k]] < tol, Return[k - 1], k++]];
    k - 1];

gpTestMD[S_, d_, tol_] := Module[{PS, SS, tst, i, k, n},
  PS = Table[Switch[Length[s], d, Append[s, 1], d + 1, s, _, {}], {s, S}];
  Do[If[Length[s] == 0, Echo["Bad Point"]; Abort[]], {s, PS}];
  tst = True;
  n = Min[d, Length[S] - 1];
  SS = Subsets[PS, {n + 1}];
  k = Length[SS];
  i = 1;
  While[i ≤ k && tst,
   tst = (matrixrank[SS[[i]], tol] == n + 1);
   i++];
  If[tst == False, Echo[SS[[i - 1]], "Test Fails at "];
   Return[False], Return[True]];
 ]
```

Examples: Note combination of affine and projective points.

```
S1 = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0}, {1, 2, 3}, {2, 3, 4, 1}};
gpTestMD[S1, 3, .003]
```

Out[ ]= True

In[ ]:=
```
S2 = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0}, {1, 2, 3}, {2, 3, 4, 0}};
gpTestMD[S2, 3, .003]
```

» Test Fails at {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {2, 3, 4, 0}}

Out[ ]= False

In the second case it fails because there are 4 infinite points for $d = 3$.

Here is an interesting application to RNC. Consider the RNC with polynomial parameterization {t^4, t^3, t^2, t} with d=4.

In[ ]:=
```
S7 = ({t^4, t^3, t^2, t} /. {t → #}) & /@ RandomReal[{-5, 5}, 7]
```

Out[ ]=
```
{{109.44, 33.8363, 10.4614, 3.2344},
 {9.06345, -5.2236, 3.01056, -1.7351}, {6.44303, -4.04406, 2.53831, -1.59321},
 {569.586, -116.592, 23.866, -4.88528}, {84.0456, 27.7579, 9.16764, 3.02781},
 {239.172, 60.8181, 15.4652, 3.93258}, {1.83751, 1.57824, 1.35555, 1.16428}}
```

```
In[ ]:= gpTestMD[S7, 4, dTol]
Out[ ]= True

In[ ]:= S10 = ({t^4, t^3, t^2, t} /. {t → #}) & /@ RandomReal[{-5, 5}, 10];
        gpTestMD[S10, 4, dTol]
Out[ ]= True
```

These are examples of the known fact that *any finite set of points on a RNC is in, theoretically, general position.* Of course if we try

```
In[ ]:= gpTestMD[S7, 4, .0003]
» gpTest Fails at {{109.44 , 33.8363 , 10.4614 , 3.2344 , 1},
    {9.06345 , -5.2236 , 3.01056 , -1.7351 , 1}, {6.44303 , -4.04406 , 2.53831 , -1.59321 , 1},
    {569.586 , -116.592 , 23.866 , -4.88528 , 1}, {84.0456 , 27.7579 , 9.16764 , 3.02781 , 1}}
Out[ ]= False
```

it is not numerically true!

4. Constructing RNC from d+3 points.

In this section I give my algorithm for finding a parametric RNC of degree d from $d+3$ points. This is based on Harris' argument but not completely the same. We could just give the simple code but it does require some explanation.

Harris looks at a special easy case where one can essentially write down the function with almost no computation, in particular there are no equations to solve. He then argues that up to transformation this is typical.

So for the special case we assume that the the $d+3$ points consist of the $d+1$ *coordinate points* in the projective d-plane, that is projective points with all zeros except for one 1. We put the coordinate point with the 1 in the last place last on our list, this is actually the affine origin. We then have two additional points we call p, q.

For $d = 3$ the six points look like

```
{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0}, p, q
```

For this to be in general position p, q must be affine with no zero coordinates. Moreover p, q must be independent as d-vectors. I claim the following rational function, where k is an arbitrary non-zero constant, passes through all these points.

$$F0 = \left\{ \frac{p[\![1]\!]\, t}{k\, \frac{p[\![1]\!]-q[\![1]\!]}{q[\![1]\!]} + t}, \cdots, \frac{p[\![i]\!]\, t}{k\, \frac{p[\![i]\!]-q[\![i]\!]}{q[\![i]\!]} + t}, \cdots, \frac{p[\![d]\!]\, t}{k\, \frac{p[\![d]\!]-q[\![d]\!]}{q[\![d]\!]} + t} \right\}$$

The fact that p, q are not scalar multiples of each other and have no zero coordinates guarantee that the first terms in each denominator, $\tau_j = -k\, \frac{p[\![i]\!]-q[\![i]\!]}{q[\![i]\!]}$ are different. The j^{th} coordinate becomes infinite at τ_j but the other coordinates are non-zero at τ_j. But what this means is that, given the rules for

working with projective points, at τ_j F0 takes the value {0,…,0,1,0,…,0} where the 1 is in the j^{th} place. This was Harris' idea.

Since the constants in the denominator are not zero it is immediate that at $t = 0$ then F0 = 0.

Taking the limit as $t \to \infty$ the constant terms in the denominator become insignificant so the t's cancel giving p[[i]]. Hence Limit$_{t\to\infty}$ F0 = p, so p is in the closure of F0, i.e. in the projective curve determined by F0. This is my idea.

Finally we observe by simple algebra

