DAMTP/1996/NA17

Pseudo-Quadratic Rational Speed Splines.

(a longer version of a paper submitted to AFA curves and surfaces conference,

Chamonix, June 1996)

Malcolm Sabin

Abstract Farouki has shown that piecewise rational-speed curves of order $2n-1$ can be constructed with the same number of freedoms as polynomial curves of order n.

This paper explores the building of cubic rational speed pieces into curves which mimic the behaviour of quadratic B-splines, but which have the rational speed property.

The new results show that it is possible stably to construct a composite curve in two dimensions which has continuity of magnitude of curvature at the knots.

Department of Applied Mathematics and Theoretical Physics,
University of Cambridge,
Silver Street,
Cambridge CB3-9EW,
England.

September 1996.

Pseudo-Quadratic Rational Speed Splines

Malcolm Sabin

March 1996

Abstract One of the major discoveries in curve theory recently is that the rational speed curves explored by Farouki [] tend to have excellent aesthetic properties.

Farouki showed that a particular subset of parametric Bezier curves had the property that the algebraic expression for the first derivative of the curve with respect to parameter had a polynomial magnitude, so that the speed (rate of change of arc length with parameter) was always rational. Most parametric polynomial curves have irrational speeds, requiring a square root.

This subset can be offset exactly, and can also have closed forms for the length and area integrals.

What was not expected was that they would also be among the best looking curves, but this was also discovered by Farouki [IMA6].

This paper elaborates on the possibility of curves with a similar level of control to B-splines being built from rational speed pieces, exploring, analysing and comparing four useful variants.

Contents

1 Pseudo-Quadratic Rational Speed Theory

A pseudo-quadratic is a cubic parametric polynomial satisfying the rational speed constraint, and constrained to turn through less than π. Its Bezier polygon has the property that its three legs have lengths in geometric progression and the two angles of the polygon are equal.

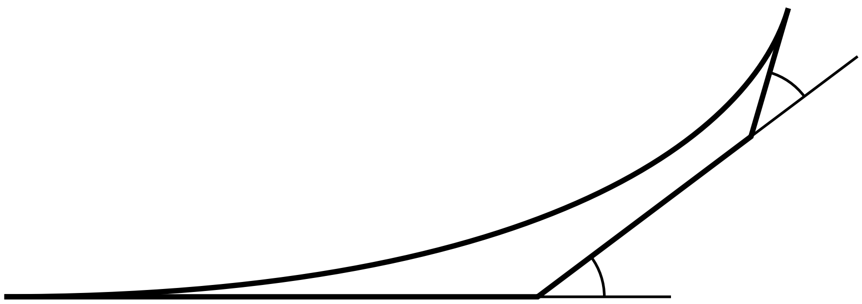


FIG 1

Because the two angles are equal in sense as well as magnitude, and because the total angle turned through is limited, the configuration must be essentially that of FIG 2, with the two end legs projected meeting in a point which we can use as a control point.

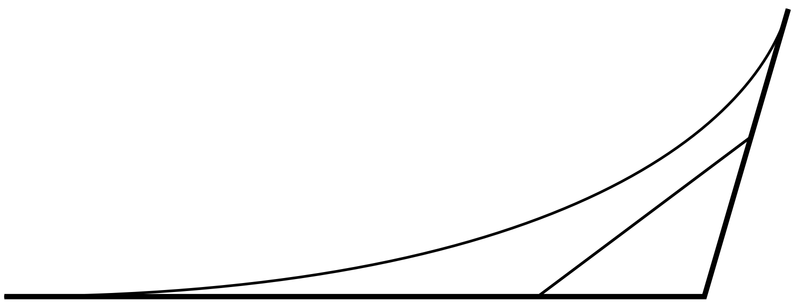


FIG 2

We can concatenate pseudo-quadratic pieces with continuity of tangent direction to form a piecewise curve, and then control the curve by locating the intersection points.

The construction process from a given composite control polygon has two stages. The first is to decide where on each side the two pieces meet. This has a degree of freedom, and the main issue in this paper is discussion of the options for fixing it.

The second stage is to decide how much of each corner is to be cut off to form the Bezier cubic control polygon. This is determined by the rational speed condition.

