Curvature Continuous Piecewise Pythagorean Hodograph Curves

Technical Report CS-2026-01, School of Computer Science, University of Waterloo

Stephen Mann

Abstract

This report develops constructions for cubic, quartic, and quintic Pythagorean hodograph curves (PHC). The cubic construction is a rederivation of an earlier cubic PHC construction, but in the real plane rather than the imaginary plane. The quartic construction is also a rederivation of an earlier construction, again in the real plane rather than the imaginary plane. In addition, the free parameter in this quartic construction is used to interpolate curvature at one end of the curve. The quintic construction builds a quintic PHC curve that interpolates the position, tangent, and curvature at two points.

1 Introduction

Pythagorean-hodograph curves [Farouki] are a type of parametric, planar polynomial curve whose "parametric speed" is a polynomial. Farouki [Farouki] notes that these curves have several nice features:

> For example, it is possible to compute their arc lengths, bending energies, and offset (parallel) curves in an essentially exact manner, without recourse to approximations; and they are exceptionally well–suited to problems of real–time motion control and spatial path planning based on the use of rotation–minimizing frames.

Our interest here will be in testing if a cubic Bézier curve is a Pythagorean hodograph curve; interactively constructing a parametric, C^1, piecewise polynomial curve, where each piece is a cubic Pythagorean-hodograph curve; and constructing a piecewise, cubic geometric Hermite Pythagorean-hodograph curve to interpolate a set of points and directions. Additionally, we will do similar constructions for quartic Pythagorean hodograph curves, where an extra degree of freedom in the construction is used to interpolate curvature at one end of the curve. Finally, I will develop a quintic PHC that interpolates the position, tangent and curvature at two points.

There is previous work on constructing cubic, quartic, and quintic PHCs. Meek and Walton Meek and Walton [MeekWalton97] showed how to construct a cubic geometric Hermite Pythagorean hodograph curve. My construction (Section 2.2) differs from theirs in that they were working in the imaginary plane while my construction is in the real plane, although my result is essentially reproducing their work.

Similarly, Wang and Fang [WangFang09] gave conditions on the control points of a quartic Bézier curve to be a PH curve (see below). They also gave a construction for a G^1 Hermite interpolant using quartic PH curves. Again, my work (Section 3.1) is essentially reproducing their work. However, this quartic construction has a degree of freedom in it, and I show how to use this degree of freedom to have the Hermite interpolant interpolate curvature at one end of the curve.

My work on quintic PHC curves is based on the work of Hormann et al. [HRV24]. In their paper, they give several formulations of conditions on the control points of a quintic Bézier curve for the curve to be a PHC. One set of these conditions uses similar triangles; in Section 4, I use these conditions to find a set of equations that must hold for a curve to interpolate the position, tangent, and curvature at two points and for the curve to also be a PHC.

In this paper, I work in PGA (Point-Based Geometric Algebra) [DdK, Gunn11]. While this algebra is convenient for doing rotations, all the results hold in a standard affine space. For more on PGA, see [DdK, LM25].

1.1 Bézier curves

A degree n Bézier curve [Farin91] is a parametric polynomial curve with **control points** P_i blended by the Bernstein polynomials $B_i^n(t) = \binom{n}{i}(1-t)^{n-1}t^i$:

$$B(t) = \sum_{i=0}^{n} P_i B_i^n(t),$$

with a parameterization of $[0,1]$. Note that $1 = ((1-t)+t)^n = \sum_{i=0}^n \binom{n}{i}(1-t)^{(n-i)}t^i = \sum_{i=0}^n B_i^n(t)$, so a Bézier curve is an affine combination of its control points. Further, for $t \in [0,1]$, $B_i^n(t) \geq 0$, and the curve is a convex combination of its control points.

The derivative of a Bézier curve $B(t)$ at its end points is computed as the scaled difference of the first (last) control points:

$$B'(0) = n(P_1 - P_0), \quad B'(1) = n(P_n - P_{n-1}).$$

The second derivatives at the ends of the curve are

$$B''(0) = n(n-1)\big((P_2-P_1)-(P_1-P_0)\big), \quad B''(1) = n(n-1)\big((P_n-P_{n-1})-(P_{n-1}-P_{n-2})\big).$$

For more details on Bézier curves, see, for example, Farin's textbook [Farin91].

2 Cubic Pythagorean-Hodograph Curves

Mathematically, a curve B is a Pythagorean-hodograph (PH) curve if $||B'(t)|| = \sigma(t)$ for some polynomial $\sigma(t)$. All linear curves are Pythagorean-hodograph curves, but no quadratic curves are Pythagorean-hodograph curves (other than those with colinear control points). Our interest is in cubic Pythagorean-hodograph curves in Bézier form, an example of which is shown in Figure 1, and later in quartic and quintic PH curves. For a cubic Bézier curve $B(t) = \sum_{i=3}^{e} P_i B_i^3(t)$ to be a Pythagorean-hodograph curve, the interior angles of the control polygon must be equal (the α's in Figure 1) and the edges lengths $\ell_i = ||P_i - P_{i-1}||$ must be in a geometric progression; e.g., $\ell_2/\ell_1 = \ell_3/\ell_2$ in this figure. An equivalent condition to this second condition, that $\ell_l\ell_3 = \ell_2^2$, will be

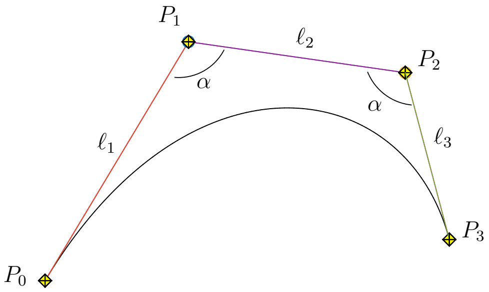


Figure 1: A cubic Pythagorean-hodograph curve. Edge lengths ℓ_1, ℓ_2, and ℓ_3 are in a geometric progression.

used in Section 2.2.

2.1 Creating a Piecewise, C^1 Cubic Pythagorean Hodograph Curve

I first present an interactive construction to create a piecewise, C^1, cubic PH curve. The process is illustrated in Figure 2. As a first step, the user selects three points on the screen (Figure 2(a)). From these points, we construct two lines, one through P_0, P_1 and one through P_1, P_2 using `join` (Figure 2(b)), and compute the magnitudes of these lines (the magnitude of the join of two points is the distance between the two points):

```
>> ln1 = join(P0,P1); l1 = norm(ln1);
>> ln2 = join(P1,P2); l2 = norm(ln2);
```

We then compute the rotor that rotates line `ln1` around the intersection of `ln1` and `ln2` by twice the angle between the two lines:

```
>> R = ln2/ln1/l1/l2;
```

Note that we needed to normalize the lines to get the rotation we want. We then apply this rotor to `l1`:

```
>> ln3 =  ln3 = R*ln1*inverse(R);
```

This gives us the green line in Figure 2(c).

