Evolution–based Least–Squares fitting using Pythagorean Hodograph Spline curves

Martin Aigner, Zbynek Šír and Bert Jüttler

Institute of Applied Geometry
Johannes Kepler University Linz, Austria

Abstract

The problem of approximating a given set of data points by splines composed of Pythagorean Hodograph (PH) curves is addressed. We discuss this problem in a framework that is not only restricted to PH spline curves, but can be applied to more general representations of shapes. In order to solve the highly non-linear curve fitting problem, we formulate an evolution process within the family of PH spline curves. This process generates a family of curves which depends on a time–like variable t. The best approximant is shown to be a stationary point of this evolution process, which is described by a differential equation. Solving it numerically by Euler's method is shown to be related to Gauss–Newton iterations. Different ways of constructing suitable initial positions for the evolution are suggested.

Key words: PH-curves, Least-Squares Fitting

1. Introduction

Curves with simple closed form descriptions of their parametric speed and arc–length are useful for various applications, such as NC machining. They greatly facilitate the control of the tool along a curved trajectory with constant (or user–defined) speed. In addition, these curves admit a simple exact representation of their offset curves.

This motivated the investigation of the interesting class of Pythagorean Hodograph (PH) curves, see (Farouki, 2002) and the references cited therein. This class consists of (piecewise) polynomial curves with a (piecewise) polynomial parametric speed, see Fig. 1 for an example. Various constructions for PH curves were developed. Due to the non–linear nature of PH curves,

Email address: {martin.aigner|zbynek.sir|bert.juettler}@jku.at (Martin Aigner, Zbynek Šír and Bert Jüttler).

URL: http://www.ag.jku.at (Martin Aigner, Zbynek Šír and Bert Jüttler).

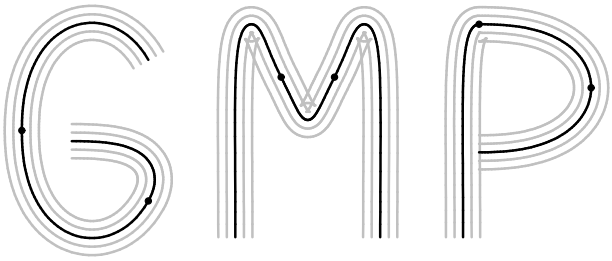

Fig. 1. Examples of piecewise polynomial Pythagorean hodograph curves (black) and their piecewise rational offsets (grey). Each character is composed of three PH quintics.

which rely on quadratic representation formulas for Pythagorean triplets and quadruplets in the ring of polynomials, these constructions are mainly based on local techniques, such as the interpolation of Hermite boundary data (Meek and Walton, 1997; Moon et al., 2001; Farouki et al., 1998a; Šír and Jüttler, 2005; Šír et al., 2006) [1] . These local constructions lead to small non–linear systems of equations, which can be dealt with efficiently.

In many situations, it is more appropriate to use global approximation techniques, such as least–squares fitting, since this generally reduces the data volume and produces a more compact representation. In the case of PH curves, very few global methods are available, dealing with interpolation and least–squares fitting (Farouki et al., 1998b, 2001). In the latter paper, the authors use non–linear optimization to generate a PH quintic which interpolates two boundary points and approximates additional points, where the parameter values assigned to them (i.e., the parameterization of the given points) are kept constant.

Even for simple curve representations, such as polynomial spline curves, curve fitting is a non–linear problem, due to the influence of the parameterization. Different approaches for dealing with the effects of this non–linearity have been developed (Alhanaty and Bercovier, 2001; Hoschek and Lasser, 1993; Rogers and Fog, 1989; Pottmann and Leopoldseder, 2003; Pottmann et al., 2005; Speer et al., 1998; Wang et al., 2006), such as 'parameter correction' or the use of quasi–Newton methods. Clearly the choice of a good initial solution is of outmost importance for the success of the optimization. Geometrically motivated optimization strategies (Pottmann and Leopoldseder, 2003; Pottmann et al., 2002, 2005; Wang et al., 2006), where the initial solution is replaced by an initial curve and the formulation of the problem uses some geometric insights, may lead to more robust techniques. Due to the iterative nature of the techniques for non–linear optimization, one may view the intermediate results as a time–dependent curve which tries to adapt itself to the target shape defined by he unorganized point data (Pottmann et al., 2002; Wang et al., 2006). This is related to the idea of 'active curves' used for image segmentation in Computer Vision (Kass et al., 1987).

Recently we formulated a general framework for evolution–based fitting of general objects (Aigner and Jüttler, 2006). It can be shown that the evolution process defines a flow on the manifold of objects (Aigner and Jüttler, 2007); consequently, the resulting path is independent on the choice of the shape parameters describing the objects. In the present paper we apply this framework to the case of PH curves. In addition we analyze its relation to the Gauss–Newton method.

The remainder of this paper is organized as follows. In the next two sections we recall some basics about PH curves, and we introduce a general framework for abstract curve fitting. Then,

[1] Similar techniques for space curves exist also, see (Farouki, 2002).

this framework will be applied to the special case of Pythagorean hodograph curves, and its relation to Gauss–Newton iteration will be analyzed. Finally we conclude the paper.

2. Pythagorean hodograph curves

The *hodograph* of a planar polynomial curve $\mathbf{c}(u) = [x(u), y(u)]^\top$ of degree n is the vector $\mathbf{h}(u) = [x'(u), y'(u)]^\top$ of degree $n-1$, where $'$ denotes the first derivative. Recall that a polynomial curve is called *Pythagorean Hodograph (PH)* if the length of its tangent vector is a (piecewise) polynomial of the parameter u. More precisely, $\mathbf{c}(u) = [x(u), y(u)]^\top$ is called a *planar PH curve* if there exists a polynomial $\sigma(u)$ such that

$$x'(u)^2 + y'(u)^2 = \sigma^2(u). \tag{1}$$

Three polynomials x', y' and σ satisfy equation (1) [2] if and only if there exist three polynomials α, β, ω such that

$$x' = \omega(\alpha^2 - \beta^2), \quad y' = \omega(2\alpha\beta), \quad \sigma = \omega(\alpha^2 + \beta^2), \tag{2}$$

see Kubota (1972). As a major advantage of PH curves, compared to 'ordinary' polynomial curves, they possess a (piecewise) polynomial arc length function

$$s(u) = \int_{u_0}^{u} |\sigma(v)| \, \mathrm{d}v \tag{3}$$

and (piecewise) rational offset curves (parallel curves)

$$\mathbf{o}_d(u) = \mathbf{c}(u) + \frac{d}{|\sigma(u)|}[y'(u), -x'(u)]^\top, \tag{4}$$

where d is the (oriented) offset distance.

