Algorithms for Rational Spline Curves

by

Klaus Höllig

Computer Sciences Technical Report #710
August 1987

UNIVERSITY OF WISCONSIN-MADISON
COMPUTER SCIENCES DEPARTMENT

Algorithms for Rational Spline Curves

Klaus Höllig[1]

Technical Report # 710
August 1987

ABSTRACT

The theory of univariate splines is well understood. However, applying the standard techniques for spline **functions** does not make use of some important features of piecewise polynomial **curves**. Since curves are invariant under reparametrization, the smoothness conditions for splines are less restrictive and standard approximation methods can be improved. This note discusses rational cubic spline curves and describes in particular two basic algorithms: the construction of smooth splines from control points and Hermite interpolation.

AMS (MOS) Subject Classifications: 41A15, 41A25, 53A04

Key Words: spline curves, Bézier form, geometric smoothness, control points, interpolation

[1] supported by the United States Army under Contract No. DAAG29-80-C-0041 and sponsored by the National Science Foundation under Grant No. DMS-8351187

Algorithms for Rational Spline Curves

Klaus Höllig[1]

ABSTRACT. The theory of univariate splines is well understood. However, applying the standard techniques for spline **functions** does not make use of some important features of piecewise polynomial **curves**. Since curves are invariant under reparametrization, the smoothness conditions for splines are less restrictive and standard approximation methods can be improved. This note discusses rational cubic spline curves and describes in particular two basic algorithms: the construction of smooth splines from control points and Hermite interpolation.

1. Introduction

We first review briefly two basic algorithms for "standard" cubic spline curves as a preparation for the generalizations to be discussed in the following sections. These algorithms are best described using the Bézier form for polynomials which allows a particularly simple characterization of smoothness constraints for splines. The Bézier coefficients b of a cubic polynomial p are defined by

$$p(t) = \sum_{\nu=0}^{3} b_\nu \, B_\nu(t), \quad 0 \le t \le 1,$$

where $B_\nu(t) := \binom{3}{\nu} t^\nu (1-t)^\nu$ are the Bernstein polynomials. Therefore, as is illustrated in Figure 1, a piecewise cubic spline curve p can be represented by a sequence of Bézier coefficients

$$b_\nu^j, \quad \nu = 0, \ldots, 3, \quad j = 0, \ldots, J.$$

It is assumed that $b_3^{j-1} = b_0^j$, i.e. that the curve segments join continuously. Continuity of the first and second derivatives of the parametrization is equivalent to the conditions

$$\begin{aligned} b_1^+ - b_0^+ &= b_3^- - b_2^- \\ (b_2^+ - b_1^+) - (b_1^+ - b_0^+) &= (b_3^- - b_2^-) - (b_2^- - b_1^-) \end{aligned}$$

where $b^\pm$ denote the Bézier coefficients of two adjacent curve segments. The particular form of these conditions yields a very simple algorithm for constructing the Bézier coefficients of twice continuously differentiable spline parametrizations from control points (cf. Figure 2).

---

[1] supported by the United States Army under Contract No. DAAG29-80-C-0041 and sponsored by the National Science Foundation under Grant No. DMS-8351187

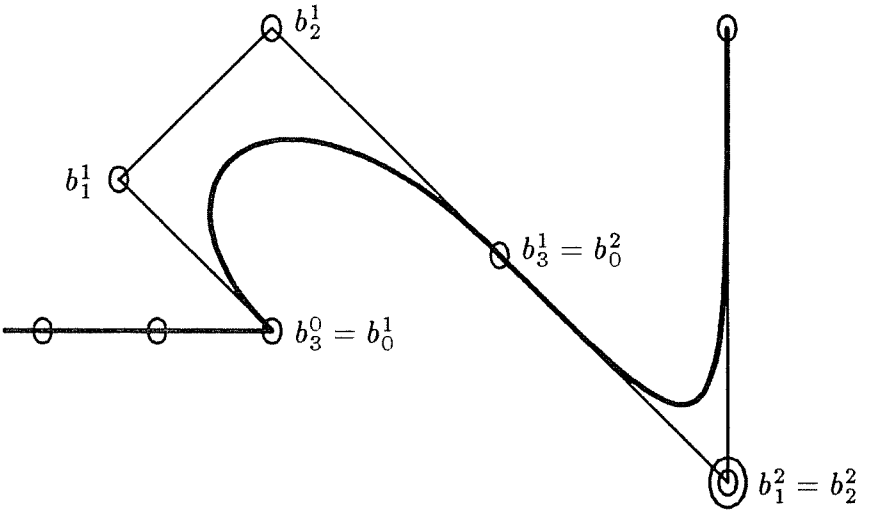

Figure 1. Bézier polygon of a cubic spline curve

Algorithm 1. $(c \Rightarrow b)$ The Bézier coefficients b corresponding to a sequence of control points c are given by

$$b_1^j := (2c^{j+1} + c^{j+2})/3, \quad b_2^{j-1} := (2c^{j+1} + c^j)/3$$
$$b_3^{j-1} = b_0^j := (b_2^{j-1} + b_1^j)/2.$$