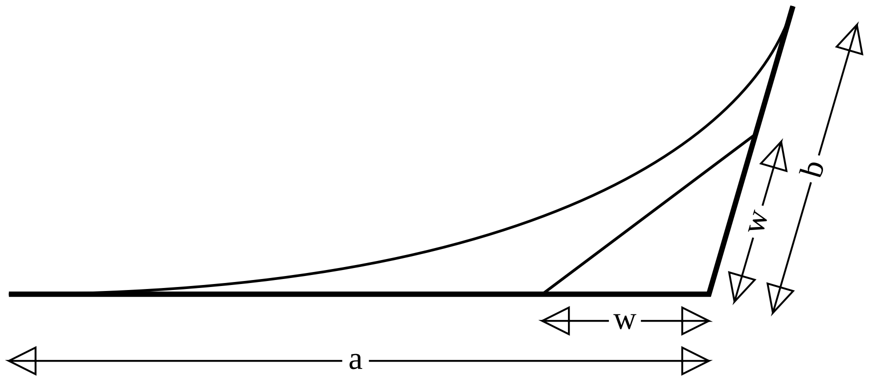


FIG 3

In fig 3 let a and b be the lengths of the legs from the vertex to the two change-points. θ is half of the angle turned through by the polygon at the vertex being considered. Let w be the distance along each leg that the cutting is done to. Because the two angles between the new cut and the original legs are equal (both are equal to θ), the distances along the two legs are the same. Then the rational speed condition gives a quadratic equation for w

$$(a-w)(b-w) = 4w^2\cos^2\theta$$

The closed form solution for this is well conditioned in the reciprocal form

$$w = \frac{2ab}{a+b+\sqrt{(a+b)^2+4ab(2\cos 2\theta\ +1)}}$$

especially if this is further transformed to

$$w = \frac{2ab}{a+b+\sqrt{(a-b)^2+8ab(\cos 2\theta\ +1)}}$$

It turns out that the positioning of each change point depends on the cutting of the two corners which bracket it, while the corner cutting obviously depends on the change point positions.

We thus have two coupled sets of nonlinear equations to solve.

Although the fastest solution takes the complete set, for exploration and experimentation purposes, it is sufficient merely to alternate the computing of change-points with that of corner cutting.

2 Options for the change-points

We consider four options for change-point positioning.

- fixed positions
- fixed parametric ratios
- C2 condition
- near to C2

2.1 Fixed positions

This is the only case where iteration is unnecessary. Each corner is isolated from the next and can be computed independently. Subvariants include always using the midpoint, and dividing the edge in the ratio of the angles at its ends.

The operational advantage of the midpoint variant is that the finite support property of B-splines is retained. Any change to the polygon causes change only in a finite part of the curve.

A fixed division depending on the local angles also gives a finite support, but with a width of five spans rather than three.

2.2 Fixed parametric ratios

Once two adjacent bezier pieces are set up, we may determine the relative knot spacings in the equivalent B-spline by looking at the lengths of the local legs $a-w$ and $b-w$ adjacent to the changepoint.

The ratio of parametric intervals for the two spans is just the ratio of these two lengths.

Thus a required set of parametric ratios (equal interval, for example) can be achieved by positioning the changepoint to give that ratio of legs.

2.3 C2 condition

If the changepoint is positioned to be very close to one end of the available range, the curvature of the curve on the shorter side can be made to exceed any specified threshold, while the curvature on the other side drops to a finite limit.

We can therefore use the degree of freedom to equate the magnitudes of the curvatures on the two sides of the changepoints.

If the polygon does not have an inflection in that leg, this will give continuity of curvature of the curve.

Let a, b be the distances from the left and right ends of the edge to the change point, so that $a + b = l$ the length of the edge.

Let w_l and w_r be the corner cutting distances cut off along the edge.

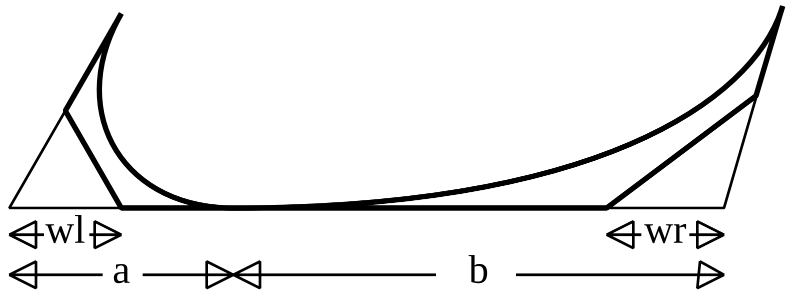


FIG 5

Let s_l and s_r be the magnitudes of the sines of the polygon turning angles at left and right ends.

Then for continuity of magnitude of curvature, we have

$$\frac{(a - w_l)^2}{s_l w_l} = \frac{(b - w_r)^2}{s_r w_r}$$

This may be solved in closed form for a

$$a := \frac{w_l \sqrt{s_l w_l} + (l - w_r)\sqrt{s_r w_r}}{\sqrt{s_l w_l} + \sqrt{s_r w_r}}$$

2.4 near to C2 condition

The true C2 condition is well conditioned except in the limit as the polygon tends to straight, when the straight corner gets cut more and more finely. The problem appears in a possible divide by zero s_r in the optimised code for computing the changepoints.

$$a := w_l + \frac{(l - w_l - w_r)}{1 + \sqrt{s_l w_l / s_r w_r}}$$

where s_i is the sin of the angle turned through by the polygon at the ith vertex.