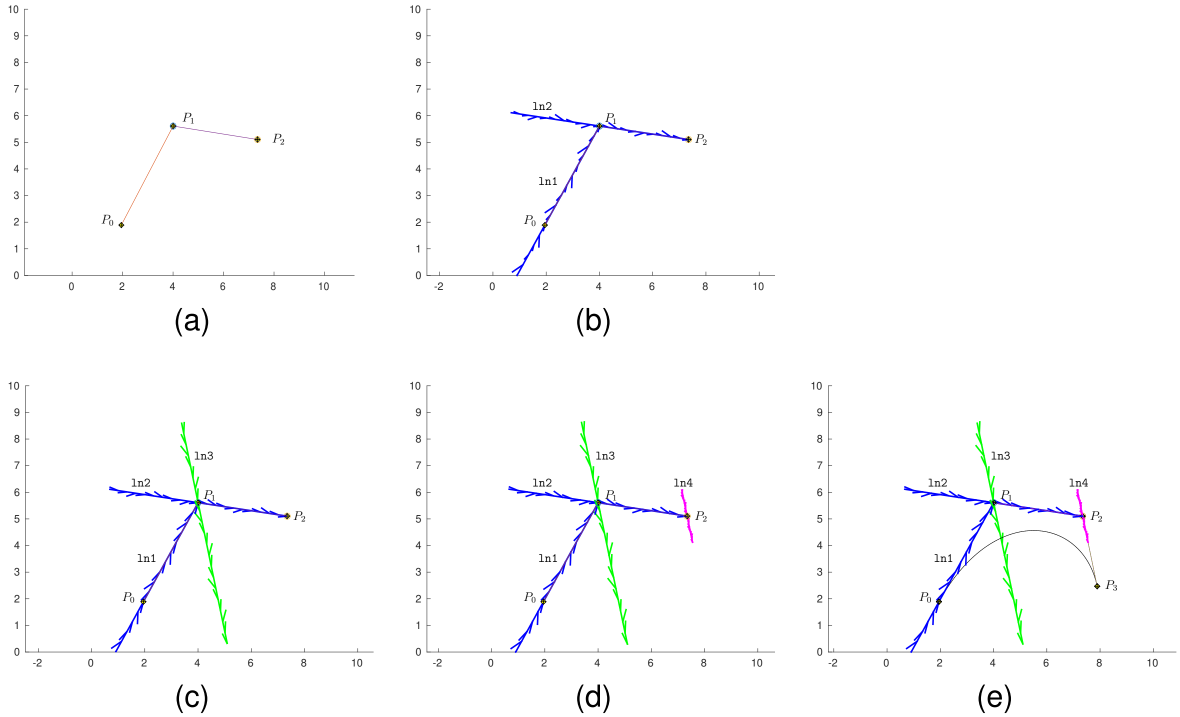


Figure 2: Creating a cubic Pythagorean-hodograph curve. (a) Three control points selected by the user. (b) Lines `ln1=join(P0,P1)` and `ln2=join(P1,P2)`. (c) Line `ln3` created as the rotation of line `ln1`. (d) Line `ln4` created as the translation of line `ln3`. (e) Placement of control point `P4` to create a Pythagorean-hodograph curve.

By chance, for this problem, the rotor rotating by twice the angle between the two lines was almost exactly what we wanted to construct the third leg of the Bézier control polygon. However, we want the line parallel to `ln3` that passes through `P2`. Thus, we construct the translation from `P1` to `P2`, and apply it to `ln3` to get `ln4`, which we normalize (Figure 2(d)):

```
>> T = gexp(-e0*hdual(P1-P2)/2);
>> ln4 = T*ln3*inverse(T);  ln4 = ln4/norm(ln4);
```

By construction, the angle between `ln1` and `ln2` is equal to the angle between `ln2` and `ln4`. Thus, our fourth Bézier control point needs to lie on `ln4`. Since we want the distances between the control points to be in geometric progression, we set `P3` to

```
>>  P3=P2-(l2^2/l1)*(e0*ln4);
```

(see equation (19) of PGA4CS [DdK] for details on this equation). The final curve appears in Figure 2(e).

The user can add additional PH segments, with each new segment meeting the previous segment with C^1 continuity. Requiring each segment to meet the previous segment with C^1 continuity sets the first two control points of the next segment. The user then interactively sets the third control point, and the program computes the fourth control point of the new segment to ensure that the segment is a PH curve in the same manner as above. One caution: the geometric progression requirement essentially gives exponential growth (up or down); placing additional points too close or too far from existing points will likely give unwanted results.

2.2 G^1 Piecewise Cubic Pythagorean Hodograph Curves

Meek and Walton [MeekWalton97] showed how to construct a cubic geometric Hermite Pythagorean hodograph curve segment, that is, given two points P_1 and P_2 and two vector $\vec{v}_0$ and $\vec{v}_1$, they show how to construct a cubic Pythagorean hodgraph curve C such that $C(0) = P_1$, $C(1) = P_2$ and $C'(0) = \alpha_1\vec{v}_1$ and $C'(1) = \alpha_2\vec{v}_2$ for some scalars $\alpha_1, \alpha_2 > 0$, where we can assume $|\vec{v}_1| = |\vec{v}_2| = 1$. Their approach expresses the Pythagorean hodograph curve as a Bézier curve in complex number notation, and constructs the interior control points by rotating two vectors around the end control points and solving a quadratic system to find the solution.

The approach taken here is somewhat different. Since we know that the two interior angles of the Bézier control polygon must be equal, we construct the two lines, `ln1` and `ln2`, from the given positions and directions, and we construct the rotor that rotates from `ln1` to `ln2` (around the intersection point `ip` of the two lines) but by half the angle between the two lines. We apply this rotor to `ln1` to get a line `ln3` that is the "halfway" line between `ln1` and `ln2`. By constructing the interior edge of the Bézier control polygon to be parallel to `ln3`, the Bézier curve will meet the equal angle constraint. We then derive a quadratic equation to translate the interior edge of the Bézier control polygon (`ln3`) so that the edge length ratio constraint of a Pythagorean hodograph curve is met. This process is illustrated in Figure 3 and detailed in the rest of this section.

We wish to construct Bézier control points `cp(1),...,cp(4)` so that the curve interpolate the points `P1` and `P2` at its endpoints, and is tangent to `v1`,`v2` at these endpoints (see Figure 3(a)). The end control points are given by the data: `cp(1)=P1` and `cp(4)=P2`. To find control points `cp(2),cp(3)` , we construct the lines `ln1` and `ln2` through the given data (Figure 3(b)):

```
>> ln1 = galine(v1,p1); ln1 = ln1/norm(ln1);
>> ln2 = galine(v2,p2); ln2 = ln2/norm(ln2);
```

We now construct the rotor `R=ln2/ln1` (which rotates by twice the angle between the lines), and then construct the rotor `Ra` that rotates by half the angle between the two lines, and apply this second rotor to `ln1`, to get `ln3` (which will be parallel to the interior leg of the Bézier control polygon, Figure 3(c)):

```
>> %...
>> R = ln2/ln1;
>> Ra = gexp(glog(R)/4);
>> ln3 = Ra*ln1*inverse(Ra);
>> ip = ip = e3^glog(R); ip = ip/norm(ip);
```

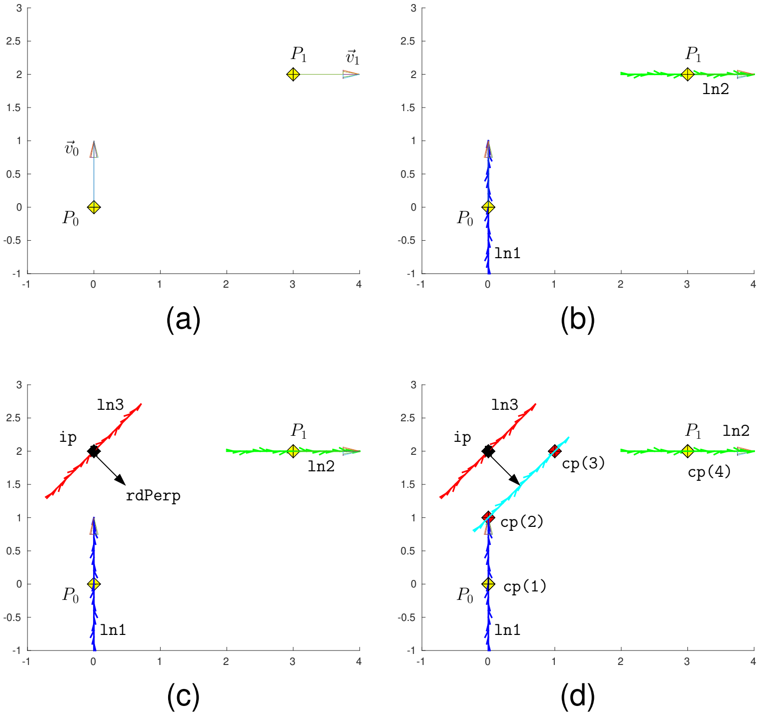


Figure 3: Finding the interior points on a Pythagorean hodograph curve that interpolates two positions and tangent directions. (a) The data. (b) Construct lines through the data. (c) Construct a line `ln3` that is half the rotation from `ln1` to `ln2` passing through the intersection point of the two lines. (d) The goal is to find an amount to translate `ln3` so that the intersection of this translated line with `ln1` and with `ln2` gives us the desired control points for a PH curve.

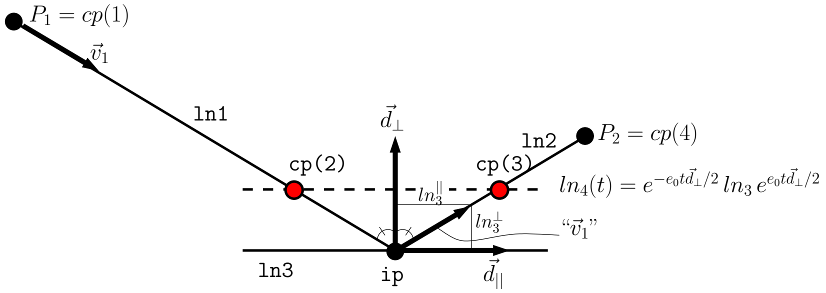


Figure 4: Setting up the equations in t for translating `ln3` to find locations of `cp(2)` and `cp(3)` for a Pythagorean hodograph curve.. We have drawn $\vec{v}_1$ mirrored along both axes to better indicate $ln_3^{||}$ and $\mathrm{ln}_3^{\perp}$ in the figure.