Combining the steps in Algorithm 1, $f^j := b_3^{j-1} = b_0^j$ can be expressed in terms of the control points,

$$f^j = (c^j + 4c^{j+1} + c^{j+2})/6 \tag{1}$$

which yields

Algorithm 2. $(f \Rightarrow c)$ The control points c of the natural spline interpolant (which has zero curvature at the endpoints) corresponding to the data f^j, $j = 0, \ldots, J$, are computed by solving the linear system (1) for $j = 1, \ldots, J-1$ with the end conditions

$$c^1 = f^0, \quad c^0 = 2c^1 - c^2$$
$$c^{J+1} = f^J, \quad c^{J+2} = 2c^{J+1} - c^J.$$

Figure 3 shows an example which illustrates a slight disadvantage of the method: possible oscillations near inflection points.

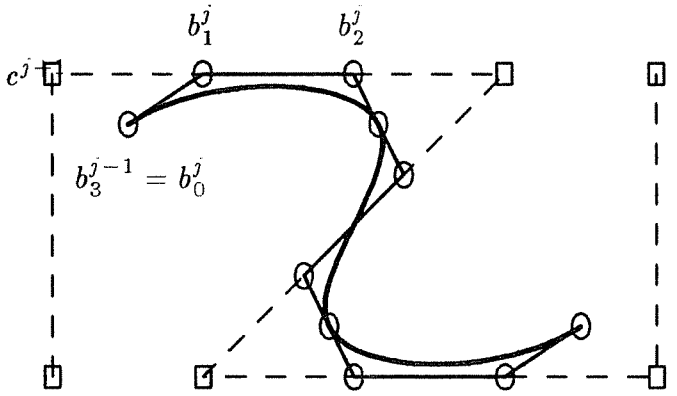

Figure 2. Control polygon, Bézier coefficients and corresponding spline curve

While straightforward, the above algorithms do not make use of the additional flexibility due to the weaker smoothness constraints for curves. This observation has led to the development of β-splines and interesting new approximation and design techniques (cf. [BBB80], [Bö85]). So far, the new geometric ideas have been primarily applied to polynomial splines. We discuss in this note the generalization of the basic algorithms to rational cubic splines. First, we describe the rational Bézier form in Section 2. Then, in Section 3 we discuss the analogues of Algorithms 1 and 2. Section 4 lists MACSYMA computations which establish the main result of this note.

2. Rational Bézier form

We review the definition and some basic facts about the Bézier form and refer to [FP79] for details. The Bézier form of a rational cubic parametrization r is defined as

$$r(t) = \frac{p}{q} = \frac{\sum_{\nu=0}^{3} w_\nu b_\nu \, B_\nu(t)}{\sum_{\nu=0}^{3} w_\nu \, B_\nu(t)}, \quad 0 \le t \le 1, \tag{2}$$

where the coefficients b_ν are vectors in $\mathbb{R}^3$ and the weights w_ν are positive numbers. The homogeneous form, i.e. multiplication with the weights in the numerator, simplifies the algebra and geometric interpretation. As in the polynomial case, the control polygon is tangent to the curve at the end points. The weights control the influence of the corresponding coefficients, i.e. increasing w_ν "pulls" the curve towards the coefficient b_ν as is illustrated in Figure 4.

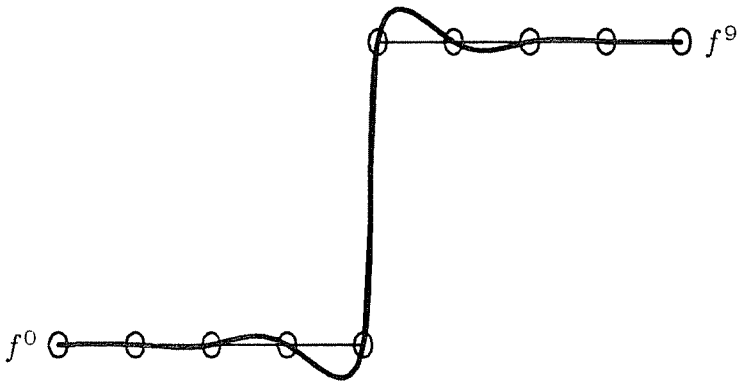

Figure 3. Natural spline interpolant

A piecewise rational curve is represented by a sequence of coefficients and weights,

$$(b_\nu^j, w_\nu^j), \quad \nu = 0, \ldots, 3, \quad j = 0, \ldots, J,$$

where, for continuity, $b_3^{j-1} = b_0^j$. To characterize higher order smoothness, we recall the definition of