Throughout the remainder of this paper we will assume that $\omega = 1$, restricting ourselves to curves with hodographs of the form

$$x'(u) = \alpha^2(u) - \beta^2(u), \quad y'(u) = 2\alpha(u)\beta(u). \tag{5}$$

As to be justified by Proposition 1, these PH curves will be called *regular PH curves*. They form a subset of all PH curves distinguished by the property that $\gcd(x'(u), y'(u))$ is a square of a polynomial. [3]

Regular PH curves can be constructed as follows: First we choose two polynomials $[\alpha(u), \beta(u)]^\top$ which define the so–called *preimage* curve. We generate the hodograph using (5) and integrate the two components. This gives the parametric representation of the PH curve.

Since two curves $\mathbf{c}(u)$, $\tilde{\mathbf{c}}(u)$ have the same hodograph if and only if they differ only by translation, a regular planar PH curve $\mathbf{p}(u)$ is fully determined by the preimage $[\alpha(u), \beta(u)]^\top$ and by the location of its starting point $\mathbf{c}(0)$ (which is specified by choosing the integration constant).

While the 'ordinary' PH curves may have cusps (namely for all parameter values of u which are roots of ω), regular PH curves are always tangent continuous.

Proposition 1 *Any regular (i.e., generated using* (5)*) Pythagorean hodograph curve* $\mathbf{c}(u)$*, where the two polynomials* $\alpha(u)$ *and* $\beta(u)$ *defining the preimage are not both identically to zero,* $(\alpha(u), \beta(u)) \not\equiv (0, 0)$*, has a smooth field of unit tangent vectors for all values* $u \in \mathbb{R}$ *of the*

[2] They are said to form a Pythagorean triplet in the ring of polynomials.
[3] This includes the generic case $\gcd(x'(u), y'(u)) = 1$.

curve parameter. Moreover its parametric speed and arc-length are polynomial functions, and its offsets are rational curves.

PROOF. Clearly, $\sigma(u) = \alpha(u)^2 + \beta(u)^2$ is a non-negative polynomial representing the speed function of $\mathbf{c}(u)$. The absolute value can be omitted in (3) and the arc-length function is a polynomial. Consider

$$\mathbf{q}(u) = \left[\frac{\alpha(u)^2 - \beta(u)^2}{\alpha(u)^2 + \beta(u)^2}, \frac{2\alpha(u)\beta(u)}{\alpha(u)^2 + \beta(u)^2}\right]^\top. \tag{6}$$

Except for the real roots of $\alpha(u)^2 + \beta(u)^2$, the vector $\mathbf{q}(u)$ is a unit vector tangent to $\mathbf{c}$ at $\mathbf{c}(u)$ and it has the same orientation as $[x'(u), y'(u)]^\top$. Moreover, any real root u_0 of $\alpha(u)^2 + \beta(u)^2$ has an even multiplicity $2k$, since $\alpha(u)^2 + \beta(u)^2$ is non-negative. Also $(u - u_0)^k$ must divide both $\alpha(u)$ and $\beta(u)$ and therefore $(u - u_0)^{2k}$ divides the numerators and the denominator of (6). After eliminating all common factors of numerators and denominators, we can therefore extend $\mathbf{q}(u)$ smoothly to $u \in \mathbb{R}$ and we obtain a smooth unit vector field along $\mathbf{c}(u)$.

Finally we note that the offset formula (4) simplifies to

$$\mathbf{o}_d(u) = \mathbf{c}(u) + d\mathbf{q}(u)^\perp, \tag{7}$$

and it defines a rational curve. □

The observation formulated in Proposition 1 can be seen as another advantage of PH curves, compared to the more general class of standard polynomial (Bézier) curves. The more general curves are not necessarily tangent continuous, since cusps may be present.

3. An abstract framework for curve fitting via evolution

We describe a general framework for the evolution-based approximation of a given data set by a curve. Later we will apply it to the special case of PH spline curves.

3.1. *Families of parametric curves and evolution of shape parameters*

We consider a parameterized family of planar parametric curves $(\mathbf{s}, u) \mapsto \mathbf{c}_\mathbf{s}(u)$. Two different kinds of parameters appear in the representation of the curve; the curve parameter u and a vector of shape parameters $\mathbf{s} = (s_1, \ldots, s_n)$.

For instance, one may consider a family of spline curves, where the shape parameters are both the control points and the knots. Later, in the case of PH spline curves, the shape parameters will be the control points defining the preimage curve and the integration constants.

We assume that the curve parameter varies within a fixed interval $I = [a, b]$ (the parameter domain of the curve), and that the vector of shape parameters $\mathbf{s}$ is contained in some domain $\Omega \subset \mathbb{R}^n$. For all $(\mathbf{s}, u) \in \Omega \times I$ the curve $\mathbf{c}_\mathbf{s}(u)$ shall depend continuously on the parameters. We assume that the curve has a well–defined normal vector at all points. Due to Proposition 1, this assumption is satisfied in the case of regular PH curves.

Among the curves of this family, we identify a curve that approximates a given set of (unordered) data points $\{\mathbf{p}_j\}_{j=1..N}$ in the least–squares sense. More precisely, we are looking for the vector of shape parameters that defines this curve.

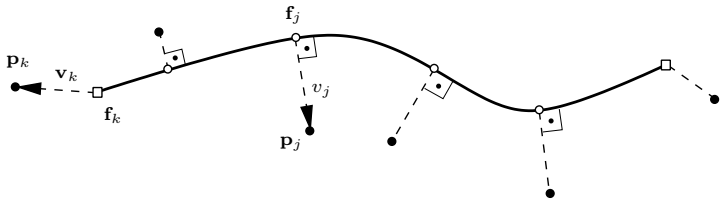

Fig. 2. Closest points and derived (normal) velocities.

We let the shape parameters $\mathbf{s}$ depend smoothly on an evolution parameter t, $\mathbf{s}(t) = (s_1(t), \ldots, s_n(t))$. The parameter t can be identified with the time. Starting with certain initial values, these parameters are modified continuously in time such that a given initial curve moves closer to the data points.

This movement will be governed by a system of differential equations of the form $\dot{\mathbf{s}}(t) = F(\mathbf{s}(t))$. By numerically solving this system and (approximately) computing the limit $\lim_{t\to\infty} \mathbf{s}(t)$, we obtain a curve $\mathbf{c}_{\mathbf{s}(t)}(u)$ which has minimal distance from the data points.