We now wish to translate `ln3` in the direction perpendicular to `ln3` until the intersection of the translated `ln3` with `ln1` and with `ln2` are in the location to give the appropriate ratios for the control polygon segments to be in the correct location for a cubic Pythagorean hodograph curve. The remaining steps are more algebraic, so we shift to a more mathematical notation; hopefully the correspondence to the Matlab variables is clear. See Figure 4 for an illustration of the quantities involved.

To find the correct amount of translation of `ln3` we will use the condition that $\ell_1\ell_3 = \ell_2^2$. We set up a local coordinate system relative to the direction of `ln3` ($\vec{d}_{||}$) and perpendicular to the direction of `ln3` ($\vec{d}_\perp$), and express the slope of `ln1` (which has direction $\vec{v}_1$) relative to this coordinate system: $s = \frac{ln_3^\perp}{ln_3^{||}}$ where $ln_3^\perp = \vec{v}_0 \cdot \vec{d}_\perp$ and $ln_3^{||} = \vec{v}_0 \cdot d^{||}$.

We will translate `ln3` by a distance t in the direction $d_\perp$,

$$ln_4(t) = e^{-e_0\, t\, r^\perp/2}\, ln_3\, e^{e_0\, t\, r^\perp/2},$$

and intersect `ln4` with `ln1` and `ln2` to get `cp(2)` and `cp(3)`. Let $\ell_1 = |cp(1) - cp(2)|$, $\ell_2 = |cp(2) - cp(3)|$, and $\ell_3 = |cp(3) - cp(4)|$, and let $l_1 = |ip - cp(1)|$ and $l_2 = |ip - cp(4)|$.

As we vary t, ℓ_1, ℓ_2, and ℓ_3 will vary:

$$\begin{aligned}
\ell_1 &= l_1 - \frac{|v_1|}{ln_3^{\perp}}t = l_1 - \frac{t}{ln_3^{\perp}} \\
\ell_2 &= 2t/s \\
\ell_3 &= l_2 - \frac{|v_1|}{ln_3^{\perp}}t = l_2 - \frac{t}{ln_3^{\perp}}
\end{aligned}$$

since $|\vec{v_1}| = 1$. Expanding the ratio condition $\ell_1\ell_3 = \ell_2^2$ gives us

$$(\ell_1 - \frac{t}{ln_3^{\perp}})(\ell_2 - \frac{t}{ln_3^{\perp}}) = \left(\frac{2}{s}t\right)^2 \quad \Rightarrow \quad \ell_1\ell_2 - (\ell_1 + \ell_2)\frac{t}{ln_3^{\perp}} + \left(\frac{1}{(ln_3^{\perp})^2} - \frac{4}{s^2}\right)t^2 = 0.$$

There are two solutions to this equation; one solution is typically preferred over the other (for example, if one solution is positive and the other negative, then the negative solution leads to a curve with a loop while the positive solution gives a simple arc; we would generally prefer the positive solution in this case).

We then use the chosen t value to construct a translation rotor to apply to `ln3`,

```
>> %...
>> LT = gexp(-e0*(-rts(2))*rdPerp/2)
>> ln4 = LT*ln3*inverse(LT)
```

where `rdPerp` is the Euclidean vector perpendicular to `ln3`, and we then intersect `ln4` with `ln1` and with `ln2` to get the interior control points of our interpolatory PH curve (Figure 3(d)).

Figure 5 shows an example that fits four cubic Hermite PH curve segments to five points and tangent directions. Figure 5 (e) shows the resulting curve if you switch which root is used to construct each curve segment. Note the loops and note that last segment meets the previous segment only C^0 since the last curve segment's end point derivatives are reversed.

2.3 Discussion of cubic Pythagorean Hodograph Constructions

In Section 2.1, we constructed a piecewise, cubic PH curve where the pieces met with C^1 continuity, while in Section 2.2, we constructed a piecewise, cubic PH curve

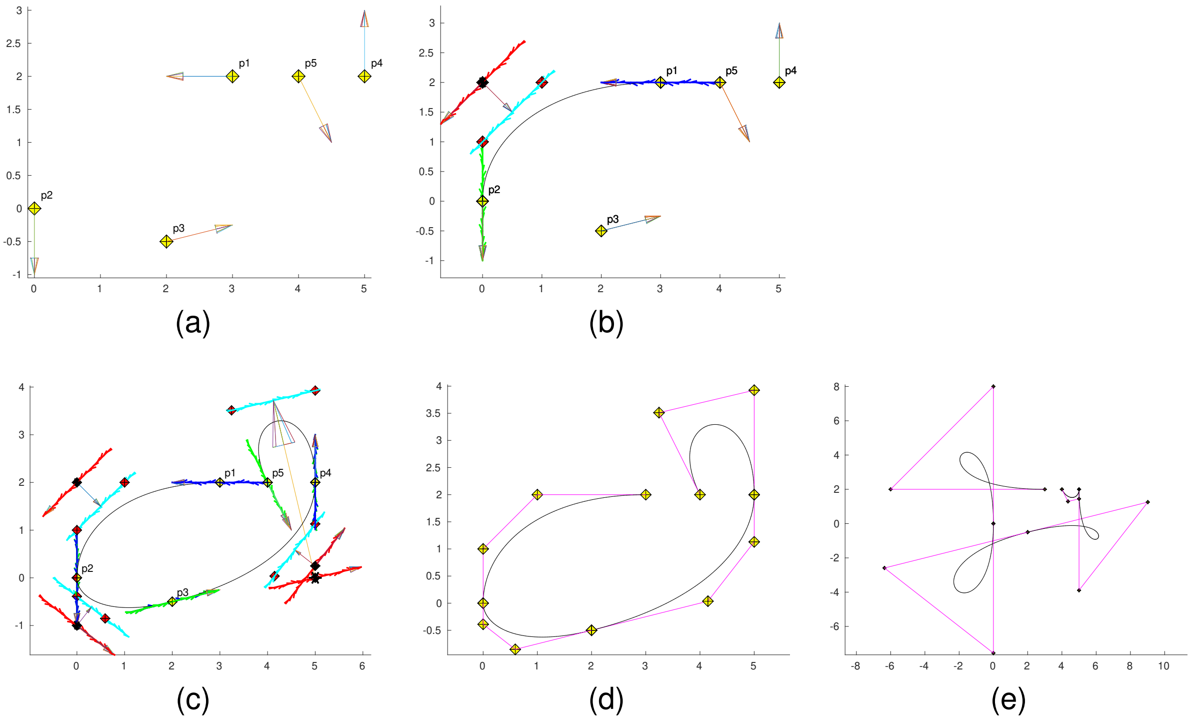


Figure 5: Constructing a cubic Hermite PH curve. (a) The data. (b) Construction of the first segment. (c) Construction of all four segments. (d) Just the curve. (e) Same data but choosing the other solution for each curve segment.

where the pieces met with G^1 continuity. The question is: why the weaker form of continuity (G^1) in the second construction? The answer has to do with the data being interpolated and how it is processed.

In the first construction, the user sets the first three Bézier control points of a curve, and the program sets the final control point to make the curve a PH curve. If a second segment is desired, the **program** sets the first two control points of the second segment, ensuring C^1 continuity between the first and second segments. The user then sets the third control point, and the program again sets the fourth control point to ensure that the second Bézier curve segment is a PH curve.

In the second construction, the user gives a sequence of positions and tangent lines to interpolate. The program then constructs a cubic PH curve between adjacent points/tangents. In constructing each individual segment, there is only enough freedom to achieve G^1 continuity (i.e., we can only interpolate tangent direction and not the exact derivative vector).

Note that from a user interface stand point, the first construction is a poor way to model a curve. I.e., the selection of the third control point (and only the third control point) is a fairly poor way to specify the curve you want.

3 C^1 Quartic Pythagorean Hodograph Curve

Wang and Fang [WangFang09] gave conditions on the control points of a quartic Bézier curve to be a PH curve (see below). They also gave a construction for a G^1 Hermite interpolant using quartic PH curves. In this section, I give an alternate construction for C^1 Hermite interpolation using quartic PH curves.