Smoothness for Curves. Smoothness of a curve, $t \mapsto f(t) \in \mathbb{R}^3$, is characterized in terms of differentiability with respect to arclength $s := \int^t |f'|$. Since $dt/ds = 1/|f'|$ where $|\ |$ denotes the length of a vector, the first and second derivatives of f with respect to arclength are given by

$$\frac{d}{ds} f = f'/|f'|$$
$$\frac{d^2}{ds^2} f = (|f'|^2 f'' - (f' \cdot f'') f')/|f'|^4.$$

Taking the cross product of the second equation with $f'/|f'|$, this means that C^2-continuity is equivalent to continuity of the vectors

$$\xi_f := f'/|f'|, \quad (\kappa\eta)_f := f' \times f''/|f'|^3.$$

The vector ξ is the unit tangent vector, κ is the curvature and η is the binormal vector which is a unit vector orthogonal to the osculating plane.

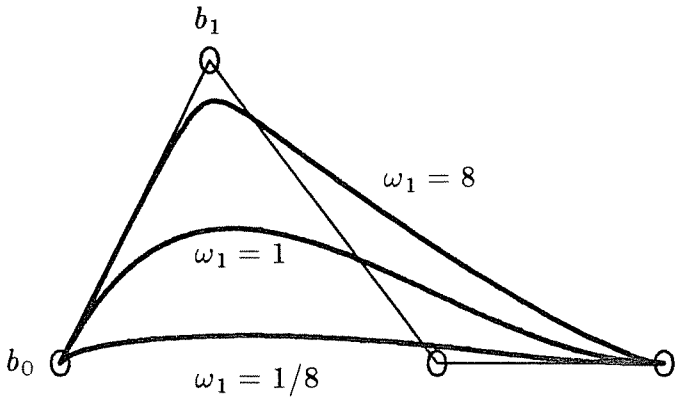

Figure 4. Rational Bézier form

Computing the vectors ξ and η for the parametrization (2) at the endpoints gives

$$\begin{aligned} \xi_r(0) &= \frac{b_1 - b_0}{|b_1 - b_0|}, \quad (\kappa\eta)_r(0) = \frac{2}{3}\,\frac{w_0 w_2}{w_1^2}\,\frac{(b_1 - b_0) \times (b_2 - b_1)}{|b_1 - b_0|^3} \\ \xi_r(1) &= \frac{b_3 - b_2}{|b_3 - b_2|}, \quad (\kappa\eta)_r(1) = \frac{2}{3}\,\frac{w_1 w_3}{w_2^2}\,\frac{(b_3 - b_2) \times (b_1 - b_2)}{|b_3 - b_2|^3}. \end{aligned} \tag{3}$$

Therefore, two adjacent curve segments with Bézier coefficients $b^{\pm}$ and weights $w^{\pm}$ join twice continuously differentiable at $b_3^- = b_0^+$ if the following two conditions are satisfied:

(C1) b_2^-, $b_3^- = b_0^+$, b_1^+ are collinear;

(C2) b_1^-, b_2^-, $b_3^- = b_0^+$, b_1^+, b_2^+ lie in a half plane and the parallelograms $R_{\pm}$ in Figure 5 satisfy

$$\frac{w_1^- w_3^-}{(w_2^-)^2}\,\frac{\text{area}(R_-)}{|b_3^- - b_2^-|^3} = \frac{w_0^+ w_2^+}{(w_1^+)^2}\,\frac{\text{area}(R_+)}{|b_1^+ - b_0^+|^3}. \tag{4}$$

3. Control points and interpolation

The geometric description of the smoothness conditions easily yields the analogue of Algorithm 1 which is a variant of the corresponding method in the polynomial case [Bö85]. Denote by $\beta_{\pm}$ and $\delta_{\pm}$ the relative length of adjacent line segments as is indicated in Figure 5. For example,