During the evolution of a curve $\mathbf{c}_{\mathbf{s}(t)}(u)$, each point travels with the velocity

$$\mathbf{v}_{\mathbf{s}(t)}(u) = \dot{\mathbf{c}}_{\mathbf{s}(t)}(u) = \sum_{i=1}^{n} \left.\frac{\partial \mathbf{c}_{\mathbf{s}}(u)}{\partial s_i}\right|_{\mathbf{s}=\mathbf{s}(t)} \dot{s}_i(t). \tag{8}$$

The dot denotes the derivative with respect to the time variable t. Since the tangential component of the velocity (8) can be seen as a reparameterization of the curve, we consider for points that do not lie on the boundary solely the normal velocity

$$v_{\mathbf{s}(t)}(u) = \mathbf{v}_{\mathbf{s}(t)}(u)^\top \mathbf{n}_{\mathbf{s}(t)}(u) = \sum_{i=1}^{n} \left(\left.\frac{\partial \mathbf{c}_{\mathbf{s}}(u)}{\partial s_i}\right|_{\mathbf{s}=\mathbf{s}(t)} \dot{s}_i(t) \right)^\top \mathbf{n}_{\mathbf{s}(t)}(u), \tag{9}$$

where $\mathbf{n}_{\mathbf{s}(t)}(u)$ denotes the unit normal of the curve in the point $\mathbf{c}_{\mathbf{s}(t)}(u)$. Note that the normal velocity depends linearly on the derivatives $\dot{s}_i(t)$ of the shape parameters.

3.2. *Evolution for approximation*

We will derive the evolution equation by specifying suitable velocities for some points of the curve. We assume that a set of data points $\{\mathbf{p}_j\}_{j=1,\ldots,N}$ is given. For each point, we consider the associated closest point $\mathbf{f}_j = \mathbf{c}_{\mathbf{s}(t)}(u_j)$ of the curve,

$$u_j = \arg\min_{u\in[a,b]} \|\mathbf{p}_j - \mathbf{c}_{\mathbf{s}(t)}(u)\|. \tag{10}$$

During the evolution, these points are expected to travel towards their associated data points. Consequently, the normal velocity $v(u_j)$, see (9), of a curve point $\mathbf{c}_{\mathbf{s}(t)}(u_j) = \mathbf{f}_j, u_j \notin \{a, b\}$, should be

$$d_j = (\mathbf{p}_j - \mathbf{f}_j)^\top \mathbf{n}_{\mathbf{s}(t)}(u_j). \tag{11}$$

If a closest point is one of the two boundary points ($u_j \in \{a, b\}$), then we consider the velocity, see (8), which should then be $\mathbf{d}_j = \mathbf{p}_j - \mathbf{f}_j$, cf. Fig. 2.

Following (8) and (9) we can compute for each point $\mathbf{f}_j$ the velocity or normal velocity on the one hand and the expected velocity (11) on the other hand. In general, the number of data points to be fitted exceeds the degrees of freedom of the curve to be fit to these data ($N \gg n$). Hence

the conditions for the velocities in the closest points cannot be fulfilled exactly. We choose the time derivatives of the shape parameters such that the conditions for the velocities are satisfied in the least–squares sense,

$$\dot{\mathbf{s}} = \arg\min_{\dot{\mathbf{s}}} \omega_\perp \sum_{\substack{j=1 \\ u_j \notin \{a,b\}}}^{N} (v_{\mathbf{s}(t)}(u_j) - d_j)^2 + \omega_{\mathbf{v}} \sum_{\substack{j=1 \\ u_j \in \{a,b\}}}^{N} \|\mathbf{v}_{\mathbf{s}(t)}(u_j) - (\mathbf{p}_j - \mathbf{f}_j)\|^2 + \omega_R R, \quad (12)$$

see (8), (9), (10) and (11). The non–negative weights $\omega_\perp \neq 0, \omega_{\mathbf{v}}$ and ω_R are used to control the influence of the three different terms. In order to ensure that a unique minimizer for the least–squares problem (12) exists, a regularization term R is added in (12). As a possibility one may use Tikhonov regularization, where $R = \|\dot{\mathbf{s}}(t)\|^2$.

As a necessary condition for a minimum, the derivatives of the right–hand side in (12) with respect to the $\dot{s}_i(t)$ vanish. Since these factors enter linearly in (8) and (9), the optimality condition yields a system of linear equations

$$\mathbf{M}(\mathbf{s}(t))\, \dot{\mathbf{s}}(t) = r(\mathbf{s}(t)), \quad (13)$$

where $\mathbf{M}$ is the coefficient matrix and r denotes the right–hand side. In general, this ODE cannot be solved exactly. Nevertheless, the vector $\dot{\mathbf{s}}$ can easily be computed for each given vector $\mathbf{s}$ by solving the linear system,

$$\dot{\mathbf{s}}(t) = F(\mathbf{s}(t)) = \mathbf{M}^{-1}(\mathbf{s}(t))\, r(\mathbf{s}(t)). \quad (14)$$

Using explicit Euler-steps $s_i(t+h) = s_i(t) + h\dot{s}_i(t)$, with a suitable step-size h, one can trace the evolving curves. This method for the numerical solution of the ODE corresponds to a discretization of the evolution in time. Using this simple method is sufficient, since the stationary point of the evolution is more important than the path leading to it.

In order to reduce the computational effort needed for computing the closest points on the curve (especially when the curve is still relatively far from the data), one can proceed as follows. As a preprocessing step, the distance field of the target shape is computed. This can be done efficiently using the graphics hardware, see Hoff et al. (1999). Starting with some equally spaced sensor points on some initial shape, the velocities (or normal velocities in the case of vertex points) can be defined with the help of the distance field. Finally, the sensor points are replaced by the closest points, if the distance to the data points drops below a certain threshold. Similarly, one may use velocities derived from other data, such as images, in order to deal with applications such as image segmentation.

3.3. *Stationary points of the evolution*

The solutions of the least–squares problem

$$\arg\min_{\mathbf{s}} \sum_{j=1}^{N} \min_{u_j \in [a,b]} \|\mathbf{p}_j - \mathbf{c}_{\mathbf{s}(t)}(u_j)\|^2 \quad (15)$$

are closely related to the evolution process defined by (14). In order to establish this connection, we need some technical assumptions. We assume, that the curve is non–singular ($\mathbf{c}'_{\mathbf{s}(t)}(u_j) \neq \mathbf{0}$) at the closest points $\mathbf{c}_{\mathbf{s}(t)}(u_j)$ to the data points. In addition, we exclude certain singular cases, e.g., when the number of degrees of freedom exceeds the number of data points when the data points lie in some degenerate position. This is made precise in the following definition.

Definition 2 *For a given curve $\mathbf{c}_{\mathbf{s}(t)}(u)$, consider a set $U = \{u_j\}_{j=1..N} \subset [a,b]$ of parameter values such that $\mathbf{c}'_{\mathbf{s}(t)}(u) \neq \mathbf{0}$ and $\{a,b\} \cap U = \emptyset$. The corresponding unit normal vectors are $\mathbf{n}_j = \mathbf{n}_{\mathbf{s}(t)}(u_j)$. The set of parameters U is said to be* **regular** *if the $N \times n$ matrix*

$$A_{j,k} = \mathbf{n}_j^\top \left. \frac{\partial \mathbf{c}_{\mathbf{s}}(u_j)}{\partial s_k} \right|_{\mathbf{s}=\mathbf{s}(t)}, \tag{16}$$

where s_k is the k–th component of the vector $\mathbf{s}$ of shape parameters, has maximal rank.