```
In[ ]:= Clear[p, q, t, q]
        Simplify[p t / (k (p - q) / q + t) /. {t → k}]
```

```
Out[ ]= q
```

So with t = k then F0=q .

It appears that we have one free variable, k, here, but all k does is speed up or slow down the parame - ter so we get the same point set. For theoretical use k can be taken to be 1 but we will find out later that choice of k , which gives a simple change of parameterization, may affect the numerical condition - ing of F0. Along this parameterized curve the point q could be far from the origin so parameter 1 is very constraining.

Here is a simple example for d=3. Let

```
In[ ]:= S = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 0, 1, 0}, {0, 0, 0}, {2, 1, 3}, {-4, -3, 2}};
```

```
In[ ]:= F0 = {2 t / (k (2 + 4) / -4) + t, 1 t / (k (1 + 3) / -3) + t, 3 t / (k (3 - 2) / 2) + t}
```

$$\text{Out[ ]= } \left\{\frac{2\,t}{-\frac{3\,k}{2}+t},\ \frac{t}{-\frac{4\,k}{3}+t},\ \frac{3\,t}{\frac{k}{2}+t}\right\}$$

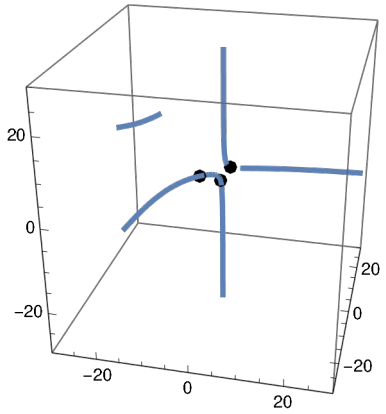
We plot

```
In[ ]:= F1 = F0 /. {k → 1}
```

$$\text{Out[ ]= } \left\{\frac{2\,t}{-\frac{3}{2}+t},\ \frac{t}{-\frac{4}{3}+t},\ \frac{3\,t}{\frac{1}{2}+t}\right\}$$

In[ ]:=
```
Show[ParametricPlot3D [F1, {t, -3, 3}, PlotRange → 30],
 Graphics3D [{Black, PointSize[Large], Point[{{0, 0, 0}, {2, 1, 3}, {-4, -3, 2}}]}]]
```

Out[ ]=



Note that passing through infinite points {1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0} means the asymptotes are parallel to the axes.

Now we consider the general case. Suppose S is a set of $d+3$ points in affine/projective d space, in the Mathematica setting a set of lists of d or $d+1$ real or complex numbers, possibly mixed. Our first step is to invertibly transform to the special case. So we first write them all projectively, that is with $d+1$ components, but now points need to be column vectors rather than rows. So we set

```
SP = Transpose[Table[Switch[Length[s], d, Append[s, 1], d + 1, s, _, {}], {s, S}]]
```

with

```
Asp = Transpose[Take[SP, All, d + 1]]
```

which, because of the general position hypothesis will be an invertible $d+1$ matrix.

The projective linear transformation now is simply matrix multiplication. Clearly, multiplying the first d+1 columns will give by Asp give columns representing the coordinate points and by general position the last 2 columns are affine, that is have non zero bottom term. Since our F0 will be affine we divide the columns by this bottom term and then drop that bottom 1 to get our p, q.

Now we are in the special case so we can simply write down our RNC F0.

Finally we reverse our transformation and recover our desired RNC F using TransformationFunction[As - p][F0].

Here is an example

```
d = 3;
```

```
S = {{-1, -5, -9, 1}, {8, -3, 7, -7}, {1, 1, 7}, {-7, 0, -2, 0}, {-5, 4, -3}, {5, 8, 9, 7}}
```

In[ ]:=
```
gpTestMD[S, 3, .003]
```

Out[ ]=
```
True
```

```
In[ ]:= SP = Transpose[Table[Switch[Length[s], d, Append[s, 1], d + 1, s, _, {}], {s, S}]]
Out[ ]= {{-1, 8, 1, -7, -5, 5}, {-5, -3, 1, 0, 4, 8}, {-9, 7, 7, -2, -3, 9}, {1, -7, 1, 0, 1, 7}}

In[ ]:= Asp = Take[N[SP], All, 4]
Out[ ]= {{-1., 8., 1., -7.}, {-5., -3., 1., 0.}, {-9., 7., 7., -2.}, {1., -7., 1., 0.}}

In[ ]:= S1 = Chop[Transpose[Inverse[Asp].SP]]
Out[ ]= {{1., 0, 0, 0}, {0, 1., 0, 0}, {0, 0, 1., 0},
          {0, 0, 0, 1.}, {-0.757471, -0.386207, -0.945977, 0.245977},
          {-0.842529, -1.01379, 0.745977, -1.64598}}
```

This last step sends SP to our special case form where, because of general position, we see the last components are affine, their last components are non-zero. We prefer them in affine form.