Figure 6 illustrates Wang and Fang's conditions for a quartic Bézier curve to be a Pythagorean hodograph curve. Starting with control points `cp((1:5))`, we find the line `s1,s2` that passes through `cp(3)` and forms the same angle with the line `cp(1),s1` as it forms with the line `s2,cp(5)`, and where `s1` lies on the line `cp(1),cp(2)` and `s2` lies on the line `cp(4),cp(5)`. We then compute the control polygon edge lengths $E_i = |$`cp(i+1)-cp(i)`$|$, as well as the lengths $F_0 = |$`s1-cp(2)`$|$, $F_1 = |$`cp(3)-s1`$|$, $F_2 = |$`s2-cp(3)`$|$, and $F_3 = |$`cp(5)-s2`$|$. The curve is a Pythagorean hodograph curve

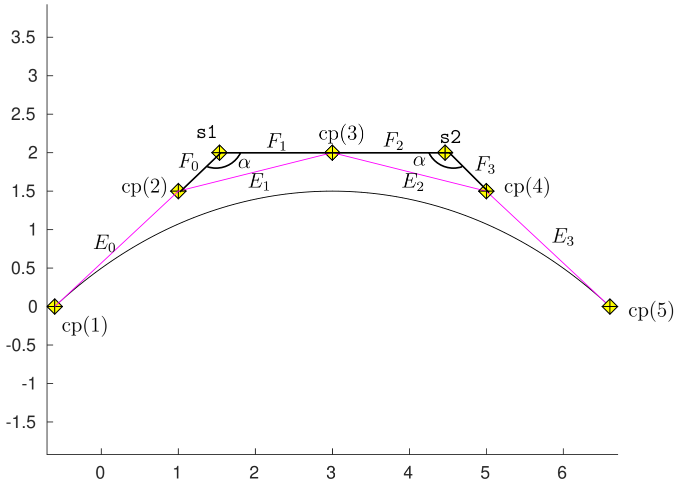


Figure 6: A quartic PH curve.

if

$$
\begin{aligned}
E_0 F_2 &= 3F_0 F_1 \\
E_3 F_1 &= 3F_2 F_3 \qquad (1) \\
F_1 F_2 &= 4F_0 F_3.
\end{aligned}
$$

3.1 Quartic G^1 Pythagorean Hodograph Curves with $F_1 = F_2$

The problem with (1) is that these equations do not incorporate the geometry of Figure 6, since (1) only constrains ratios of distances and not distances. Thus, a solution to these equations may not lead to a PH curve since the geometry constraints aren't satisfied. We add three geometry constraints, which are variations of the geometric conditions we saw for cubics:

$$
\begin{aligned}
E_0 + F_0 &= \ell_1 - t/hwl_\perp \\
E_3 + F_3 &= \ell_2 - t/hwl_\perp \\
F_1 + F_2 &= 2t\frac{hwl_\parallel}{hwl_\perp}.
\end{aligned}
$$

The first two constraints make $E_0 + F_0$ and $E_3 + F_3$ into distances, while the third constraint make $F_1 + F_2$ into a distance.

At this point, we have six equations and seven unknowns $(E_0, E_1, F_0, F_1, F_2, F_3, t)$, i.e., the problem is under-constrained. For now, we arbitrarily add the restriction that

$$F_1 = F_2,$$

which places `cp(3)` at the midpoint of `s1,s2` (we will discuss this constraint further, and look at another alternative in Section 3.2).

The assumption that $F_{12} \equiv F_1 = F_2$ simplifies our equations to

$$\begin{aligned} E_0 &= 3F_0 && (2)\\ E_3 &= 3F_3 && (3)\\ F_{12} &= t\frac{hwl_{\|}}{hwl_{\perp}} && (4)\\ F_{12}^2 &= 4F_0F_3 && (5)\\ 4F_0 &= \ell_1 - t/hwl_{\perp} && (6)\\ 4F_3 &= \ell_3 - t/hwl_{\perp} && (7) \end{aligned}$$

We can eliminate the variables $E_0, E_1, F_0, F_1, F_2, F_3$ from these equations, leaving a quadratic equation in t:

$$0 = \ell_1\ell_2 \quad - \left(\frac{\ell_1 + \ell_2}{hwl_{\perp}}\right) t \quad + \left(\frac{1}{hwl_{\perp}^2} - 4\frac{hwl_{\|}}{hwl_{\perp}^2}\right) t^2. \tag{8}$$

We solve this quadratic equation for t; again, we will need to select the “better” of the two solutions, which we do by constructing both solutions and choosing the one that doesn’t yield a loop. There are multiple ways to compute the control points `cp(2),cp(3),cp(4)` at this point. We chose to compute $F_1 = F_2 = F_{12}$ from (4), F_0 and F_1 from (6) and (7), and then compute E_0 and E_3 from (2) and (3). We now compute

```
>> cp(2) = p1+hdual(E0*v1); cp(2) = cp(2)/norm(cp(2));
>> cp(4) = p2-hdual(E3*v2); cp(4) = cp(4)/norm(cp(4));
```

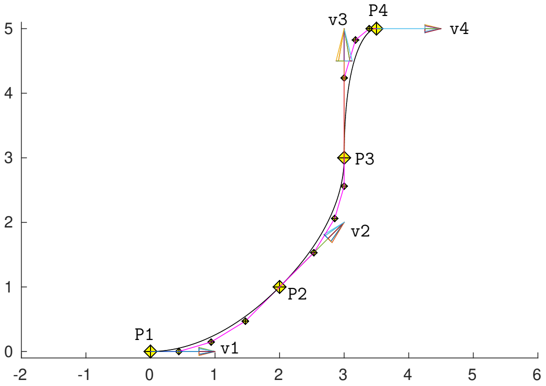


Figure 7: Three quartic Pythagorean hodograph curves fit to four points and vectors. The larger points are the data points (as well as the first/last control points of the segments); the smaller points are the interior control points of the segments.

and finally compute `cp(3)` as the translation of `ip` by t in the $\vec{d_\perp}$ direction. Alternatively, we could have computed `cp(2),cp(3),cp(4)` by translating `hwl` by t (so that it passes through what will be `cp(3)`), intersected this translated line with the lines `P1,v1` and `P2,v2` to get the points `s1` and `s2`, and then used E_0, F_0, F_1, F_2, F_3 and E_3 as weights to compute `cp(2),cp(3),cp(4)` as affine combinations of `P1`, `s1`, `s2`, and `P2`.

Figure 7 gives an example of constructing three quartic segments to interpolate four points, P_1, P_2, P_3, P_4, and four directions $\vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4$. Note that the significant difference in the lengths of the derivatives at the joint at `P3`, as expected at times with G^1 curves.

3.2 Quartic G^2 Pythagorean Hodograph Curves

The problem with setting $F_1 = F_2$ in the previous section is that we are basically wasting a degree of freedom. Potentially, there could be (and likely is) data that cannot be interpolated by setting $F_1 = F_2$ but could be interpolated if we were to choose some other relative setting of F_1 to F_2. Further, by setting $F_1 = F_2$, the resulting curve is just a degree raised version of the cubic curve fit to the same data. Regardless, we can make use of this extra degree of freedom to achieve other geometric goals. In this section, we use this degree of freedom to obtain G^2 continuity between segments while still interpolating the position and tangent directions at a sequence of points.

We start with the Pythagorean hodograph constraints of [WangFang09]:

$$\begin{aligned}
&\#1\ E_0F_2 = 3F_0F_1 \quad \Rightarrow \quad F_0 = E_0F_2/(3F_1)\\
&\#2\ E_3F_1 = 3F_2F_3 \quad \Rightarrow \quad F_3 = E_3F_1/(3F_2)\\
&\#3\ F_1F_2 = 4F_0F_3
\end{aligned}$$

Again let $s = \frac{hwl_{\|}}{hwl_{\perp}}$ and $h_{\perp} = hwl_{\perp}$, and add the geometric constraints of the last section:

$$\begin{aligned}
\#4\ F_1 + F_2 &= 2st\\
\#5\ E_0 + F_0 &= \ell_1 - t/h_{\perp}\\
\#6\ E_3 + F_3 &= \ell_2 - t/h_{\perp}
\end{aligned}$$

Note that these constraints do two things: first, they enforce the geometry that accompanies the PH constraints, and second, they enforce the G^1 interpolation conditions; i.e., without these constraints, you can get solutions where the PHC curve satisfies the constraints of Wang and Fang, but the curve fails to interpolate the tangent vectors.