Lemma 3 *In a regular case and if all closest points are neither singular nor boundary points, then any solution of the usual least–squares fitting* (15) *of a curve $\mathbf{c}_{\mathbf{s}(t)}(u)$ is a stationary point of the differential equation derived from the evolution process.*

PROOF. As a necessary condition, the first derivatives of F with respect to the curve parameters $\{u_j\}_{j=1..N}$ and the shape parameters $\{s_i\}_{i=1..n}$ vanish, where F is the sum of squared errors in (15),

$$\frac{\partial F}{\partial u_j} = 2\left(\mathbf{p}_j - \mathbf{c}_{\mathbf{s}(t)}(u_j)\right)^\top \frac{\partial \mathbf{c}_{\mathbf{s}(t)}(u_j)}{\partial u_j} = 0, \tag{17}$$

and

$$\left. \frac{\partial F}{\partial s_i} \right|_{\mathbf{s}=\mathbf{s}(t)} = 2\sum_{j=1}^{N} \left(\mathbf{p}_j - \mathbf{c}_{\mathbf{s}(t)}(u_j)\right)^\top \left. \frac{\partial \mathbf{c}_{\mathbf{s}}(u_j)}{\partial s_i} \right|_{\mathbf{s}=\mathbf{s}(t)} = 0. \tag{18}$$

On the other hand, the ODE defining the curve evolution is found by computing the first derivatives of

$$\sum_{j=1}^{N} \left(\left(\mathbf{v}_j - (\mathbf{p}_j - \mathbf{f}_j)\right)^\top \mathbf{n}_j \right)^2$$

with respect to the derivatives of the shape parameters $\dot{s}_k$, where $\mathbf{v}_j = \mathbf{v}_{\mathbf{s}(t)}(u_j)$, see (9). This yields

$$\forall k: \; 2\sum_{j=1}^{N} \left(\mathbf{v}_j - (\mathbf{p}_j - \mathbf{f}_j)\right)^\top \mathbf{n}_j \mathbf{n}_j^\top \left. \frac{\partial \mathbf{c}_{\mathbf{s}}(u_j)}{\partial s_k} \right|_{\mathbf{s}=\mathbf{s}(t)} = 0 \tag{19}$$

Due to (17), the error vectors $\mathbf{p}_j - \mathbf{f}_j$ are perpendicular to the tangent vectors, hence

$$(\mathbf{p}_j - \mathbf{f}_j)^\top \mathbf{n}_j \mathbf{n}_j^\top = (\mathbf{p}_j - \mathbf{f}_j)^\top. \tag{20}$$

Taking (18) into account, (19) simplifies to

$$\sum_{j=1}^{N} \mathbf{v}_j^\top \mathbf{n}_j \mathbf{n}_j^\top \left. \frac{\partial \mathbf{c}_{\mathbf{s}}(u_j)}{\partial s_k} \right|_{\mathbf{s}=\mathbf{s}(t)} = \sum_{j=1}^{N} \left(\sum_{i=0}^{n} \left. \frac{\partial \mathbf{c}_{\mathbf{s}}(u_j)}{\partial s_i} \right|_{\mathbf{s}=\mathbf{s}(t)} \dot{s}_i(t) \right)^\top \mathbf{n}_j \mathbf{n}_j^\top \left. \frac{\partial \mathbf{c}_{\mathbf{s}}(u_j)}{\partial s_k} \right|_{\mathbf{s}=\mathbf{s}(t)} = 0.$$

Rewriting this equation we get

$$\forall k: \; \sum_{i=0}^{n} \sum_{j=1}^{N} \left[\left(\left. \frac{\partial \mathbf{c}_{\mathbf{s}}(u_j)}{\partial s_i} \right|_{\mathbf{s}=\mathbf{s}(t)} \right)^\top \mathbf{n}_j \mathbf{n}_j^\top \left. \frac{\partial \mathbf{c}_{\mathbf{s}(t)}(u_j)}{\partial s_k} \right|_{\mathbf{s}=\mathbf{s}(t)} \right] \dot{s}_i(t) = 0. \tag{21}$$

Using matrix notation this can be rewritten as

$$A^\top A\, \dot{\mathbf{s}}(t) = 0, \tag{22}$$

where the components of A are defined as in (16). This system has only the trivial solution if the matrix $A^\top A$ is regular which corresponds to rank(A)$= n+1$. In a regular case this condition holds. □

We will continue this discussion in Section 6, where we prove that the evolution is equivalent to a Gauss–Newton step for the least–squares problem (15).

4. Evolution of PH splines

In this section we apply the general framework of the previous section to the case of PH splines. In order to simplify the notation, we will simply write $\mathbf{s}$ instead of $\mathbf{s}(t)$, omitting the dependency of the shape parameters on the time variable t.

In order to generate PH spline curves, we represent the preimage $[\alpha(u), \beta(u)]$ as an open integral B-spline curve (Hoschek and Lasser, 1993, p. 176). Let

$$(u_0 = u_1 = \ldots = u_{k-1}, u_k, u_{k+1}, \ldots, u_m, u_{m+1} = u_{m+2} = \ldots = u_{m+k}) \tag{23}$$

be a given knot vector and $N_{i,k}(u)$, $(i = 0, \ldots, m)$ the associated B-spline functions of order k. Then $N_{i,k}(u)$ form a basis of the linear space of piecewise polynomials of degree $k-1$ on the interval $[u_{k-1}, u_{m+1}]$ which are C^{k-2} at the points $\{u_i, i = k, \ldots, m\}$. We choose the components $\alpha(u), \beta(u)$ of the preimage from this space of functions,

$$\alpha(u) = \sum_{i=0}^{m} \alpha_i N_{i,k}(u) \quad \text{and} \quad \beta(u) = \sum_{i=0}^{m} \beta_i N_{i,k}(u). \tag{24}$$

The resulting PH spline is obtained as

$$\mathbf{c_s}(u) = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \int_{u_{k-1}}^{u} \begin{bmatrix} \alpha^2(\tilde{u}) - \beta^2(\tilde{u}) \\ 2\alpha(\tilde{u})\beta(\tilde{u}) \end{bmatrix} \mathrm{d}\tilde{u} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \sum_{i=0}^{m}\sum_{j=0}^{m} \begin{bmatrix} \alpha_i\alpha_j - \beta_i\beta_j \\ 2\alpha_i\beta_j \end{bmatrix} K_{i,j}(u)$$

where the piecewise polynomials $K_{i,j}(u)$ of degree $2k-1$ are defined as

$$K_{i,j}(u) := \int_{u_{k-1}}^{u} N_{i,k}(\tilde{u}) N_{j,k}(\tilde{u}) \mathrm{d}\tilde{u}. \tag{25}$$