```
In[ ]:= p = Take[S1[[5]] / S1[[5, 4]], 3]
        q = Take[S1[[6]] / S1[[6, 4]], 3]
Out[ ]= {-3.07944, -1.57009, -3.84579}

Out[ ]= {0.511872, 0.615922, -0.453212}
```

Now we can write down our special case rational function, we will use two steps;

```
In[ ]:= τ = Table[-(p[[i]] - q[[i]]) / q[[i]], {i, 3}]
Out[ ]= {7.01604, 3.54918, -7.48564}
```

We note in passing that these are distinct and non-zero. We are using $k = 1$ here. These will be the parameter values of the coordinate points.

```
In[ ]:= F0 = Table[p[[i]] t / (τ[[i]] + t), {i, 3}]
Out[ ]= {-3.07944 t / (-7.01604 + t), -1.57009 t / (-3.54918 + t), -3.84579 t / (7.48564 + t)}
```

Now apply the inverse transformation function.

```
In[ ]:= F = Together[TransformationFunction[Asp][F0]]
Out[ ]= {-(5. (64.1905 - 42.3774 t - 3.36557 t^2 + 1. t^3)) / (-145.415 t + 8.28223 t^2 + 1. t^3),
          (4. (-46.257 + 6.36183 t + 1. t^2)) / (-145.415 + 8.28223 t + 1. t^2),
          -(3. (30.5669 + 59.1233 t - 32.3475 t^2 + 1. t^3)) / (-145.415 t + 8.28223 t^2 + 1. t^3)}
```

We should be able to recover our original points by

```
In[ ]:= F /. {t → τ[[1]]}
Out[ ]= {-1., -5., -9.}

In[ ]:= F /. {t → τ[[2]]}
Out[ ]= {-1.14286, 0.428571, -1.}
```

Here S[[2]] was given projectively, as affine it is

```
In[ ]:= N[Take[S[[2]] / S[[2, 4]], 3]]
Out[ ]= {-1.14286 , 0.428571 , -1.}

In[ ]:= F /. {t → τ[[3]]}
Out[ ]= {1. , 1. , 7.}

In[ ]:= Limit[F, {t → ∞}]
Out[ ]= {-5. , 4. , -3.}

In[ ]:= F /. {t → 1}
Out[ ]= {0.714286 , 1.14286 , 1.28571}
```

Again S[[6]] as affine point

```
In[ ]:= N[Take[S[[6]] / S[[6, 4]], 3]]
Out[ ]= {0.714286 , 1.14286 , 1.28571}
```

Now point 4 was an infinite point. This is unfortunate in the sense that F can not be evaluated directly at that parameter value. In addition infinite points do not have unique representations which gives a further difficulty. We will address this situation at the end of this section.

If we put the steps in the above example together we have a simple procedure which works for general position affine set. It is assumed this set has already been checked for general position, *k* is the parame - ter value for the last point.

```
In[ ]:= rncIshort[S_, k_] := Module[{d, f0, p, q},
          d = Length[S[[1]]];
          A = Transpose[Table[Append[S[[i]], 1], {i, d + 1}]];
          p = TransformationFunction [Inverse[A]][S[[d + 2]]];
          q = TransformationFunction [Inverse[A]][S[[d + 3]]];
          f0 = Table[N[p[[i]] t / (k (p[[i]] - q[[i]]) / q[[i]] + t)], {i, d}];
          Simplify[TransformationFunction [A][f0]]]
```

Here is a nice example:

```
In[ ]:= S = {{-0.661499199020156` , -5.919953297244803` , -6.510244424264634` },
           {0.044324937771346384` , 0.8641488397762416` , 2.9343887135856512` },
           {5.171989530669297` , -4.566836546940028` , -4.195178551692724` },
           {-6.09980134268519` , 0.029250389432121437` , -5.148186154557898` },
           {6.05865662573202` , -7.047040772208991` , -0.35961378453643533` },
           {6.212000809929034` , -2.145705675747042` , 3.439006676843963` }}
Out[ ]= {{-0.661499 , -5.91995 , -6.51024}, {0.0443249 , 0.864149 , 2.93439},
         {5.17199 , -4.56684 , -4.19518}, {-6.0998 , 0.0292504 , -5.14819},
         {6.05866 , -7.04704 , -0.359614}, {6.212 , -2.14571 , 3.43901}}
```

We first note the difference k makes