We now add the G^2 constraint. Since we only have one degree of freedom remaining after interpolating the two positions and tangents, we will only require that the curvature of the curve at $t = 0$ interpolates a given value. The curvature k of a curve f is given by the formula $k = (f' \times f'')/|f'|^3$. Using the facts that $f'(0) = 4(cp_2 - cp_1) = 4E_0v_1$, and $cp_3 - cp_2 = |cp_3 - cp_2|(\alpha v_1 + \gamma v_1^{\perp}) = F_1\gamma v_1^{\perp}$ gives

$$\begin{aligned}
k &= (f' \times f'')/|f'|^3\\
&= 4(cp_2 - cp_1) \times 12((cp_3 - cp_2) - (cp_2 - cp_1))/|4(cp_2 - cp_1)|^3\\
&= 3(E_0v_1) \times F_1\gamma v_1^{\perp}/(4E_0^3)\\
&= 3F_1\gamma/(4E_0^2),
\end{aligned}$$

giving us our seventh equation,

$$\#7 \; 4kE_0^2 = 3F_1\gamma$$

Thus, we have seven equations in seven unknowns (E_0,E_3,F_0,F_1,F_2,F_3,t). Using Maple, we eliminated F_0,F_3,F_2,F_1,E_3,t (in that order), leaving a quartic equation in E_0:

$$0 = 16k^3\,\mathbf{E}_0^4 + 32\gamma h_\perp k^2 s\,\mathbf{E}_0^3 + 4k\gamma^2 - 36k^2\gamma h_\perp l_1 s\,\mathbf{E}_0^2 + (6k\gamma^2)(l_2 - l_1)\,\mathbf{E}_0 + 3\gamma^3 h_\perp l_2 s$$

We can now compute the remaining six variables in sequence as

$$\begin{aligned}
t &= -2/3kE_0(2E_0 - 3l1)h_\perp/(\gamma h_\perp s + 2kE_0)\\
E_3 &= -12(E_0 h_\perp - h_\perp l1 + t)E_0^2k^2/(h_\perp\gamma^2)\\
F_1 &= 4kE_0^2/(3\gamma)\\
F_2 &= (4E_0E_3)/(9F_1)\\
F_3 &= (E_3F_1)/(3F_2)\\
F_0 &= (E_0F_2)/(3F_1)
\end{aligned}$$

We can now construct a sequence of quartic Bézier curves that interpolate the position and tangents at a sequence of points, where the segments meet with G^2 continuity; we construct the first segment to interpolate the first two points and tangents, and then construct each subsequent segment to interpolate the next two points and tangents, as well as meet the previous segment with G^2 continuity. Figure 8 shows an example of three such segments.

This is approach is somewhat limited. In particular, quartic PHC's do not have inflection points. Thus, adding an inflection to to the curve sequence (e.g., to get the curve to bend the other way) likely requires giving up G^2 continuity at a joint point where the inflection occurs.

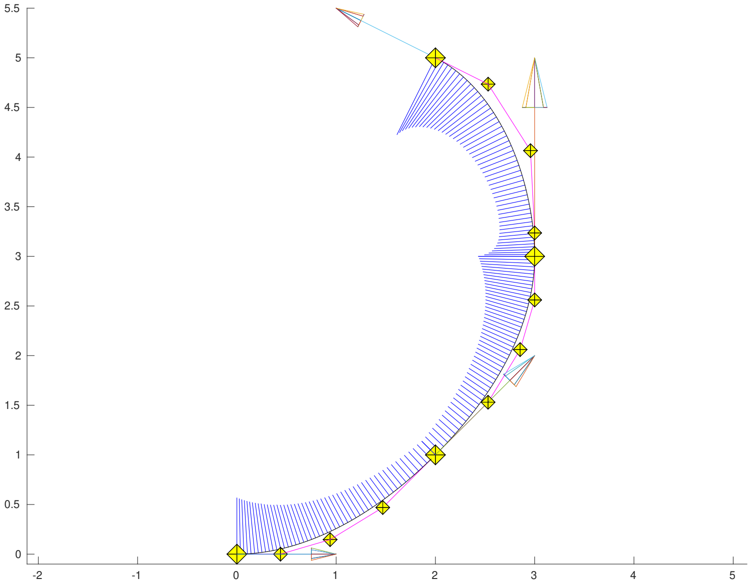


Figure 8: Three quartic segments that meet with G^2 continuity. The blue line segments point in the direction of the normal to the curve and are scaled by the curvature of the curve at that point. The larger points are the interpolated data points as well as the first/last control points of the curve segments; the smaller points are the interior control points of the curve segments.

4 Quintic PHC

Hormann et al. [HRV24] gave multiple characterizations of when a quintic curve is a PH curve. Here we work with one of their geometric constructions. They later gave a C^1 Hermite construction, interpolating two position and first derivative vectors [HRV25]. In this section, we give a G^2 construction, where our quintic PHC will interpolate position, tangent direction, and curvature at two points.

The characterization of Hormann et al. that we use constructs a sequence of similar triangles. Referring to Figure 9(b), we start with control points `cp(1),...,cp(6)`. We trisect the segment `cp(2),cp(3)` (giving `s1` and `s2`) and we trisect the segment `cp(4),cp(5)`. (giving `s3` and `s4`). We then construct `q3` and `q2` such that we have two sets of similar triangles:

$$\begin{aligned} \triangle s_1, cp(3), q_3 &\sim \triangle cp(1), cp(2), cp(3) \\ \triangle q_2, cp(4), s_4 &\sim \triangle cp(4), cp(5), cp(6). \end{aligned}$$

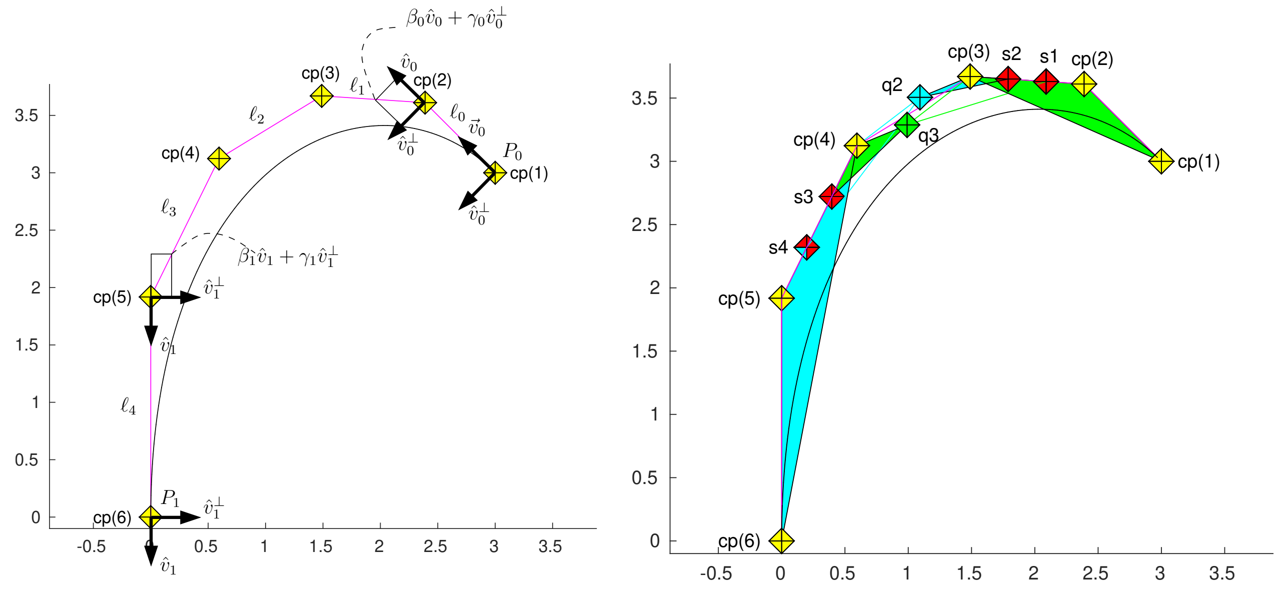
Then Hormann et al. showed that the curve is a PH curve if and only if

$$\begin{aligned} \triangle q_3, cp(4), s_3 &\sim \triangle cp(1), cp(2), cp(3) \\ \triangle s_2, cp(3), q_2 &\sim \triangle cp(4), cp(5), cp(6), \end{aligned}$$

with the additional requirement that $\ell_1, \ell_3 \neq 0$.

In short, starting from control points `cp(1)`,...,`cp(6)`, we trisect the segment `cp(2),cp(3)` giving `s1`,`s2`, and we trisect the segment `cp(4),cp(5)` giving `s3`,`s4` (see Figure 9). We then construct `q2` so that we have similar triangles $\triangle s1, cp3, q3 \sim \triangle cp1, cp2, cp3$. The first condition is that we have similar triangles $\triangle q3, cp4, s3 \sim \triangle cp1, cp2, cp3$. Constructing `q2` in an analogous way from the other end of the control polygon, the second condition is that $\triangle q2, cp3, s2 \sim \triangle cp6, cp5, cp4$.