As shape parameters – in the sense of the previous section – we can consider the spline end point coordinates x_0, y_0, the spline coefficients α_i, β_i and even the knots u_i. (Note that from now on we suppress the t denoting the time dependency.) In our implementation we have kept the knot vector fixed and considered only an evolution with respect to the following $n = 2m + 4$ shape parameters

$$\mathbf{s} = \{x_0, y_0, \alpha_0, \ldots, \alpha_m, \beta_0, \ldots, \beta_m\}. \tag{26}$$

We compute the quantities occurring in (12). The partial derivatives of $\mathbf{c_s}(u)$ with respect to the shape parameters are

$$\frac{\partial \mathbf{c_s}(u)}{\partial x_0} = [1,0]^\top, \qquad \frac{\partial \mathbf{c_s}(u)}{\partial y_0} = [0,1]^\top,$$

$$\frac{\partial \mathbf{c_s}(u)}{\partial \alpha_i} = 2\sum_{j=0}^{m} [\alpha_j, \beta_j]^\top K_{i,j}(u) \text{ and } \quad \frac{\partial \mathbf{c_s}(u)}{\partial \beta_i} = 2\sum_{j=0}^{m} [-\beta_j, \alpha_j]^\top K_{i,j}(u).$$

The velocity (8) of any curve point $\mathbf{c}(u)$ equals

$$\mathbf{v_s}(u) = [\dot{x}_0, \dot{y}_0]^\top + 2\sum_{i=0}^{m}\sum_{j=0}^{m} [\alpha_j\dot{\alpha}_i - \beta_j\dot{\beta}_i, \beta_j\dot{\alpha}_i + \alpha_j\dot{\beta}_i]^\top K_{i,j}(u), \tag{27}$$

which is linear in the derivatives $\dot{x}_0, \dot{y}_0, \dot{\alpha}_i, \dot{\beta}_i$ of the shape parameters. The unit normals are

$$\mathbf{n}_{\mathbf{s}}(u) = \frac{\mathbf{c}'_{\mathbf{s}}(u)^{\perp}}{\alpha(u)^2 + \beta(u)^2} = \frac{\displaystyle\sum_{i=0}^{m}\sum_{j=0}^{m} \begin{bmatrix} 2\alpha_i\beta_j \\ \beta_i\beta_j - \alpha_i\alpha_j \end{bmatrix} N_{i,k}(u)N_{j,k}(u)}{\displaystyle\sum_{i=0}^{m}\sum_{j=0}^{m}(\alpha_i\alpha_j + \beta_i\beta_j)N_{i,k}(u)N_{j,k}(u)} \tag{28}$$

which makes it simple to evaluate the normal speed (9).

In each time step of the discretized evolution, we need to find the closest point. For instance, this can be formulated as a polynomial root–finding problem, since

$$\mathbf{c}'_{\mathbf{s}}(u)^{\top}(\mathbf{c}_{\mathbf{s}}(u) - \mathbf{p}_j) = 0 \tag{29}$$

is piecewise polynomial in u. For each $\mathbf{p}_j$ we can find all solutions of (29) and compare the distance of the closest one with the distance of $\mathbf{p}_i$ to the end-points.[4] Due to Proposition 1, the normal direction is well defined at all inner points of the curve.

The length of the PH spline has the particularly simple expression

$$L_{\mathbf{s}} = \int_{u_{k-1}}^{u_{m+1}} \left(\alpha(u)^2 + \beta(u)^2\right) \mathrm{d}u = \sum_{i=0}^{m}\sum_{j=0}^{m}(\alpha_i\alpha_j + \beta_i\beta_j)K_{i,j}(u_{m+1}). \tag{30}$$

Clearly, the $K_{i,j}(u_{m+1})$ are constant numbers which have to be computed only once, and $L_{\mathbf{s}(t)}$ is a quadratic function in the shape parameters with partial derivatives

$$\frac{\partial L_{\mathbf{s}}}{\partial \alpha_i} = 2\sum_{j=0}^{m}\alpha_j K_{i,j}(u) \quad \text{and} \quad \frac{\partial L_{\mathbf{s}}}{\partial \beta_i} = 2\sum_{j=0}^{m}\beta_j K_{i,j}(u). \tag{31}$$

The simple expression of the length of the PH spline inspired us to use the regularization term

$$R := \left(L_e - L_{\mathbf{s}} - \dot{L}_{\mathbf{s}}\right)^2, \tag{32}$$

which forces the curve length $L_{\mathbf{s}}$ to converge to some constant value L_e. We assume that a suitable value L_e is specified by the user or can be estimated from the data (e.g., using the length of the minimum spanning tree), if the level of noise is not too high.

5. Examples of PH splines evolution

We apply the procedure described in Section 4 to several examples. In all cases we use piecewise PH cubics defined by piecewise linear C^0 preimages. The resulting PH spline consists of polynomial pieces of degree 3 joined with C^1 continuity.

First, we present a simple example. In order to get a good PH approximation for more complicated data sets, it is necessary to choose a suitable initial position of the evolving curve. Here we suggest two strategies. The first one is based on subdivision, while the second one relies on Hermite interpolation.

[4] Note that the frequent closest point computation can be avoided during the first part of the evolution, when the curve is still relatively far from the data, see Remark 3.2.

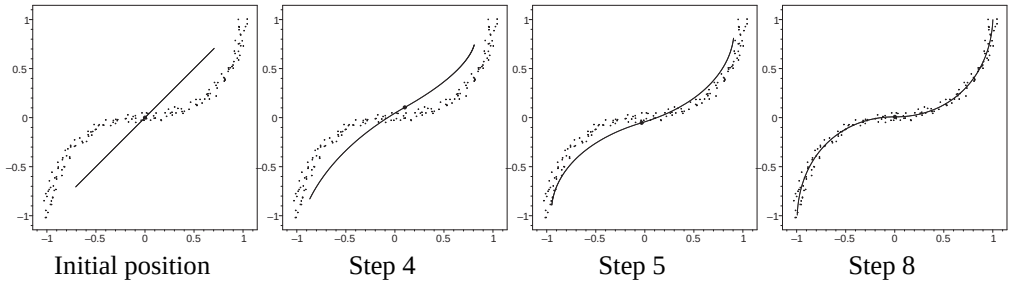

Fig. 3. Approximation of noisy data.