```
In[ ]:= F1 = rncIshort[S, 3]
```

$$\text{Out[ ]}= \left\{\frac{345.962 - 302.948\,t + 45.6657\,t^2 + 6.05866\,t^3}{-56.7169 + 29.4017\,t - 6.28951\,t^2 + 1.\,t^3}, \frac{1.65899 + 55.8847\,t - 39.5049\,t^2 + 7.04704\,t^3}{56.7169 - 29.4017\,t + 6.28951\,t^2 - 1.\,t^3}, \frac{291.989 - 363.243\,t + 90.4359\,t^2 - 0.359614\,t^3}{-56.7169 + 29.4017\,t - 6.28951\,t^2 + 1.\,t^3}\right\}$$

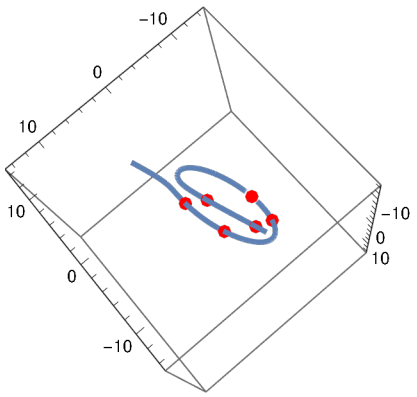
```
In[ ]:= F1 = rncIshort[S, 1]
```

$$\text{Out[ ]}= \left\{\frac{12.8134 - 33.6608\,t + 15.2219\,t^2 + 6.05866\,t^3}{-2.10063 + 3.26685\,t - 2.0965\,t^2 + 1.\,t^3}, \frac{0.0614442 + 6.20941\,t - 13.1683\,t^2 + 7.04704\,t^3}{2.10063 - 3.26685\,t + 2.0965\,t^2 - 1.\,t^3}, \frac{10.8144 - 40.3603\,t + 30.1453\,t^2 - 0.359614\,t^3}{-2.10063 + 3.26685\,t - 2.0965\,t^2 + 1.\,t^3}\right\}$$

This does not give the parameter values for testing but since it is in 3-space we can plot.

```
In[ ]:= Show[ParametricPlot3D[F1, {t, -20, 20}, PlotRange → 15],
  Graphics3D[{Red, PointSize[Large], Point[S]}]]
```

Out[ ]=



We also notice that the answer is not quite given in standard form, that is the denominator for the second coordinate is the negative of the others. With more work we can create a function which can take infinite as well as affine input and gives more complete and desirable output.

4.1 The case of infinite points in the interpolation set.

The algorithm works correctly with up to d infinite points, as long as all points are in general position. Unfortunately it takes a bit of extra work to check that the curve goes through these infinite points. The obvious problem is that this affine function has only d coordinates while infinite points have $d + 1$. We also need to remember that projective points have non-unique representation. Here is the extended algorithm now S can contain affine points of length d, or projective points of length d+1, the latter may be affine or infinite. This function checks general position. Also note that we do use **TransformationFunction** to convert back but rather a special version that forces our standard form with

common denominator.

```
Options[rncInterpolate] = {ProjectiveForm → False};
rncInterpolate[S_, k_, d_, OptionsPattern[]] :=
 Module[{A, F0, p, q, p1, q1, τ, PF0, PF1, F},
  If[Length[S] ≠ d + 3, Echo["S must be of length d+3"]; Abort[]];
  If[gpTestMD[S, d, .0003] ≠ True, Echo["Not General Position"];
   Abort[],
   A = Transpose[Table[Switch[Length[S[[i]]], d,
        Append[N[S[[i]]], 1], d + 1, N[S[[i]]], _, Echo["Bad Data"];
        Abort[]], {i, d + 1}]]];
  p1 = If[Length[S[[d + 2]]] == d, N[S[[d + 2]]], Take[S[[d + 2]] / S[[d + 2, d + 1]], d]];
  p = TransformationFunction[Inverse[A]][p1];
  q1 = If[Length[S[[d + 3]]] == d, N[S[[d + 3]]], Take[N[S[[d + 3]]] / S[[d + 3, d + 1]], d]];
  q = TransformationFunction[Inverse[A]][q1];
  τ = Table[-k (p[[i]] - q[[i]]) / q[[i]], {i, d}];
  F0 = Table[p[[i]] t / (τ[[i]] + t), {i, d}]; (*Special case *)
  denom = Expand[Product[(t - τ[[i]]), {i, d}]];
  num[i_] := nDivideMD[Numerator[F0[[i]]] * denom, (t - τ[[i]]), {t}, 1.*^-10];
  PF0 = Append[Table[num[i], {i, d}], denom];
  PF1 = Expand[A.PF0];
  F = If[OptionValue[ProjectiveForm], PF1, Table[PF1[[i]] / PF1[[d + 1]], {i, d}]];
  {Join[τ, {0, ∞, k}], F}
 ]
```

In addition we have two extra functions which will help us use the option `ProjectiveForm→True`

```
projectiveLimitMD[F_, rule_] := Module[{a},
  a = Limit[Normalize[F], rule];
  If[Abs[a[[-1]]] < .00001, Chop[a], Drop[a / a[[-1]], -1]]]

rncRecoverAffineForm[F_] := With[{n = Length[F]}, Table[F[[i]] / F[[n]], {i, n - 1}]]
```

First we apply this to an affine example using the default option.