$$|b_1^+ - b_0^+| : |b_3^- - b_2^-| = \delta_+ : \delta_-.$$

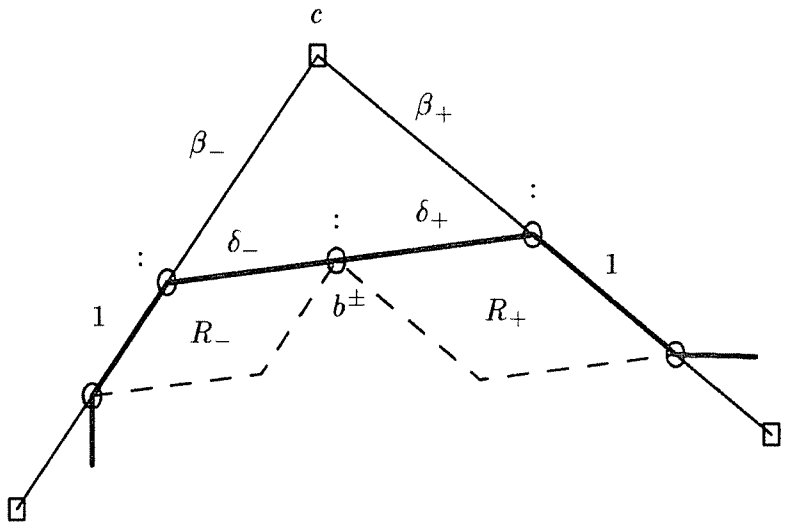

Figure 5. Geometric smoothness constraints

Then, since $\text{area}(R_+) : \text{area}(R_-) = (\delta_+\beta_-) : (\delta_-\beta_+)$, condition (4) becomes

$$\delta^2 := (\delta_+/\delta_-)^2 = (\beta_-/\beta_+)\ \frac{w_0^+ w_2^+}{(w_1^+)^2}\ \frac{(w_2^-)^2}{w_1^- w_3^-}. \tag{5}$$

This means, one can select the points b_1^-, b_2^-, b_1^+, b_2^+ and the weights $w^{\pm}$ essentially arbitrarily and it is then possible to select δ, and hence $b_3^- = b_0^+$, so that the smoothness conditions (C) are satisfied. This yields the following algorithm.

Algorithm I. $(c, w, \beta \Rightarrow b)$ The Bézier coefficients b_ν^j of a piecewise rational spline curve corresponding to the sequence of control points c^j, weights w_ν^j and parameters $\beta_\pm^j > 0$ are given by

$$\begin{aligned}
b_1^j &:= ((1+\beta_-^{j+2})c^{j+1} + \beta_+^{j+1}c^{j+2})/(1+\beta_+^{j+1}+\beta_-^{j+2}) \\
b_2^{j-1} &:= ((1+\beta_+^j)c^{j+1} + \beta_-^{j+1}c^j)/(1+\beta_-^{j+1}+\beta_+^j) \\
b_3^{j-1} = b_0^j &:= (\delta^{j+1}b_2^{j-1} + b_1^j)/(1+\delta^{j+1})
\end{aligned}$$

where δ^{j+1} is defined in terms of w^{j-1}, w^j, $\beta_\pm^{j+1}$ according to (5).

Algorithm 1 is a special case corresponding to $w_\nu^j = \beta_\nu^j = 1$. The weights w and the parameters β permit local control of the "shape" of the curve while keeping the control points fixed. This is illustrated in Figure 6. Decreasing β increases the curvature at the knots and the curve approaches the "control polygon" which connects the points c^j. Increasing a particular weight stresses the influence of the corresponding Bézier coefficient. If this additional flexibility is not needed, the parameters can be set according to suitable optimality criteria.

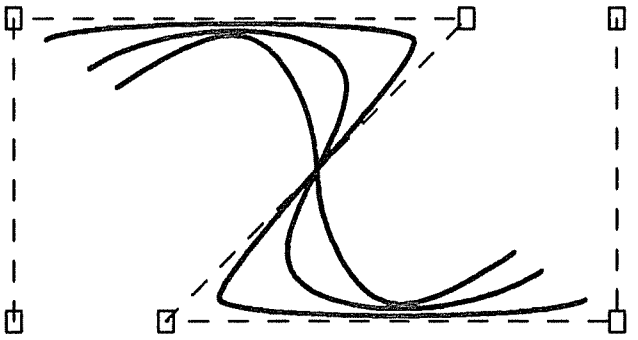

Figure 6. Control points and corresponding rational spline curve for $\beta = 1/4,\ 1,\ 4$

Algorithm 2 for interpolation requires the solution of a linear system, i.e. changes in the data have a global influence. Using the additional degrees of freedom due to the weaker smoothness constraints, it is possible to construct smooth interpolants by a local method. This method is suggested by the expressions (3) for ξ and η. Setting $\delta_i := |b_{1+2i} - b_{2i}|$, $f^i := r(i)$, $\xi^i := \xi_r(i)$, $(\kappa\eta)^i := (\kappa\eta)_r(i)$ for $i = 0, 1$ and substituting

$$b_2 - b_1 = (f^1 - f^0) - \delta_0\xi^0 - \delta_1\xi^1,$$

the equations for $\kappa\eta$ in (3) can be rewritten as

$$(\kappa\eta)^i = (-)^i\varrho_i\xi^i \times (f^1 - f^0) \ + \ \sigma_i\xi^1 \times \xi^0, \quad i = 0, 1, \tag{E}$$

where

$$\varrho_i := \frac{2}{3}\ \frac{w_i w_{2+i}}{w_{1+i}^2}\frac{1}{\delta_i^2}, \quad \sigma_i := \varrho_i\delta_{1-i}. \tag{6}$$