5.1. *Simple example*

In this example (see Figure 3), the input points were obtained from two circular arcs with radius 1. We added additional random noise to the sample points ranging from -0.05 to 0.05 in both x and y point coordinates. We evolved a PH spline composed of two cubic PH segments depending on 8 shape parameters: 3 for each of the piecewise linear preimage components u, v and 2 integration constants determining the position of the start point $\mathbf{c}_{\mathbf{s}(0)}(u)$ of the PH spline. In the initial position, the spline degenerates into a straight line.

Since the target shape is quite simple, no special adjustment of the evolution control values $\omega_\perp, \omega_{\mathbf{v}}, \omega_R$ and L_e is necessary. The length of the spline was estimated as $L_e = \pi$ and the regularization term (32) was kept unchanged during the whole evolution. Also, the weights occurring in (12) were all set to 1 and the maximal permitted change of the curve to 0.2 during the whole evolution. [5]

Figure 3 shows the evolution of the spline from its initial position towards a stationary solution, which is reached after 8 steps. The maximum error is then $6.02\ 10^{-2}$ which corresponds to the magnitude of the noise.

5.2. *Subdivision-based adaptation*

Let us consider a point set (see Figure 4) taken from a rather complicated free-form curve. In this case the evolution had to be controlled in a more sophisticated way.

The first strategy is as follows. We start with a PH spline which is in a rather poor initial position but, it consists only of a small number of cubic segments. Therefore, only few shape parameters s_i are involved, and the danger of an evolution towards a local minimum is reduced. After several evolution steps, we raise the number of spline segments (via knot insertion) without modifying the shape of the curve $\mathbf{c}(u)$. Then we continue the evolution until some stable situation is reached. This procedure can be repeated until the maximum error is sufficiently low.

In our example we started with a PH spline composed of two straight line segments. The maximal permitted change was again kept equal to 0.2 through the whole evolution. In order to match the global shape of the curve we started with a small imposed curve length $L_e = 8$ and with weights $\omega_\perp = \omega_{\mathbf{v}} = \omega_R = 1$. After step 30 the global shape of the curve is already well matched and the actual curve length is already 9.89.

[5] At each step the step-size $h \leq 1$ was estimated so that no point of the curve changes more then 0.2. When the curve is sufficiently close to a stationary point, then $h = 1$.

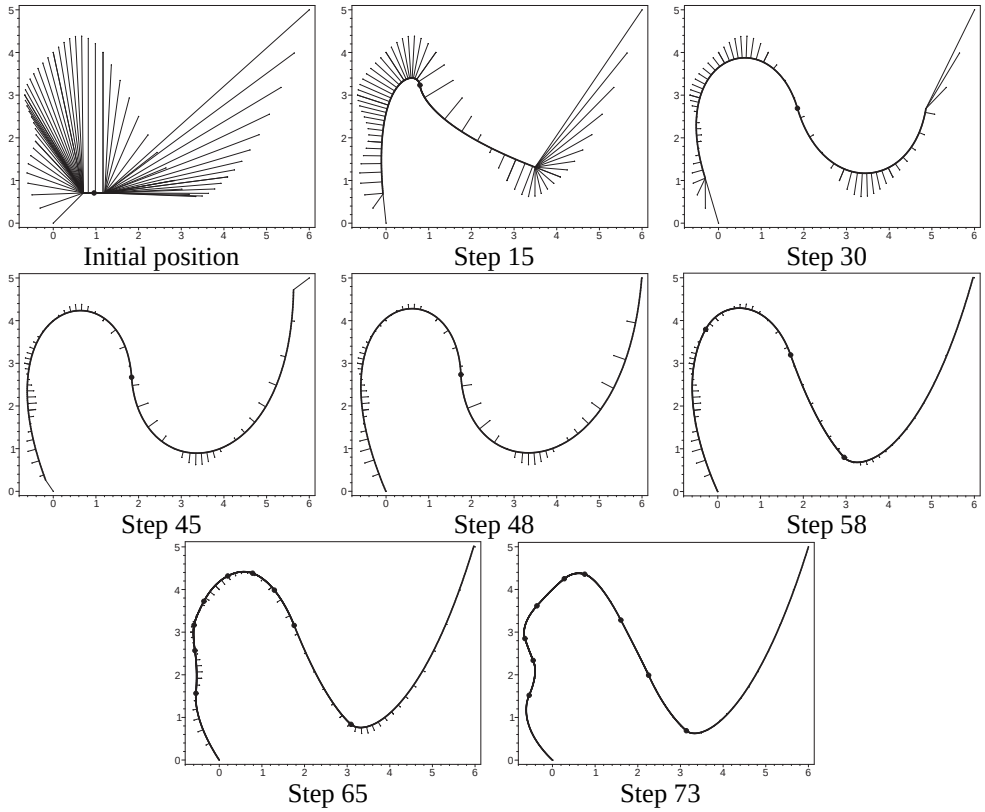

Fig. 4. Approximation of points taken from a spline curve.

Through steps 31 to 45 we gradually raised L_e length up to 14, the real length being at this moment only slightly greater. At this stage of the evolution it was necessary to fix the end points. For this purpose we relaxed the curve length condition by putting the weight w_R equal to 0.1 (while keeping the required length $L_e = 14$) and we set the end-point weight $\omega_{\mathbf{v}} = 100$. After only three steps the end points were fixed (see Step 48). At this moment the length was 15.5 and the maximum error 0.328.

Then we started the knot insertion. For a spline composed of 4 segments we reached a maximum error of 0.227 at step 58. Then we inserted 6 knots in the intervals where the error has been large (at the left part of the curve). The non-uniform spline composed of 10 parts converged at a stationary position in step 73. The length equals 15.3, and the maximum error is $1.63\ 10^{-2}$.

5.3. *Initial value by Hermite interpolation*

The initial position of the evolving curve can be also obtained directly from the original curve or from the given data points using Hermite interpolation.

First, we order the data and estimate tangent lines at each point, or sample them from the given curve. Since we want to use PH cubics, which cannot reproduce inflections, we split the data points into segments at estimated inflections. If necessary, each of the data segments is then split again until each segment has less than a predefined number of points and the variation of

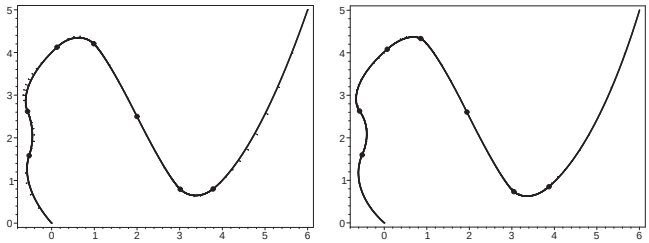

Fig. 5. Evolution with initial position constructed via G^1 Hermite interpolation.

the tangents is below a certain prescribed angle.