```
In[ ]:= S = {{9, 2, 1}, {2, -7, 8}, {-7, -6, 4}, {-6, 5, -2}, {-7, -1, 3}, {5, -9, -8}};
        {τ, F} = rncInterpolate [S, 1, 3]
```

$$\text{Out[ ]= } \left\{\{0.525573,\ 0.451662,\ 0.950439,\ 0,\ \infty,\ 1\},\ \left\{\frac{1.3537 - 8.05295\ t + 15.1509\ t^2 - 8.46866\ t^3}{-0.225617 + 1.32879\ t - 2.31639\ t^2 + 1.20981\ t^3},\right.\right.$$
$$\frac{-1.12808 + 2.67313\ t - 0.304514\ t^2 - 1.20981\ t^3}{-0.225617 + 1.32879\ t - 2.31639\ t^2 + 1.20981\ t^3},$$
$$\left.\left.\frac{0.451233 + 0.601756\ t - 4.65511\ t^2 + 3.62943\ t^3}{-0.225617 + 1.32879\ t - 2.31639\ t^2 + 1.20981\ t^3}\right\}\right\}$$

We can check this using

```
In[ ]:= Table[Limit[F, t → τ[[i]]], {i, 6}]
```

```
Out[ ]= {{9., 2., 1.}, {2., -7., 8.}, {-7., -6., 4.}, {-6., 5., -2.}, {-7., -1., 3.}, {5., -9., -8.}}
```

We have to use the Limit to evaluate because $\tau[[5]] = \infty$., for the others we could have just used regular evaluation, eg

```
In[ ]:= F /. {t → 0}
```

```
Out[ ]= {-6., 5., -2.}
```

When there is an infinite point in the list S or when there is an affine point using projective coordinates, that is d+1 coordinates, then we will need to work projectively to check. Our trick is to make Mathemat - ica find the normalized limit using its sophisticated limit finding algorithm. We need the function given projectively also with d+1 coordinates where the last is the denominator. Here we use the option ProjectiveForm→True. Our example is

```
In[ ]:= S = {{1, 5, 3, 0}, {2, 1, 3}, {5, 2, -3}, {2, 5, 2}, {1, 8, 6, 7}, {8, 7, 9}};
        {τ, H} = rncInterpolate [S, 1, 3, ProjectiveForm → True]
```

$$\text{Out[ ]= } \{\{0.387092,\ 1.03935,\ 0.33945,\ 0,\ \infty,\ 1\},$$
$$\{-0.273136 + 1.02843\ t - 0.819326\ t^2 + 0.11315\ t^3,\ -0.68284 + 2.95463\ t -$$
$$3.13401\ t^2 + 0.905199\ t^3,\ -0.273136 + 1.31735\ t - 1.66786\ t^2 + 0.678899\ t^3,$$
$$-0.136568 + 0.792091\ t - 1.44143\ t^2 + 0.792049\ t^3\}\}$$

Now we check using

```
In[ ]:= chk = Table[projectiveLimitMD [H, t → τ[[i]]], {i, 6}]
```

```
Out[ ]= {{0.169031, 0.845154, 0.507093, 0}, {2., 1., 3.},
         {5., 2., -3.}, {2., 5., 2.}, {0.142857, 1.14286, 0.857143}, {8., 7., 9.}}
```

In the two cases of disagreement it is because the projective form of the input is not unique. For S[[1]] this is projective so the check is returning

```
In[ ]:= Normalize [N[S[[1]]]]
```

```
Out[ ]= {0.169031, 0.845154, 0.507093, 0.}
```

In the case of S[[5]] this is an affine point so the check is returning the standard numerical affine form

```
In[ ]:= Drop[N[{1, 8, 6, 7}/7], -1]
Out[ ]= {0.142857 , 1.14286 , 0.857143}
```

Another use for our ProjectiveLimitMD function is that we can also calculate the infinite points that were not part of our set S.

Here is another example, generated somewhat randomly, that is I did a number of random examples but picked the most visually appealing one. We use a choice of *k* to control the size of the coefficients, we don't want them too large or two small.

```
In[ ]:= S = {{1, 1, 0, 0}, {1.9557125847811783` , -0.4881475780844795` , -2.798797344104713` },
           {0.9869763820862296` , -4.785283892173496` , -5.481948119393831` },
           {-2.1166897584661726` , -3.992574953820508` , 0.4784362247514622` },
           {-6.1540001432381395` , -1.907215841711018` , -6.641385691990251` },
           {-3.4592167609678803` , -0.17710172391338475` , 2.410987119680282` }}
Out[ ]= {{1, 1, 0, 0}, {1.95571 , -0.488148 , -2.7988},
         {0.986976 , -4.78528 , -5.48195}, {-2.11669 , -3.99257 , 0.478436},
         {-6.154 , -1.90722 , -6.64139}, {-3.45922 , -0.177102 , 2.41099}}
```

We first look at the projective form to find our infinite points.