Let ℓ_{i-1} be the length of the edge $e_{i-1} = cp(i+1) - cp(i)$ (e.g., the edge $cp(1), cp(2)$ is e_0). If we want to interpolate positions P_0, P_1 and directions $\hat{v}_0$ and $\hat{v}_1$, then we have



(a)

(b)

Figure 9: Quintic. (a) Given data sans k_i with resulting control points; (b) Similar triangles.

the following constraints:

$$
\begin{aligned}
cp1 &= P_0\\
cp2 &= P_0+\ell_0\hat{v}_0\\
cp6 &= P_1\\
cp5 &= P_1-\ell_4\hat{v}_1
\end{aligned}
$$

We next add the PH constraints. We begin by first expressing `cp(3)` relative to `cp(2)` and (implicitly) `cp(1)`:

$$
\begin{aligned}
cp3 &= cp2+\ell_1(\beta_0\hat{v}_0+\gamma_0\hat{v}_0^\perp)\\
cp4 &= cp5-\ell_3(\beta_1\hat{v}_1+\gamma_1\hat{v}_1^\perp)
\end{aligned}
$$

where $\hat{v}_0^\perp$ is the unit vector perpendicular to $\hat{v}_0$ and $\beta_i^2+\gamma_i^2=1$ and thus $e_1=\ell_1(\beta_0\hat{v}_0+\gamma_0\hat{v}_0^\perp)$ and $e_3=\ell_3(\beta_1\hat{v}_1+\gamma_1\hat{v}_1^\perp)$, and define $\hat{e}_1=\beta_0\hat{v}_0+\gamma_0\hat{v}_0^\perp$ and $\hat{e}_1^\perp=-\gamma_0\hat{v}_0+\beta_0\hat{v}_0^\perp$, and $\hat{e}_3 = \beta_1 \hat{v}_1 + \gamma_1 \hat{v}_1^\perp$ and $\hat{e}_3^\perp = -\gamma_1 \hat{v}_1 + \beta_1 \hat{v}_1^\perp$.

To compute q3 (and similarly, q2), we use the β, γ weightings, but on $\hat{e}_1$ and $\hat{e}_1^\perp$ and scaled by 2/3 and by ℓ_1^2/ℓ_0:

$$\begin{aligned} q3 &= cp3 + \frac{2}{3}\frac{\ell_1^2}{\ell_0}(\beta_0 \hat{e}_1 + \gamma_0 \hat{e}_1^\perp) \\ q2 &= cp4 - \frac{2}{3}\frac{\ell_3^2}{\ell_4}(\beta_1 \hat{e}_3 + \gamma_1 \hat{e}_3^\perp). \end{aligned}$$

where the $\frac{2}{3}\frac{\ell_1^2}{\ell_0}$ scaling is because we want the ratio of the lengths of the edge `s1,cp(3)` to `cp(3),q3` to be the same as the ratio between `cp(1),cp(2)` and `cp(2),cp(3)`, or ℓ_0/ℓ_1, with the distance scaled by 2/3 since the distance from `s1` to `cp(3)` is 2/3 the distance from `cp(2)` to `cp(3)`.

Note that we can also express q3 and q2 “from the other side”. I.e., q3 can be expressed relative to s3 and `cp(4)`. Essentially, we need to express `cp(1)` in terms of `cp(2)`, $\hat{e}_1$, and $\hat{e}_1^\perp$ and apply those weights to `cp(4)`, $\hat{e}_3$ and $\hat{e}_3^\perp$.

We start with

$$cp1 = cp2 - \ell_0 \hat{v}_0.$$

We want to reexpress $\hat{v}_0$ in terms of $\hat{e}_1$ and $\hat{e}_1^\perp$. Since $\hat{e}_1 = \beta_0 \hat{v}_0 + \gamma_0 \hat{v}_0^\perp$ and $\hat{e}_1^\perp = -\gamma_0 \hat{v}_0 + \beta_0 \hat{v}_0^\perp$, we have

$$\begin{aligned} \hat{e}_1 - \frac{\gamma_0}{\beta_0}\hat{e}_1^\perp &= \beta_0 \hat{v}_0 + \gamma_0 \hat{v}_0^\perp - \frac{\gamma_0}{\beta_0}(-\gamma_0 \hat{v}_0 + \beta_0 \hat{v}_0^\perp) \\ &= \beta_0 \hat{v}_0 + \gamma_0 \hat{v}_0^\perp + \frac{\gamma_0^2}{\beta_0}\hat{v}_0 - \gamma_0 \hat{v}_0^\perp \\ &= \frac{\beta_0^2 + \gamma_0^2}{\beta_0}\hat{v}_0 = \frac{1}{\beta_0}\hat{v}_0 \\ \Rightarrow & \\ \hat{v}_0 &= \beta_0 \hat{e}_1 - \gamma_0 \hat{e}_1^\perp. \end{aligned}$$

Thus,

$$cp1 = cp2 - \ell_0(\beta_0 \hat{e}_1 - \gamma_0 \hat{e}_1^\perp).$$

We take a similar weighting of $\hat{e}_3$ and $\hat{e}_3^{\perp}$ as an offset from `cp(4)` to get `q3` ”from the other side:”

$$q3 \quad = \quad cp4 - \frac{1}{3}\ell_3\frac{\ell_0}{\ell_1}(\beta_0\hat{e}_3 - \gamma_0\hat{e}_3^{\perp}),$$

where we have to rescale to get the appropriate similar triangle. Likewise, we can get an expression for `q2`:

$$q2 \quad = \quad cp3 + \frac{1}{3}\ell_1\frac{\ell_4}{\ell_3}(\beta_1\hat{e}_1 - \gamma_1\hat{e}_1^{\perp}).$$

Now equate the two `q3`’s and reexpress to eliminate the control points:

$$\begin{aligned}
cp3 + \frac{2}{3}\frac{\ell_1^2}{\ell_0}(\beta_0\hat{e}_1 + \gamma_0\hat{e}_1^{\perp}) &= cp4 - \frac{1}{3}\ell_3\frac{\ell_0}{\ell_1}(\beta_0\hat{e}_3 - \gamma_0\hat{e}_3^{\perp}) \\
cp2 + \ell_1(\beta_0\hat{v}_0 + \gamma_0\hat{v}_0^{\perp}) + \frac{2}{3}\frac{\ell_1^2}{\ell_0}(\beta_0\hat{e}_1 + \gamma_0\hat{e}_1^{\perp}) &= cp5 - \ell_3(\beta_1\hat{v}_1 + \gamma_1\hat{v}_1^{\perp}) - \frac{1}{3}\ell_3\frac{\ell_0}{\ell_1}(\beta_0\hat{e}_3 - \gamma_0\hat{e}_3^{\perp}) \\
P_0 + \ell_0\hat{v}_0 + \ell_1(\beta_0\hat{v}_0 + \gamma_0\hat{v}_0^{\perp}) + \frac{2}{3}\frac{\ell_1^2}{\ell_0}(\beta_0\hat{e}_1 + \gamma_0\hat{e}_1^{\perp}) &= P_1 - \ell_4\hat{v}_1 - \ell_3(\beta_1\hat{v}_1 + \gamma_1\hat{v}_1^{\perp}) \\
& \quad -\frac{1}{3}\ell_3\frac{\ell_0}{\ell_1}(\beta_0\hat{e}_3 - \gamma_0\hat{e}_3^{\perp}) \\
\ell_0\ell_1P_0 + \ell_0^2\ell_1\hat{v}_0 + \ell_0\ell_1^2(\beta_0\hat{v}_0 + \gamma_0\hat{v}_0^{\perp}) &= \ell_0\ell_1P_1 - \ell_0\ell_1\ell_4\hat{v}_1 - \ell_0\ell_1\ell_3(\beta_1\hat{v}_1 + \gamma_1\hat{v}_1^{\perp}) \\
+\frac{2}{3}\ell_1^3(\beta_0\hat{e}_1 + \gamma_0\hat{e}_1^{\perp}) & \quad -\frac{1}{3}\ell_3\ell_0^2(\beta_0\hat{e}_3 - \gamma_0\hat{e}_3^{\perp}) \\
\ell_0\ell_1P_0 + \ell_0^2\ell_1\hat{v}_0 + \ell_0\ell_1^2(\beta_0\hat{v}_0 + \gamma_0\hat{v}_0^{\perp}) &= \ell_0\ell_1P_1 - \ell_0\ell_1\ell_4\hat{v}_1 - \ell_0\ell_1\ell_3(\beta_1\hat{v}_1 + \gamma_1\hat{v}_1^{\perp}) \\
+\frac{2}{3}\ell_1^3(\beta_0(\beta_0\hat{v}_0 + \gamma_0\hat{v}_0^{\perp}) + \gamma_0(-\gamma_0\hat{v}_0 + \beta_0\hat{v}_0^{\perp})) & \quad -\frac{1}{3}\ell_3\ell_0^2(\beta_0(\beta_1\hat{v}_1 + \gamma_1\hat{v}_1^{\perp}) - \gamma_0(-\gamma_1\hat{v}_1 + \beta_1\hat{v}_1^{\perp}))
\end{aligned}$$