An engineering solution to this situation is to allow the C2 condition to be lost in situations of low curvature. A suitable way of coding this is to prevent any of the s_i from taking the value zero.

Variants here are to add a bias to all s_i values, or to clip s_i

$$\text{if } s_i < .1 \text{ then } s_i := .1 \text{ fi}$$

3 Analysis

In this section is it shown that for each of the main variants a solution exists, that that solution is unique, and that the construction converges.

The variants are:-

– fixed positions

– fixed parametric ratios

– C2 condition

– near to C2

The first of these has a trivial proof, since from the fixed positions of the change points within the legs of the polygon, the rational speed condition is merely applied to each vertex independently. The closed form is well conditioned and leads immediately to a unique solution without iteration.

For the other variants we need to work harder.

3.1 Proof of Existence

We prove that a solution satisfying both the rational speed and the continuity of magnitude of curvature conditions exists, by induction on the number of points in the control polygon. A simpler proof with the same structure applies to the fixed parametric ratios variant.

Suppose that a solution exists for a polygon on n vertices, $n \geq 3$

We can divide a polygon of $n+1$ vertices into two pieces by breaking it at a candidate tangency point in the interior of the $\lfloor (n+1)/2 \rfloor$th edge, thus producing one piece of $l = \lfloor (n+1)/2 \rfloor + 1$ vertices and the other of $n+3-l$ vertices.

For $n \geq 3$ each half has no more than n vertices, and so whichever tangency point is chosen, a curve can be found in each half, satisfying both conditions. The only remaining condition to satisfy is the continuity of magnitude of curvature condition at the candidate tangency point.

As this point moves along its edge from, say, the left end to the right, the configuration changes from one where the magnitude of curvature of the left is unbounded and that on the right has a definite finite value to one where the opposite holds. Both curvatures vary continuously with the position of the tangency point. and so there must be a position at which their magnitudes are equal.

(In the fixed parametric ratios case, the change point merely has to be positioned in such a way that the distance to the left cut point and that to the right cut point are in the required ratio. The actual ratio of distances also varies from 0:+ at one end to +:0 at the other.)

Thus a polygon of $n + 1$ vertices has a curve satisfying both conditions.

It is trivial to show that a polygon of 3 vertices has such a curve, since it only needs to satisfy the rational speed condition, and by induction a curve exists for all polygons with 3 or more vertices.

3.2 Proof of Uniqueness

This follows the same structure as that of existence, except that we have also to prove that there is only one place for the tangency point which gives equality of magnitude of curvature. (or correct ratio of distances to the cut point).

To do this we show that the curvatures on both sides of the tangency point are monotonic functions of the position of the tangency point as it moves along its polygon edge.

It is sufficient to consider one side, as the proof will also apply to the other, so we take the left side The complication is that as the tangency point moves along the edge, the value of w will change, to maintain the rational speed condition, and the tangency point in the previous edge will also have to move. In fact the effects ripple all the way back to the beginning of the curve.

The proof of monotonicity is again by induction.

Suppose that vertex V_i is distant a_i from the tangency point T_{i-1} on its left and b_i from the tangency point T_i on its right. Let the w value at this vertex be w_i. Let the curvatures at the tangency points be k_{i-1} and k_i.

If the left part of the curve has only one piece of curve, it is straightforward to show that

$$\partial k_i / \partial b_i < 0$$

Suppose that this is true for all values of i up to $n-1$

Consider a polygon half of n vertices, so that the tangency point we want to prove results at is T_n

We know that $\partial k_{n-1}/\partial b_{n-1} < 0$, and because $b_{n-1}+a_n$ is constant, $\partial k_{n-1}/\partial a_n > 0 = P$, say.

Now the variables $k_{n-1}, k_n, a_n, w_n, b_n$ are related by the following equations:-

$$k_{n-1} = S_n w_n/(a_n - w_n)^2$$

$$k_n = S_n w_n/(b_n - w_n)^2$$

$$4Cw_n^2 = (a_n - w_n)(b_n - w_n)$$

but instead of the last of these we use the equation

$$16C^2 k_n k_{n-1} w_n^4 = S_m^2 w_n^2$$

or

$$16C^2 k_n k_{n-1} w_n^2 = S_n^2$$

Take differentials of all these equations.