Since both sides of the i-th equation are orthogonal to ξ^i, the equations (E) are equivalent to a 4×4 linear system for ϱ and σ. This system has a solution with $\varrho, \sigma > 0$ if

(A) η^i lies in the interior of the cone spanned by
$$(-)^i\xi^i \times (f^1 - f^0) \text{ and } \xi^1 \times \xi^0, \quad i = 0, 1.$$

Choosing $w_1 := w_2 := 1$, the remaining weights w_0, w_3 and the parameters δ can be expressed in terms of ϱ and σ,

$$\delta_{1-i} = \sigma_i/\varrho_i, \quad w_{3i} = (3/2)\delta_i^2\varrho_i. \tag{7}$$

The corresponding method described in Algorithm II is a generalization of Hermite interpolation.

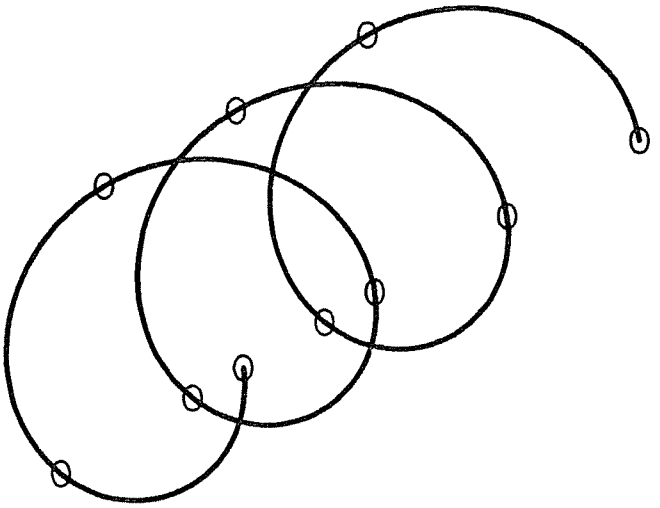

Figure 7. Rational spline interpolant of a helix

Algorithm II. $(f,\ \xi,\ \kappa\eta \Rightarrow b,\ w)$ The Bézier coefficients b_ν and weights w_0, w_3 of the j-th segment of a piecewise cubic rational spline r_f which matches the unit tangent vectors ξ^j and the vectors $(\kappa\eta)^j$ at the points f^j can be determined by solving the system (E) with

$$f^i := f^{j+i}, \quad \xi^i := \xi^{j+i}, \quad \eta^i := \eta^{j+i}$$

provided that condition (A) holds.

The following Theorem shows that, for data corresponding to a smooth curve, condition (A) is satisfied if the interpolation points f^j are sufficiently close. Moreover, the interpolant is of high order accuracy and has good shape preserving properties for smooth data. This is illustrated in Figure 7. If condition (A) is not valid for a particular curve segment, then the given curvatures and binormals cannot be interpolated and have to be modified or possibly chosen differently for adjacent segments. An interesting [open] problem is whether for given points f the vectors ξ and $\kappa\eta$ can be chosen so that (A) is valid and the resulting scheme remains accurate and shape preserving.

Theorem. Assume that the data f^j, ξ^j, η^j in Algorithm II correspond to a smooth curve f with nonvanishing curvature κ and torsion τ (cf. [FP79, p. 102] for definitions). If the distance $h := \max|f^j - f^{j-1}|$ between adjacent points is sufficiently small, then for each pair of adjacent points, condition (A) is satisfied and hence the system (E) has a unique solution with $\varrho, \sigma > 0$. Moreover, the corresponding piecewise rational interpolant r_f is 6-th order accurate, i.e.

$$\mathrm{dist}(f, r_f) = O(h^6).$$

For planar curves, a similar result was obtained in [BHS87] for interpolation with piecewise cubic polynomials. For the rational case, the proof is somewhat simpler, since the weights w and the parameters δ can be expressed explicitly in terms of the data.