Now, for each segment we consider the G^1 boundary data (the end points and the estimated tangent vectors) and construct the cubic PH interpolants to these data following Meek and Walton (1997). All segments are then collected into a C^1 continuous spline, which serves as the initial position of the evolution. Note that the knot sequence within the interval $[0, 1]$ is determined by the required C^1 continuity is achieved.

We applied this procedure to the data from Section 5.2 - see Fig. 5. In the preprocessing step, the data points were split into 8 segments. For each segment, one PH cubic interpolating the end-point G^1 data was constructed, see Fig. 5, left. All 8 cubics were then represented by one PH spline defined over the knot sequence $(0, 0.23, 0.43, 0.61, 0.68, 0.73, 0.78, 0.85, 1)$. This is a "high quality" initial position, which leads to a fast convergence to a stationary position, after only 5 steps. The final error equals $2.45\ 10^{-2}$.

Clearly, other Hermite interpolation techniques Moon et al. (2001); Farouki et al. (1998a); Šír and Jüttler (2005); Šír et al. (2006); Jüttler (2001) can be used in order to produce higher order PH splines. In the case of a given curve, the boundary information can be sampled from it. For (possibly noisy) point sets, however, the estimation of the needed quantities (beyond first order of differentiation) may become more difficult, due to potential numerical instabilities.

6. Speed of convergence

We analyze the convergence speed of the PH spline evolution. More precisely, via comparing the evolution method with the Gauss-Newton method we show the quadratic convergence in the zero-residual case.

Lemma 4 *The Euler update of the shape parameters* $\mathbf{s}$ *for the curve evolution* (14) *with step size* h *is equivalent to a Gauss-Newton with the same step size* h *of the problem*

$$\sum_{j=1}^{N} \|\mathbf{p}_j - \mathbf{c}_\mathbf{s}(u_j)\|^2 \to \min_{\mathbf{s}} \quad \textit{where} \quad u_j = \arg\min_{u\in[a,b]} \|\mathbf{p}_j - \mathbf{c}_\mathbf{s}(u)\|, \tag{33}$$

provided that $\{a, b\} \cap \{u_j \mid j = 1, \dots, N\} = \emptyset$ [6] *and* $\omega_R = 0$.

PROOF. Recall that $d_j := (\mathbf{p}_j - \mathbf{c}_\mathbf{s}(u_j))^\top \mathbf{n}_\mathbf{s}(u_j)$, see (11). In order to solve

[6] This technical assumption ensures that none of the closest points appears at the boundary. It could be avoided by considering closed curves instead of open ones.

$$f = \sum_{j=1}^{N} d_j^2 = \sum_{j=1}^{N} \|\mathbf{p}_j - \mathbf{c}_\mathbf{s}(u_j)\|^2 \to \min_{\mathbf{s}} \quad \text{where} \quad u_j = \arg\min_{u\in[a,b]} \|\mathbf{p}_j - \mathbf{c}_\mathbf{s}(u)\|,$$

one may use a Gauss-Newton iteration. The new iterate $\mathbf{s}^+ = \mathbf{s} + h\Delta\mathbf{s}$ is found by solving

$$\sum_{j=1}^{N} [d_j + (\nabla d_j)^\top \Delta\mathbf{s}]^2 \to \min_{\Delta\mathbf{s}}. \tag{34}$$

In our case, the components [7] of the gradients ∇d_j are found from

$$2d_j[\nabla d_j]_i = [\nabla(d_j^2)]_i = [\nabla\|\mathbf{p}_j - \mathbf{c}_\mathbf{s}(u_j)\|^2]_i =$$
$$= -2\left(\frac{\partial \mathbf{c}_\mathbf{s}(u_j)}{\partial s_i} + \mathbf{c}_\mathbf{s}'(u_j)\frac{\partial u_j}{\partial s_i}\right)^\top (\mathbf{p}_j - \mathbf{c}_\mathbf{s}(u_j)) = -2\left(\frac{\partial \mathbf{c}_\mathbf{s}(u_j)}{\partial s_i}\right)^\top (\mathbf{p}_j - \mathbf{c}_\mathbf{s}(u_j)),$$

where we exploited the orthogonality of the tangent vectors $\mathbf{c}_\mathbf{s}'(u_j)$ at the closest points and the error vectors $\mathbf{p}_j - \mathbf{c}_\mathbf{s}(u_j)$. Hence,

$$[\nabla d_j]_i = -\left(\frac{\partial}{\partial s_i}\mathbf{c}_\mathbf{s}(u_j)\right)^\top \mathbf{n}_\mathbf{s}(u_j), \tag{35}$$

and Gauss-Newton reads as

$$\sum_{j=1}^{N}\left[(\mathbf{p}_j - \mathbf{c}_\mathbf{s}(u_j))^\top \mathbf{n}_\mathbf{s}(u_j) - \sum_{i=1}^{n}\left(\frac{\partial}{\partial s_i}\mathbf{c}_\mathbf{s}(u_j)^\top \mathbf{n}_\mathbf{s}(u_j)\Delta s_i\right)\right]^2 \to \min_{\Delta\mathbf{s}}. \tag{36}$$

Due to (9) and (11), the time derivatives $\dot{s}_i$ obtained from the optimization problem (12), which defines the evolution of the curve, are equal to the Gauss–Newton updates Δs_i obtained from (36) [8]. Hence, for stepsize $h = 1$, the Euler method for the evolution and the Gauss–Newton iteration for (33) are equivalent. □

Gauss–Newton methods exhibit quadratic convergence, provided that the residuum vanishes (i.e., all errors vanish for the final solution). Indeed, it can be seen as a Newton iteration, where the second part of the expansion

$$\nabla^2 f = \sum_{j=1}^{N} \nabla d_j (\nabla d_j)^\top + \sum_{j=1}^{N} d_j \nabla^2 d_j \tag{37}$$

of the Hessian has been omitted. If $d_j = 0$, then this part vanishes.

Example 5 In order to demonstrate the speed of convergence, we consider an example where the input points were taken from a PH spline, see Figure 6. The initial position of the evolution has been obtained by only slightly perturbing the coefficients of the input curve. Through the first five steps of the evolution, the curve evolved to a good approximant - see Table 1 for approximation errors at different evolution steps. For all remaining steps, the approximation error at any step is essentially a square of the error at the previous step, which demonstrates the quadratic convergence of the method.

[7] Here, $[\mathbf{v}]_i$ denotes the i–th component of a vector $\mathbf{v} = (v_1, \ldots v_n)^\top$

[8] The second and third term in (12) are not present, since no closest points at the curve boundaries were assumed to exist, and $\omega_R = 0$.

Table 1
Approximation errors during the evolution.