```
In[ ]:= {tau, H2} = rncInterpolate[S, 2, 3, ProjectiveForm → True]
Out[ ]= {{2.49742 , 2.35583 , 2.19614 , 0, ∞, 2}, {27.3498 - 15.0653 t - 2.91933 t^2 + 1.87853 t^3,
           51.5882 - 44.7229 t + 8.30731 t^2 + 0.582183 t^3, -6.18189 + 14.6947 t -
            9.95584 t^2 + 2.0273 t^3, -12.921 + 9.69581 t - 1.04835 t^2 - 0.305253 t^3}}
```

We know the first entry of τ corresponds to the infinite point {1,1,0,0} but the others came from our projective transform from the special case to the specific case. This is recorded in our denominator, that is the last member of H2 which changed in this transform.

```
In[ ]:= sol2 = NSolve[H2[[4]]]
Out[ ]= {{t → -8.0399}, {t → 2.10812}, {t → 2.49742}}
```

So the first two are new while the third was forced by our data. Specifically we will use the following representations of these points

```
In[ ]:= tab = Table[projectiveLimitMD [H2, sol2[[i]]], {i, 3}]
Out[ ]= {{-0.465533 , 0.295661 , -0.834185 , 0},
         {0.361907 , -0.535191 , -0.763279 , 0}, {0.707107 , 0.707107 , 0, 0}}
```

For convenience we give this infinite points names

```
In[ ]:= a = Take[tab[[1]], 3]
        b = Take[tab[[2]], 3]
        c = Take[tab[[3]], 3]
Out[ ]= {-0.465533 , 0.295661 , -0.834185}
Out[ ]= {0.361907 , -0.535191 , -0.763279}
Out[ ]= {0.707107 , 0.707107 , 0}
```

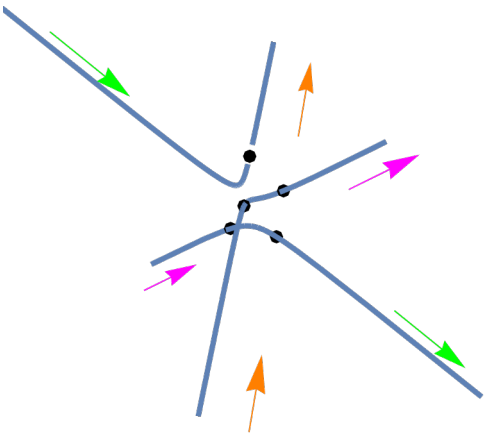
Now we will plot, using arrows to indicate the infinite points which from an affine point of view are now directions. But first we convert our projective H2 to an affine F2

```
In[ ]:= F1 = rncRecoverAffineForm [H2]
```

$$\text{Out[ ]=}\ \left\{\frac{27.3498 - 15.0653\ t - 2.91933\ t^2 + 1.87853\ t^3}{-12.921 + 9.69581\ t - 1.04835\ t^2 - 0.305253\ t^3},\ \frac{51.5882 - 44.7229\ t + 8.30731\ t^2 + 0.582183\ t^3}{-12.921 + 9.69581\ t - 1.04835\ t^2 - 0.305253\ t^3},\ \frac{-6.18189 + 14.6947\ t - 9.95584\ t^2 + 2.0273\ t^3}{-12.921 + 9.69581\ t - 1.04835\ t^2 - 0.305253\ t^3}\right\}$$

```
In[ ]:= Show[ParametricPlot3D [F1, {t, -40, 40}, PlotRange → 25],
          Graphics3D [{{Black , PointSize [Large], Point[Drop[S, 1]]}, {Orange , Arrow[{-30 a, -20 a}],
              Arrow[{15 a, 30 a}]}, {Magenta , Arrow[{-10 b, -20 b}], Arrow[{25 b, 15 b}]},
            {Green , Arrow[{15 c, 25 c}], Arrow[{-30 c, -20 c}]}}], Boxed → False , Axes → False]
```

Out[ ]=



These arrows also give instructions on traversing this simple closed curve in projective 3-space. This curve is typical of degree 3 RNC as we expect 3 distinct infinite points in general.

5. Constructing and Implicitizing parametric rational normal curves.

I review some of the material in [2], [4].

It is well known, eg. [1], that every rational curve is a projective linear transform of the rational normal curve $\{t^n, t^{n-1}, \ldots, t\}$. In particular each transformation matrix of size $(n+1)\times(d+1)$ defines a rational curve in n variables of degree no larger than d. But to get all rational normal curves of dimension and degree d we use transformation matrices which are invertible square $d+1$ matrices.

Example : Let $d = 3$, then construct a matrix, say