Proof. Consider a typical curve segment of the interpolant, e.g. corresponding to the end points f^0, f^1. Without loss we assume that f is parametrized with respect to arclength s and that

$$f^0 = [\,0 \quad 0 \quad 0\,], \quad \xi^0 = [\,1 \quad 0 \quad 0\,], \quad \eta^0 = [\,0 \quad 0 \quad 1\,], \tag{8}$$

and $f^1 := f(s)$. Denote by $r(t,s) = p(t,s)/q(t,s)$, $0 \le t \le 1$, the rational interpolant. We will show that

(i) for sufficiently small s, the system (E) has a unique solution with $\varrho, \sigma > 0$;

(ii) $q(t,0) = 1$;

(iii) $\Big(\partial_t r_1(t,s)/s\Big)_{|s=0} = 1$;

(iv) $\partial_t^i[p(t,s),\, q(t,s)] = O(s^i), \quad i = 1,2,3.$

Assertions (i) and (ii) guarantee that r is well defined for small s, i.e. as the distance of the points f^0 and f^1 becomes small. Assertion (iii) implies that the derivative of the first coordinate $x = r_1$ of r satisfies

$$c_0 s \le \partial_t r_1(t,s) \le c_1 s \tag{9}$$

for some constants c_ν and s sufficiently small. In particular, the function r_1 is monotone increasing in t. With $\check{r}_1$ denoting its inverse, i.e. $x = r_1(\check{r}_1(x,s),s)$, the rational interpolant has the equivalent parametrization

$$x \mapsto R(x,s) := [\,x \quad r_2(\check{r}_1(x,s),s) \quad r_3(\check{r}_1(x,s),s)\,].$$

Similarly, f can be parametrized with respect to the first coordinate,

$$x \mapsto F(x).$$

Since the interpolation conditions are invariant under reparametrization, the unit tangent, curvature and binormal of R and F match at $x_0 := 0$ and $x_1 := r_1(1,s) = f_1(s)$. Using that $R_1(x) = F_1(x) = x$, this implies that the derivatives of R and F match at these points up to second order. From the standard error estimate for interpolation of functions it follows that

$$|R(x,s) - F(x)| = O((x_1 - x_0)^6) = O(s^6)$$

provided that the derivatives of R with respect to x up to order 6 are bounded, uniformly in s. This follows from (iv). To see this we note that

$$R_\nu(r_1(t,s),s) = r_\nu(t,s), \quad x = r_1(t,s),$$

and compute the derivatives of R_ν inductively using the chain rule. This shows that $\partial_x^j R_\nu(x,s)$ is a sum of terms of the form

$$r_\nu^{(k)} r_1^{(\ell_1)} \cdot \ldots \cdot r_1^{(\ell_m)} / (r_1^{(1)})^{k+\ell_1+\cdots+\ell_m}$$

where superscripts denote differentiation with respect to t and all functions are evaluated at (t,s) with $t = \check{r}_1(x,s)$. Since $r^{(j)}$ is a sum of terms of the form

$$p^{(k)} q^{(\ell_1)} \cdot \ldots \cdot q^{(\ell_m)} / q^{m+1}, \quad j = k + \ell_1 + \cdots + \ell_m,$$

the boundedness of R_ν is a consequence of (ii), (iv) and (9).

It remains to verify assertions (i)-(iv). This requires elaborate Taylor expansions which are done via MACSYMA as is described in the final section.

4. MACSYMA computations

Below we list a MACSYMA program for proving (i)-(iv) of the previous section. The computation is divided into four main steps: Taylor expansion of the data; solution of equations (E); Bézier form of r; verification of (i)-(iv). To speed up the computations, we use Taylor expansion to simplify intermediate results. The order of truncation will be justified at the end of this section.

Auxiliary functions. The following auxiliary functions will be used in the program: (c1) is de Casteljau's algorithm for evaluating a polynomial at t from its Bézier coefficients b; (c2) is the vector product; (c3) generates the first $n+1$ terms of a power series with coefficients a_i; (c4) computes the Taylor expansion of the solution of the differential equation $x'(t) = f(x(t))$, $x(0) = x0$.

```
(c1) bezier_form(b,n,t) :=
        if n=0 then row(b,1)
        else t*bezier_form(submatrix(1,b),n-1,t)
                + (1-t)*bezier_form(submatrix(n+1,b),n-1,t)$

(c2) cross_product(a,b) :=
        [a[2]*b[3]-a[3]*b[2], a[3]*b[1]-a[1]*b[3], a[1]*b[2]-a[2]*b[1]]$

(c3) power_series(a,n,t) :=
        if n=0 then a[0]
        else power_series(a,n-1,t) + a[n]*t^n/factorial(n)$
```

```
(c4) solve_ode(f,x0,n,t) :=
        if n=0 then x0
        else (x1: solve_ode(f,x0,n-1,t),
                  x1 +  subst(0,t,diff(f(x1),t,n-1))*t^n/factorial(n))$
```