$$\begin{aligned}
cp4 - \frac{2}{3}\frac{\ell_3^2}{\ell_4}(\beta_1\hat{e}_3 + \gamma_1\hat{e}_3^\perp) &= cp3 + \frac{1}{3}\ell_1\frac{\ell_4}{\ell_3}(\beta_1\hat{e}_1 - \gamma_1\hat{e}_1^\perp) \\
cp5 - \ell_3(\beta_1\hat{v}_1 + \gamma_1\hat{v}_1^\perp) - \frac{2}{3}\frac{\ell_3^2}{\ell_4}(\beta_1\hat{e}_3 + \gamma_1\hat{e}_3^\perp) &= cp2 + \ell_1(\beta_0\hat{v}_0 + \gamma_0\hat{v}_0^\perp) + \frac{1}{3}\ell_1\frac{\ell_4}{\ell_3}(\beta_1\hat{e}_1 - \gamma_1\hat{e}_1^\perp) \\
P_1 - \ell_4\hat{v}_1 - \ell_3(\beta_1\hat{v}_1 + \gamma_1\hat{v}_1^\perp) - \frac{2}{3}\frac{\ell_3^2}{\ell_4}(\beta_1\hat{e}_3 + \gamma_1\hat{e}_3^\perp) &= P_0 + \ell_0\hat{v}_0 + \ell_1(\beta_0\hat{v}_0 + \gamma_0\hat{v}_0^\perp) \\
&\quad + \frac{1}{3}\ell_1\frac{\ell_4}{\ell_3}(\beta_1\hat{e}_1 - \gamma_1\hat{e}_1^\perp) \\
\ell_4\ell_3P_1 - \ell_4^2\ell_3\hat{v}_1 - \ell_4\ell_3^2(\beta_1\hat{v}_1 + \gamma_1\hat{v}_1^\perp) &= \ell_4\ell_3P_0 + \ell_4\ell_3\ell_0\hat{v}_0 + \ell_4\ell_3\ell_1(\beta_0\hat{v}_0 + \gamma_0\hat{v}_0^\perp) \\
-\frac{2}{3}\ell_3^3(\beta_1\hat{e}_3 + \gamma_1\hat{e}_3^\perp) &\quad + \frac{1}{3}\ell_1\ell_4^2(\beta_1\hat{e}_1 - \gamma_1\hat{e}_0^\perp) \\
\ell_4\ell_3P_1 - \ell_4^2\ell_3\hat{v}_1 - \ell_4\ell_3^2(\beta_1\hat{v}_1 + \gamma_1\hat{v}_1^\perp) &= \ell_4\ell_3P_0 + \ell_4\ell_3\ell_0\hat{v}_0 + \ell_4\ell_3\ell_1(\beta_0\hat{v}_0 + \gamma_0\hat{v}_0^\perp) \\
-\frac{2}{3}\ell_3^3(\beta_1(\beta_1\hat{v}_1 + \gamma_1\hat{v}_1^\perp) + \gamma_1(-\gamma_1\hat{v}_1 + \beta_1\hat{v}_1^\perp)) &\quad + \frac{1}{3}\ell_1\ell_4^2(\beta_1(\beta_0\hat{v}_0 + \gamma_0\hat{v}_0^\perp) - \gamma_1(-\gamma_0\hat{v}_0 + \beta_0\hat{v}_0^\perp))
\end{aligned}$$

We now add the G^2 constraints. Again, starting from the formula for curvature, we have at $t = 0$

$$\begin{aligned}
k_0 &= \frac{f' \times f''}{|f'|^3} \\
&= (5\ell_0)(20\ell_1\gamma_0)|5\ell_0|^3 \\
&= 4\ell_1\gamma_0/(5\ell_0^2) \\
&\Rightarrow 5\ell_0^2k_0 = 4\ell_1\gamma_0
\end{aligned}$$

and a similar derivation at $t = 1$,

$$\begin{aligned}
k_1 &= (f' \times f'')/|f'|^3 \\
&= 5(cp_6 - cp_5) \times 20((cp_6 - cp_5) - (cp_5 - cp_4))/|5(cp_6 - cp_5)|^3 \\
&= 5(cp_6 - cp_5) \times 20(-(cp_5 - cp_4))/|5(cp_6 - cp_5)|^3 \\
&= -4(\ell_4\hat{v}_1) \times (\ell_3\gamma_1\hat{v}^\perp)/(5\ell_4^3) \\
&= -4\ell_3\gamma_1/(5\ell_4^2)
\end{aligned}$$

gives

$$-5\ell_4^2k_1 = 4\ell_3\gamma_1.$$

Note that in both G^2 derivations that $f' \times f''$ is a signed quantity; thus, the sign of the curvature k_0 is the same as the sign of γ_1 and the sign of k_1 is the opposite of γ_1. This correlation between these signs becomes important when giving initial conditions for a numerical solver as detailed in the next section.

4.1 Solving the Equations

We now have eight unknowns, $\ell_0, \ell_1, \ell_3, \ell_4, \beta_0, \beta_1, \gamma_0, \gamma_1$, and eight equations:

$$\begin{array}{rcl}
\ell_0\ell_1 P_0 + \ell_0^2\ell_1\hat{v}_0 + \ell_0\ell_1^2(\beta_0\hat{v}_0 + \gamma_0\hat{v}_0^\perp) & = & \ell_0\ell_1 P_1 - \ell_0\ell_1\ell_4\hat{v}_1 - \ell_0\ell_1\ell_3(\beta_1\hat{v}_1 + \gamma_1\hat{v}_1^\perp) \\
+\frac{2}{3}\ell_1^3(\beta_0^2\hat{v}_0 + \beta_0\gamma_0\hat{v}_0^\perp - \gamma_0^2\hat{v}_0 + \beta_0\gamma_0\hat{v}_0^\perp) & & -\frac{1}{3}\ell_3\ell_0^2(\beta_0\beta_1\hat{v}_1 + \beta_0\gamma_1\hat{v}_1^\perp + \gamma_0\gamma_1\hat{v}_1 - \gamma_0\beta_1\hat{v}_1^\perp) \\
\ell_4\ell_3 P_1 - \ell_4^2\ell_3\hat{v}_1 - \ell_4\ell_3^2(\beta_1\hat{v}_1 + \gamma_1\hat{v}_1^\perp) & = & \ell_4\ell_3 P_0 + \ell_4\ell_3\ell_0\hat{v}_0 + \ell_4\ell_3\ell_1(\beta_0\hat{v}_0 + \gamma_0\hat{v}_0^\perp) \\
-\frac{2}{3}\ell_3^3(\beta_1^2\hat{v}_1 + \beta_1\gamma_1\hat{v}_1^\perp - \gamma_1^2\hat{v}_1 + \gamma_1\beta_1\hat{v}_1^\perp) & & +\frac{1}{3}\ell_1\ell_4^2(\beta_1\beta_0\hat{v}_0 + \beta_1\gamma_0\hat{v}_0^\perp + \gamma_1\gamma_0\hat{v}_0 - \gamma_1\beta_0\hat{v}_0^\perp) \\
\gamma_0^2 + \beta_0^2 & = & 1 \\
\gamma_1^2 + \beta_1^2 & = & 1 \\
5\ell_0^2 k_0 & = & 4\ell_1\gamma_0 \\
-5\ell_4^2 k_1 & = & 4\ell_3\gamma_1.
\end{array}$$

where the first two equations are each two equations, one for x and one for y.

We could use the last two equations to eliminate γ_0 and γ_1 from the equations with $\gamma_0 = 5\ell_0^2 k_0/(4\ell_1)$ and $\gamma_1 = -5\ell_4^2 k_1/(4\ell_3)$. Doing so would leave us with an equation that is linear in β_0 and another equation that is linear in β_1. We could then eliminate either of β_0 or β_1 from the equations, but not both, since after eliminating one of β_0 or β_1 the resulting set of 5 equations are all non-linear in each of the variables.