```
In[ ]:= Mf = RandomReal[{-9, 9}, {4, 4}];
        Mf // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1.81843 & 1.67439 & -1.86985 & 1.73284 \\ 6.04049 & 8.52234 & 7.34568 & -1.16999 \\ 4.27965 & -8.82913 & 2.92833 & -2.36913 \\ 4.09386 & -8.05956 & 4.40576 & 8.93662 \end{pmatrix}$$

We test its rank using a loose tolerance to avoid near-singular matrices

```
In[ ]:= matrixrank[Mf, .0003]
```

Out[ ]= 4

This is OK so our rational normal curve will be

```
In[ ]:= f = TransformationFunction[Mf][{t^3, t^2, t}]
```

Out[ ]=

$$\left\{\frac{1.73284 - 1.86985\, t + 1.67439\, t^2 + 1.81843\, t^3}{8.93662 + 4.40576\, t - 8.05956\, t^2 + 4.09386\, t^3},\ \frac{-1.16999 + 7.34568\, t + 8.52234\, t^2 + 6.04049\, t^3}{8.93662 + 4.40576\, t - 8.05956\, t^2 + 4.09386\, t^3},\ \frac{-2.36913 + 2.92833\, t - 8.82913\, t^2 + 4.27965\, t^3}{8.93662 + 4.40576\, t - 8.05956\, t^2 + 4.09386\, t^3}\right\}$$

Now if we want to find an implicit system of equations for this curve we can use the function FLTMD in my Space Curve Global Functions [4] and a basis for the RNC which, while described in [1] is calculated in [4]. This will contain $\binom{d}{2}$ such polynomials. For $d = 3$ I give this as

```
In[ ]:= tBasis3
```

Out[ ]= $\{x2^2 - x1\, x3,\ x1\, x2 - x3,\ x1^2 - x2\}$

So the implicit equation is

```
In[ ]:= B = FLTMD[tBasis3, Mf, 3, {x3, x2, x1}, {x, y, z}, dTol]
```

» Initial Hilbert Function {1, 4, 7, 10}

» Final Hilbert Function {1, 4, 7, 10}

Out[ ]= $\{-1.62609\, x + 12.988\, x^2 - 0.420577\, y + 1.26435\, x\, y - 0.595578\, y^2 + 3.46883\, x\, z - 2.24009\, y\, z + 1.\, z^2,\ -0.579616\, x + 13.5337\, x^2 - 0.132874\, y + 1.05077\, x\, y - 0.669521\, y^2 + 1.\, z + 0.881133\, x\, z - 1.88158\, y\, z,\ 1. - 8.54915\, x + 26.6566\, x^2 + 0.200252\, y - 7.47293\, x\, y + 0.689995\, y^2 + 8.63689\, x\, z - 2.18636\, y\, z\}$

In[ ]:=
```
Length[B]
```

Out[ ]= 3

Note that we do get 3 equations.

Working numerically it is always a good idea to check results. We evaluate B at f

In[ ]:=
```
fB = Simplify[B /. Thread[{x, y, z} → f]]
```

Out[ ]= $\{(1.77636 \times 10^{-15} + 1.05471 \times 10^{-14}\, t + 4.04121 \times 10^{-14}\, t^2 - 2.84217 \times 10^{-14}\, t^3 + 2.22045 \times 10^{-16}\, t^4 - 7.54952 \times 10^{-14}\, t^5 + 2.84217 \times 10^{-14}\, t^6) / (2.18293 + 1.07619\, t - 1.96869\, t^2 + 1.\, t^3)^2,\ (4.66294 \times 10^{-15} + 7.66054 \times 10^{-15}\, t + 5.01821 \times 10^{-14}\, t^2 - 2.84217 \times 10^{-14}\, t^3 - 2.13163 \times 10^{-14}\, t^4 - 8.03801 \times 10^{-14}\, t^5 + 2.17604 \times 10^{-14}\, t^6) / (2.18293 + 1.07619\, t - 1.96869\, t^2 + 1.\, t^3)^2,\ (-4.4853 \times 10^{-14} + 5.86198 \times 10^{-14}\, t + 4.21885 \times 10^{-14}\, t^2 - 7.10543 \times 10^{-14}\, t^3 + 3.73035 \times 10^{-14}\, t^4 - 1.63425 \times 10^{-13}\, t^5 + 1.02141 \times 10^{-13}\, t^6) / (2.18293 + 1.07619\, t - 1.96869\, t^2 + 1.\, t^3)^2\}$

In[ ]:=
```
Chop[fB, 1.*^-12]
```

Out[ ]= {0, 0, 0}

We may also want to check that B defines a curve. An easy, but not always accurate, way is to use my function tangentVectorJMD from [4]. If a random point on f has a tangent vector then it likely is a curve. One may want to check several times if one is not comfortable with the statistical significance of one point samples.

```
tangentVectorJMD[F_, p_, X_] := Module[{J, ns},
  J = D[F, {X}] /. Thread[X → p];
  ns = NullSpace[J];
  If[Length[ns] == 1, Return[ns[[1]]], Echo[p, "no unique tangent vector at"]];
  Table[0, {Length[X]}]]
```

```
r = RandomReal[{-5, 5}]
p = f /. {t → r}
tangentVectorJMD[B, p, {x, y, z}]
```

Out[ ]= 3.68233

Out[ ]= {0.90075, 3.68328, 0.851149}

Out[ ]= {-0.164034, -0.971042, 0.173695}

If the results are troublesome one may use **tangentVectorMD** from my **GlobalFunctionsMD .nb** in [4] which works better but requires subroutines.