Taylor expansion of the data. The data at the left endpoint are given by (8). To obtain Taylor expansions for $f^1 = f(s)$, $\xi^1 = \xi(s)$ and $(\kappa\eta)^1 = \kappa(s)\eta(s)$, (c10) approximately solves the Frenet differential equations [FP79, p.103],

$$\begin{aligned} f' &= \xi \\ \xi' &= \kappa\zeta \\ \zeta' &= \tau\eta - \kappa\xi \\ \eta' &= -\tau\zeta, \end{aligned}$$

where ζ with $\zeta(0) := [\,0 \quad 1 \quad 1\,]$ is the normal vector. This yields Taylor expansions for the data in terms of the Taylor coefficients u_i and v_i of curvature and torsion (cf. (c6), (c7)).

```
(c5) (kappa(s) := power_series(u,5,s), tau(s) := power_series(v,5,s))$
```

```
(c6) kappa(s);
```
(d6) $\quad u_5 s^5/120 + u_4 s^4/24 + u_3 s^3/6 + u_2 s^2/2 + u_1 s + u_0$

```
(c7) tau(s);
```
(d7) $\quad v_5 s^5/120 + v_4 s^4/24 + v_3 s^3/6 + v_2 s^2/2 + v_1 s + v_0$

```
(c8) (f[0]: [0,0,0], xi[0]: [1,0,0], zeta[0]: [0,1,0], eta[0]: [0,0,1])$

(c9) g(a) := [a[2], kappa(s)*a[3], tau(s)*a[4]-kappa(s)*a[2], -tau(s)*a[3]]$

(c10) ffs: solve_ode(g,[f[0], xi[0], zeta[0], eta[0]],6,s)$

(c11) (f[1]: ffs[1], xi[1]: ffs[2], zeta[1]: ffs[3], eta[1]: ffs[4])$

(c12) (kappa[0]: kappa(0), kappa[1]: kappa(s))$
```

Solution of equations (E). All vectors in the i-th equation in (E) are orthogonal to ξ^i and therefore the i-th equation is equivalent to the 2×2 system

$$eqn^i_{\nu,1} = eqn^i_{\nu,2}\,\varrho_i + eqn^i_{\nu,3}\,\sigma_i, \quad \nu = 0, 1, \tag{10}$$

obtained in (c13) by forming the dot product of the i-th equation with the vectors η^i and ξ^{1-i}. (c14) solves the system (10) by backward substitution, using that $eqn^i_{3,3} = 0$. The parameters δ and weights w are computed in (c15) and (c18) according to (7).

```
(c13) for i:0 thru 1 do
          eqn[i]: (matrix1: matrix(eta[i],xi[1-i]),
                   matrix2: matrix(kappa[i]*eta[i],
                                   (-1)^i*cross_product(xi[i],f[1]-f[0]),
                                   cross_product(xi[1],xi[0])),
                   ratsimp(taylor(matrix1.transpose(matrix2),s,0,6)))$

(c14) for i:0 thru 1 do
          (rho[i]: eqn[i][2,1]/eqn[i][2,2],
          sigma[i]: (eqn[i][1,1]-eqn[i][1,2]*rho[i])/eqn[i][1,3])$

(c15) for i:0 thru 1 do
          delta[1-i]: taylor(ratsimp(sigma[i]/rho[i]),s,0,3)$

(c16) delta[0];
(d16)      s/3 + (u0 v1 + 2 v0 u1) s^2/(36 u0 v0) + (9 u0^2 v0 v2
                 + 12 u0 v0^2 u2 - 10 u0^2 v1^2 + 2 u0 v0 u1 v1
                 - 10 v0^2 u1^2 + 6 u0^2 v0^4 + 6 u0^4 v0^2) s^3/(540 u0^2 v0^2)

(c17) delta[1]-delta[0];
(d17)      -(u0 v1 + 2 v0 u1) s^2/(18 u0 v0) - (u0^2 v0 v2 + 2 u0 v0^2 u2
                 - u0^2 v1^2 - 2 v0^2 u1^2) s^3/(36 u0^2 v0^2)

(c18) for i:0 thru 1 do
          w[i]: taylor(ratsimp((3/2)*rho[i]*delta[i]^2),s,0,2)$

(c19) w[0];
(d19)      1 + (24 u0^2 v0 v2 + 12 u0 v0^2 u2 - 35 u0^2 v1^2 - 8 u0 v0 u1 v1
                 - 20 v0^2 u1^2 + 36 u0^2 v0^4 + 36 u0^4 v0^2) s^2/(720 u0^2 v0^2)

(c20) w[1]-w[0];
(d20)      0
```

Bézier form of r. Statements (c21) and (c22) define the polynomials $p(\cdot, s)$ and $q(\cdot, s)$ using de Casteljau's Algorithms in terms of the Bézier coefficients.

```
(c21) p: (wb: matrix(w[0]*f[0], f[0]+delta[0]*xi[0],
                     f[1]-delta[1]*xi[1], w[1]*f[1]),
          ratsimp(taylor(bezier_form(wb,3,t)[1],s,0,2)))$

(c22) q: (w: matrix([w[0]], [1], [1], [w[1]]),
          ratsimp(taylor(bezier_form(w,3,t)[1,1],s,0,2)))$
```