Instead, I solved the system of 8 equations numerically using Matlab's `fsolve` function. As initial guesses for our unknowns, we need to be a bit careful with γ_0 and γ_1, and make a choice that is consistent with the sign of the corresponding curvature. Regardless, my choices for initial conditions were

$$\ell_i = |P_1 - P_0|/5, \;\; \beta_0 = \beta_1 = 1, \;\; \gamma_0 = \pm 1, \;\; \gamma_1 = \pm 1.$$

4.2 Examples

As a first example, I fit a cubic PH curve to two position and tangents,

$$P_0 = (3,3),\;\; \vec{v}_0 = -\sqrt{2}/2\,e_1 + \sqrt{2}/2\,e_2,\;\; P_1 = (0,0),\;\; \vec{v}_1 = -e_2,$$

and then extracted the curvature at the end points of this PH curve to generated data to interpolate with a quintic PH curve:

$$\begin{aligned} P_0 &= (3,3),\;\; \vec{v}_0 = -\sqrt{2}/2\,e_1 + \sqrt{2}/2\,e_2,\;\; k_0 = 0.637558471367 \\ P_1 &= (0,0),\;\; \vec{v}_1 = -e_2,\;\; k_1 = 0.129281976953. \end{aligned}$$

I fit two quintic PH curves to this data, one with initial conditions taken from the cubic PH interpolant,

$$\ell_0 = 0.86377538854,\;\ell_1 = 0.901938067008,\;\ell_2 = 1.3440710474, \ell_3 = 1.91819097627,$$
$$\beta_0 = 0.751917396853, \gamma_0 = 0.659257330873, \beta_1 = 0.896820684934, \gamma_1 = -0.442394234901,$$

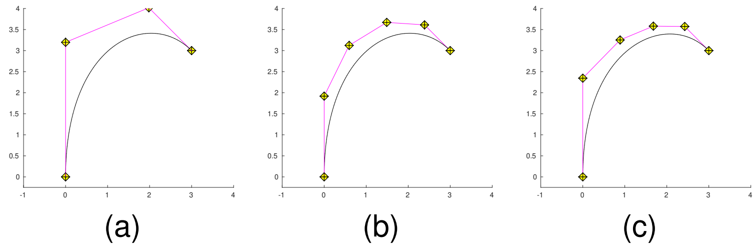
and the other with initial conditions being those described in Section 4.1:

$$\ell_0 = \ell_1 = \;\ell_2 = \ell_3 = |P_1 - P_0|/5 = 0.8485, \beta_0 = 1, \gamma_0 = 1, \beta_1 = 1,\;\gamma_1 = -1.$$

As expected, the quintic PH curve constructed using “the solution” converged immediately to the solution. The quintic PH curve constructed using the second set of parameters also converged, but to a different PH curve. See Figure 10.

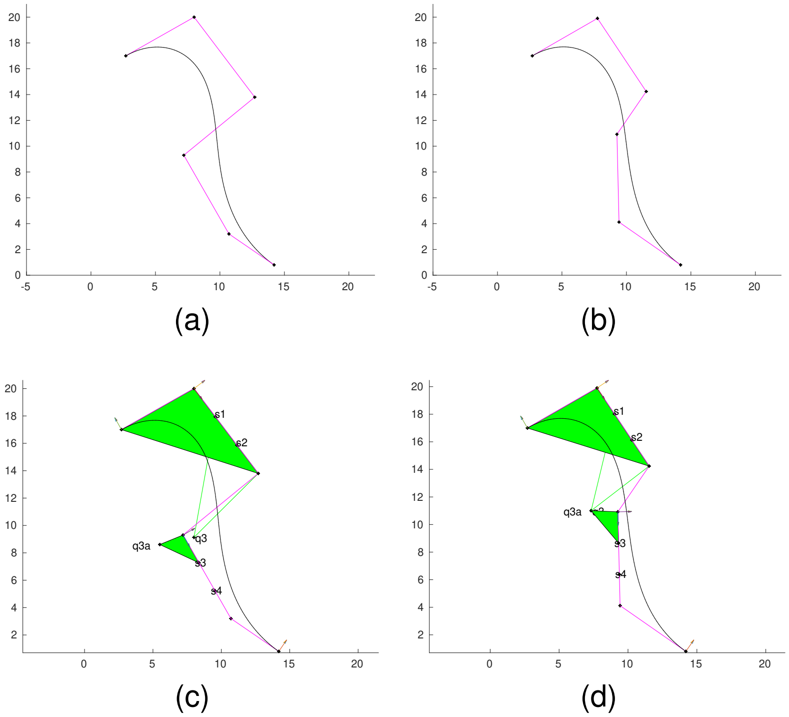
As a second example, I hand digitized the control points of a quintic PH curve in Hormann talk; note that unlike the curve in Hormann’s talk, my hand digitized curve is not a PH curve. I then evaluated this curve at its endpoints for position, tangent, and curvature. I then found a quintic PH curve that interpolated this data. See Figure 11.

As a third example, in Figure 12, I fit three four curves to a fixed pair of points and directions, and a fixed curvature on one end of the curve and varied the curvature on the other end.



(a) (b) (c)

Figure 10: (a) Cubic PH curve used to obtain curvature. (b) Quintic PH curve, starting with parameters of curve in (a) constructs the degree raised version of the cubic PH curve. (c) Quintic PH curve using “standard” parameters.



(a) (b)

(c) (d)

Figure 11: (a) Quintic non-PH curve used to obtain curvature. (b) Quintic PH curve, fit the end positions, tangents and curvature of curve in part (a). Figures (c) and (d) show the construction of one of the similar triangle conditions for the curves in parts (a) and (b).

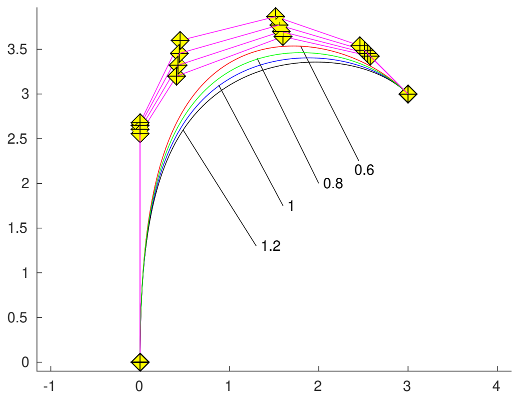


Figure 12: Quintic PH curves fit to: $P_0 = (0,0),\ \vec{v}_0 = e2,\ \kappa_0 = 0.05$; $P_1 = (3,3), \vec{v}_1 = \sqrt(2)/2(e_1 + -e_2)$. Red: $\kappa_1 = 0.6$. Green: $\kappa_1 = 0.8$. Blue: $\kappa_1 = 1$. Black: $\kappa_1 = 1.2$

4.3 Discussion

Not all data will lead to a solution (or a reasonable solution) for this quintic, G^2 Hermite PH curve construction. For example, for the data in Figure 12, if κ_1 is set to 0.5, then end tangents of the resulting curve collapse to 0, and the resulting curve is a straight line, although possibly different initial conditions would lead to a more reasonable solution.

The example in Figure 12 (and in other tests I ran) suggest that for data that has a solution, small variations on this data will also yield a solution. And the example in Figure 11 suggests that sampling the data (positions, tangents, and curvatures) from a “reasonable” curve should yield a reasonable quintic PHC that interpolates this data. However, only a small number of examples were tested, and additional work remains to determine conditions on the data that yields a solution.

Further, as shown in Figure 10, multiple solutions exist. Solving for and selecting a “best” solution is also a venue for future work.

References

[DdK] Leo Dorst and Steven De Keninck. *A Guided Tour to the Plane-Based Geometric Algebra PGA*. Available at http://bivector.net/PGA4CS.html.

[DMF07] L. Dorst, S. Mann, and D. Fontijne. *Geometric Algebra for Computer Science*. Morgan-Kaufmann, 2007.

[Farin91] G. Farin, *Curves and surfaces for CAGD: a practical guide*, 5th edition, Morgan Kaufmann Publishers Inc., San Francisco, 2001.

[Farouki] R. T. Farouki, *Pythagorean-Hodograph Curves: Algebra and Geometry Inseparable*, Springer Berlin, Heidelberg, 2008. https://doi.org/10.1007/978-3-540-73398-0

[Gunn11] C. Gunn, "Geometry, kinematics, and rigid body mechanics in Cayley-Klein geometries," Ph.D. dissertation, TUBerlin, 2011. [Online]. Available: http://page.math.tu-berlin.de/~gunn/Documents/Papers/ Thesis-final.pdf

[LM25] Zachary Leger and Stephen Mann, *A Tutorial for Plane-based Geometric Algebra*, online https://cs.uwaterloo.ca/~smann/PGABLE/, 2025.

[HRV24] Hormann, K., Romani, L., and Viscardi, A., *New algebraic and geometric characterizations of planar quintic Pythagorean-hodograph curves*, Journal of Computer Aided Geometric Design, 108 (2024), pg 1–14.

[HRV25] Hormann, K., Romani, L., and Viscardi, A. *New Characterizations of Planar Quintic Pythagorean-Hodograph Curves*, talk, SIAM Geometric Design Conference, Montreal, July, 2025.

[MeekWalton97] Meek, D. S. and Walton, D. J., *Geometric Hermite interpolation with Tschirnhausen cubits*, Journal of Computational and Applied Mathematics, 81 (1997) pg 299–309

[WangFang09] Wang, G. and Fang, L., *On control polygons of quartic Pythagorean–hodograph curves*, Journal of Computer Aided Geometric Design, 26 (2009) pg 1006–1015.