6. Uniqueness

Unlike the case $d = 2$ a set S of $d + 3$ points does not determine a unique curve. One might want to get an handle on the fact that of all curves in d-space through S only one is an RNC. Instead I formulate uniqueness as follows:

Start with a rational normal curve with both parametric equation F and implicit equation $\mathbb{C}$. *Take a random collection S of* $d + 3$ *numerically general position points from F, i.e. pick* $d + 3$ *random parameter values. Apply our algorithm* **rncInterpolate** *to get a RNC parameterized G curve through S and check that it lies in* $\mathbb{C}$.

The idea is that if there are several, more likely many, RNC curves through a given set of $d + 3$ points our algorithm, which takes only the points as data, would not have enough information to pick the original curve $\mathbb{C}$.

As our first example we use the curve f, B in the previous section.

```
In[ ]:= S = (f /. {t → #}) & /@ RandomReal[{-3, 3}, 6]
```

```
Out[ ]= {{-0.426214, 0.759864, 2.33167}, {0.0855332, 0.505055, 1.14394},
  {0.0158313, 0.45386, 1.22803}, {0.506805, 3.0605, -0.432657},
  {1.08119, 5.06439, 0.40951}, {0.536624, 3.21782, -0.427656}}
```

```
In[ ]:= {tau, G} = rncInterpolate[S, .1, 3]
```

$$\text{Out[ ]= } \left\{\{-2.20539, -2.54293, -2.44453, 0, \infty, 0.1\}, \left\{\frac{6.94792 + 10.9696\,t + 6.05548\,t^2 + 1.11889\,t^3}{13.7093 + 13.418\,t + 5.44826\,t^2 + 1.03487\,t^3}, \frac{41.9572 + 63.3212\,t + 31.5682\,t^2 + 5.24096\,t^3}{13.7093 + 13.418\,t + 5.44826\,t^2 + 1.03487\,t^3}, \frac{-5.93141 - 5.24852\,t - 0.457859\,t^2 + 0.423788\,t^3}{13.7093 + 13.418\,t + 5.44826\,t^2 + 1.03487\,t^3}\right\}\right\}$$

```
In[ ]:= simp = Simplify[B /. Thread[{x, y, z} → G]]
```

$$\text{Out[ ]= } \left\{\left(-2.50111 \times 10^{-12} + 3.87672 \times 10^{-11}\,t + 8.81073 \times 10^{-11}\,t^2 + 7.25322 \times 10^{-11}\,t^3 + 2.83933 \times 10^{-11}\,t^4 + 5.31486 \times 10^{-12}\,t^5 + 4.03677 \times 10^{-13}\,t^6\right) \Big/ \left(13.2474 + 12.966\,t + 5.2647\,t^2 + 1.\,t^3\right)^2,\right.$$
$$\left(-4.49063 \times 10^{-12} + 1.50635 \times 10^{-11}\,t + 4.95106 \times 10^{-11}\,t^2 + 4.57305 \times 10^{-11}\,t^3 + 1.99378 \times 10^{-11}\,t^4 + 4.04654 \times 10^{-12}\,t^5 + 3.51164 \times 10^{-13}\,t^6\right) \Big/ \left(13.2474 + 12.966\,t + 5.2647\,t^2 + 1.\,t^3\right)^2,$$
$$\left.\left(-1.7053 \times 10^{-12} + 2.29647 \times 10^{-11}\,t + 5.8435 \times 10^{-11}\,t^2 + 5.17275 \times 10^{-11}\,t^3 + 2.09752 \times 10^{-11}\,t^4 + 4.02167 \times 10^{-12}\,t^5 + 3.28626 \times 10^{-13}\,t^6\right) \Big/ \left(13.2474 + 12.966\,t + 5.2647\,t^2 + 1.\,t^3\right)^2\right\}$$

```
In[ ]:= Chop[simp, 1.*^-10]
```

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

Again, if Mathematica's random numbers are not random enough for you, run this again several or many times, using different d, to be convinced.

References:

1. Joe Harris, A*lgebraic Geometry, a first course* , Graduate Texts in Mathematics, Springer, 1992.
2. Barry H Dayton, *Degree vs. Dimension for Rational Parametric Curves,* Mathematica-Journal 22, 2020.
3. Barry H Dayton, *A Numerical Approach to Real Algebraic Plane Curves,* Wolfram Media, 2018. Updates and code at **https://barryhdayton.space**

4. Barry H Dayton, *Space Curve Book,* also code, **http://barryhdayton.space**