Verification of (i)-(iv). As is shown by (c23)–(c26), the dominant part of the system (10), as $s \to 0$ is given by

$$\begin{bmatrix} u_0 \\ u_0^2 v_0 s^2/2 \end{bmatrix} = \begin{bmatrix} u_0 s^2/2 & -u_0 s \\ u_0^2 v_0 s^4/12 & 0 \end{bmatrix} \begin{bmatrix} \varrho_i \\ \sigma_i \end{bmatrix} \tag{11}$$

which proves (i). Clearly, (c27) proves (ii). (c28) computes the numerator of $\partial_t r_1(t,s)/s$, where $r_1 = p_1/q$, and evaluates it at $s = 0$. In conjunction with (ii) this establishes (iii). Assertion (iv) is equivalent to the statement that

$$\partial_t^i \partial_s^j [p,q](t,s) = 0, \quad j < i \leq 3.$$

This is checked by (c29) which displays $(\partial_t^i [p(t,s), q(t,s)])/s^{i-1}$ evaluated at $s = 0$.

```
(c23) taylor(eqn[0][1],s,0,2);
(d23)       [u_0, u_0 s^2/2, -u_0 s - u_1 s^2/2]

(c24) taylor(eqn[1][1]-eqn[0][1],s,0,2));
(d24) [u_1 s + u_2 s^2/2, 0, 0]

(c25) taylor(eqn[0][2],s,0,4);
(d25)       [v_0 u_0^2 s^2/2 + (v_1 u_0^2 + 2u_1 v_0 u_0) s^3/6 - (v_0 u_0^4 + (v_0^3 - v_2) u_0^2
                    -(3u_2 v_0 + 3v_1 u_1) u_0) s^4/24, v_0 u_0^2 s^4/12, 0]

(c26) taylor(eqn[1][2]-eqn[0][2],s,0,4);
(d26)       [(u_0^2 v_1 + 2u_0 v_0 u_1) s^3/6 + (u_0^2 v_2 + 2u_0 v_0 u_2
                    +4u_0 u_1 v_1 + 2v_0 u_1^2) s^4/12, 0, 0]

(c27) subst(0,s,q);
(d27)       1

(c28) subst(0,s,ratsimp((diff(p[1],t)*q-p[1]*diff(q,t))/s));
(d28)       1
(c29) for i:1 thru 3 do
          disp(subst(0,s,ratsimp(diff([p,q],t,i)/s^(i-1))));
                    [[0, 0, 0], 0]
                    [[0, 0, 0], 0]
                    [[0, 0, 0], 0]
(d29)       done
```

It remains to justify the various orders of truncation in the intermediate Taylor expansions. Since multiplication never decreases the order of validity of truncated Taylor

expansions, we must only consider (c15) and (c18). To indicate the range of significant terms in an expansion

$$\varphi(s) = \varphi_j s^j + \varphi_{j+1} s^{j+1} + \ldots,$$

we use the notation

$$\varphi \sim [j, J]$$

if the coefficients up to index J agree with the exact expansion of φ. By (11), and since the data are computed exact up to order 6 by (c10), the coefficients in the system (10) satisfy

$$eqn^i \sim \begin{bmatrix} [0,6] & [2,6] & [1,6] \\ [2,6] & [4,6] & 0 \end{bmatrix}.$$

This shows that

$$\begin{aligned} \varrho &\sim [2,6]/[4,6] \sim [-2,0] \\ \sigma &\sim ([0,6] - [2,6] * [-2,0])/[1,6] \sim [-1,1] \\ \delta &\sim [-1,1]/[-2,0] \sim [1,3] \\ w &\sim [-2,0] * [1,3]^2 \sim [0,2] \end{aligned}$$

and hence all subsequent expansions are exact at least up to order 2 which is what is needed for the proof of (iv).

References

[BBB85] B. Barsky, R.H. Bartels and J.C. Beatty, An introduction to the use of spline functions in computer graphics, ACM Siggraph, San Francisco, 1985.

[Bö85] W. Böhm, Curvature splines, Computer-Aided Geometric Design **2** (1985), 313–324.

[BHS87] C. de Boor, K. Höllig and M. Sabin, High accuracy geometric Hermite interpolation, Proc. Conf. on CAD, G. Farin ed., Oberwolfach 1987.

[FP79] I.D. Faux and M.J. Pratt, Computational Geometry for Design and Manufacture, Ellis Horwood, 1979.