Step	Error	Step	Error	Step	Error	Step	Error	Step	Error
1	$1.02\ 10^{-1}$	3	$3.48\ 10^{-2}$	5	$5.52\ 10^{-3}$	7	$6.50\ 10^{-9}$	9	$1.10\ 10^{-30}$
2	$6.50\ 10^{-2}$	4	$1.67\ 10^{-2}$	6	$4.95\ 10^{-5}$	8	$1.37\ 10^{-16}$	10	$2.78\ 10^{-60}$

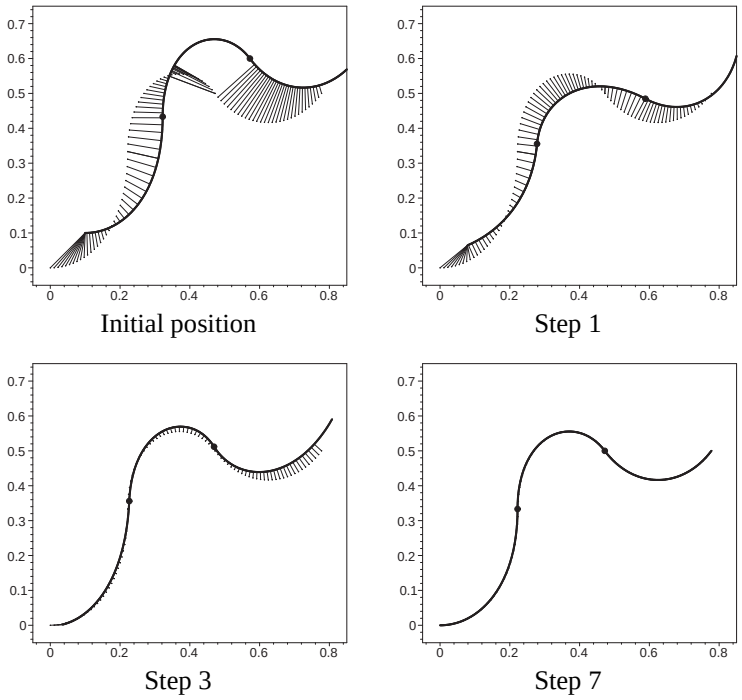

Fig. 6. Approximation of points taken from a PH spline.

7. Concluding remarks

We developed and analyzed an evolution–based fitting procedure for Pythagorean hodograph spline curves. It was shown that this problem can efficiently be dealt with, provided that a good initial solution is available. In this sense, least–squares fitting by PH spline curves is not necessarily more complicated than the same problem for standard curve representations. Indeed, the special properties of PH curves make it even easier to use certain geometrically motivated regularization terms, such as the length of the curve. Future research will be devoted to using the approximation procedure in order to obtain more compact representation of NC tool paths (currently often specified as G-code), where we will cooperate with one of our industrial partners, and on least–squares approximation by surfaces with rational offsets.

Acknowledgment. The second author was supported by the Austrian Science Fund (FWF) through project P17387–N12. The authors thank the reviewers for their valuable comments.

References

Aigner, M., Jüttler, B., 2006. Hybrid curve fitting. Computing, to appear, also available as FSP report no. 2 at `http://www.ig.jku.at`.

Aigner, M., Jüttler, B., 2007. Approximation Flows in Shape Manifolds, Curves and Surfaces (Avignon 2006), to appear, also available as FSP report at `http://www.ig.jku.at`.

Alhanaty, M., Bercovier, M., 2001. Curve and surface fitting and design by optimal control methods. Comp. Aided Design 33, 167–182.

Farouki, R. T., 2002. Pythagorean-hodograph curves. In: Handbook of computer aided geometric design. North-Holland, Amsterdam, 405–427.

Farouki, R. T., Kuspa, B. K., Manni, C., Sestini, A., 2001. Efficient solution of the complex quadratic tridiagonal system for C^2 PH quintic splines. Numer. Algorithms 27 (1), 35–60.

Farouki, R. T., Manjunathaiah, J., Jee, S., 1998a. Design of rational cam profiles with pythagorean-hodograph curves. Mech. and Mach. Theory 33 (6), 669–682.

Farouki, R. T., Saitou, K., Tsai, Y.-F., 1998b. Least-squares tool path approximation with Pythagorean-hodograph curves for high-speed CNC machining. In: The mathematics of surfaces, VIII (Birmingham, 1998). Info. Geom., Winchester, 245–264.

Hoff, K. E., Keyser, J., Lin, M., Manocha, D., Culver, T., 1999. Fast computation of generalized Voronoi diagrams using graphics hardware. In: SIGGRAPH '99. ACM Press/Addison-Wesley, New York, 277–286.

Hoschek, J., Lasser, D., 1993. Fundamentals of computer aided geometric design. A K Peters, Wellesley, MA.

Jüttler, B., 2001. Hermite interpolation by Pythagorean hodograph curves of degree seven. Math. Comp. 70 (235), 1089–1111.

Kass, M., Witkin, A., Terzopoulos, D., 1987. Snakes: active contour models. Int. J. Comp. Vision 1, 321–331.

Kubota, K., 1972. Pythagorean triplets in unique factorization domains. Amer. Math. Monthly 79, 503–505.

Meek, D. S., Walton, D. J., 1997. Geometric Hermite interpolation with Tschirnhausen cubics. J. Comput. Appl. Math. 81 (2), 299–309.

Moon, H. P., Farouki, R. T., Choi, H. I., 2001. Construction and shape analysis of PH quintic Hermite interpolants. Comp. Aided Geom. Design 18, 93–115.

Pottmann, H., Leopoldseder, S., 2003. A concept for parametric surface fitting which avoids the parametrization problem. Comp. Aided Geom. Design 20, 343–362.

Pottmann, H., Leopoldseder, S., Hofer, M., 2002. Approximation with active B-spline curves and surfaces. In: Proc. Pacific Graphics. IEEE Press, 8–25.

Pottmann, H., Leopoldseder, S., Hofer, M., Steiner, T., Wang, W., 2005. Industrial geometry: recent advances and appl. in CAD. Comp. Aided Design 37, 751–766.

Rogers, D., Fog, N., 1989. Constrained B-spline curve and surface fitting. Comp. Aided Design 21, 641–648.

Šír, Z., Feichtinger, R., Jüttler, B., 2006. Approximating curves and their offsets using biarcs and Pythagorean hodograph quintics. Comp. Aided Design 38, 608–618.

Šír, Z., Jüttler, B., 2005. Constructing acceleration continuous tool paths using Pythagorean hodograph curves. Mech. and Mach. Theory 40 (11), 1258–1272.

Speer, T., Kuppe, M., Hoschek, J., 1998. Global reparametrization for curve approximation. Comput. Aided Geom. Design 15, 869–877.

Wang, W., Pottmann, H., Liu, Y., 2006. Fitting B-spline curves to point clouds by squared distance minimization. ACM Transactions on Graphics 25 (2).