$$(a_n - w_n)^2\, dk_{n-1} + 2k_{n-1}(a_n - w_n)(da_n - dw_n) = S_n\, dw_n$$
$$(b_n - w_n)^2\, dk_n + 2k_n(b_n - w_n)(db_n - dw_n) = S_n\, dw_n$$
$$16C^2(k_n w_n^2\, dk_{n-1} + k_{n-1} w_m^2\, dk_n + 2k_n k_{n-1} = 0$$

and collect terms. Including the equation for P above gives us the matrix equation

$$\begin{bmatrix} -P & 1 & 0 & 0 & 0 \\ (a_n - w_n)^2 & 2k_{n-1}(a_n - w_n) & -[2k_{n-1}(a_n - w_n) + S_n] & 0 & 0 \\ k_n w_n^2 & 0 & 2k_n k_{n-1} & k_{n-1} w_n^2 & 0 \\ 0 & 0 & -[2k_n(b_n - w_n) + S_n] & (b_n - w_n)^2 & 2k_n(b_n - w_n) \end{bmatrix}$$

$$* \begin{bmatrix} \delta k_{n-1} \\ \delta a_n \\ \delta w_n \\ \delta k_n \\ \delta b_n \end{bmatrix} = 0$$

Some of the terms in the above have been left in an apparently awkward form. This is so that it can be confirmed that the pattern of signs in the matrix is just

$$\begin{bmatrix} - & + & 0 & 0 & 0 \\ + & + & - & 0 & 0 \\ + & 0 & + & + & 0 \\ 0 & 0 & - & + & + \end{bmatrix}$$

Taking the last column to the right hand side, and carrying out row operations to reduce the matrix to upper triangular gives

$$\begin{bmatrix} - & + & 0 & 0 \\ 0 & + & - & 0 \\ 0 & 0 & + & + \\ 0 & 0 & 0 & + \end{bmatrix} \begin{bmatrix} \delta k_{n-1} \\ \delta a_n \\ \delta w_n \\ \delta k_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ - \end{bmatrix} \delta b_n$$

which shows that $\partial k_n / \partial b_n < 0$

3.3 Proof of Convergence

Consider the configuration as shown in fig 6, and which we assume to be in the state where the rational speed conditions are satisfied, and the continuity of curvature conditions not yet satisfied.

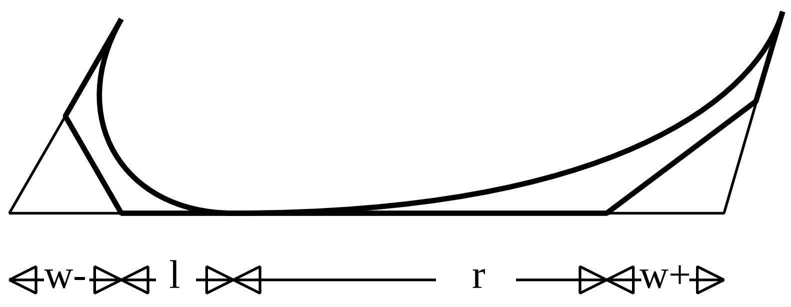


FIG 6

Note that the condition that all the distances marked in figure 6 are constant in sign under each of the two operations of the construction, and therefore under any sequence of those operations.

R_-, R and R_+ are the residuals of the local continuity of curvature equations, which we assume to be small enough to justify the use of differential arguments.

Continuity of curvature in the central span has the equation

$$\frac{S_- w_-}{l^2} = \frac{S_+ w_+}{r^2}$$

which we can put into the form

$$R = log\left(\frac{S_- w_- r^2}{S_+ w_+ l^2}\right) = 0$$

Differentiating gives the form

$$\delta R = \frac{\delta w_-}{w_-} + \frac{2\delta r}{r} - \frac{\delta w_+}{w_+} - \frac{2\delta l}{l}$$

Our first application of this form is to determine the amount by which the point of tangency moves when the continuity equation is satisfied without changing the ws.

We are choosing δa to make the residual zero.

$$\text{Now } \delta w_- = \delta w_+ = 0$$

$$\text{and } \delta l = -\delta r = \delta a$$

Thus

$$\delta R = -\frac{2\delta a}{r} - \frac{2\delta a}{l}$$

and so

$$\delta a = \frac{-rl\delta R}{2(r+l)} = \frac{rlR}{2(r+l)}$$

The effect of this change in a (and in the as of all the other edges of the polygon) is that the rational speed condition is no longer satisfied, and so it is necessary to modify all the w values to restore the situation.

The rational speed equation at the left hand end of our central edge is

$$C_- w_-^2 = r_- l$$

and again we take logarithms and differentiate

$$\frac{2\delta w_-}{w_-} - \frac{\delta r_-}{r_-} - \frac{\delta l}{l} = 0$$

Now we have

$$\delta r_- = -\delta a_- - \delta w_-$$

$$\delta l = \delta a - \delta w_-$$

so that

$$\delta w_- \left[\frac{2}{w_-} + \frac{1}{r_-} + \frac{1}{l} \right] + \frac{\delta a_-}{r_-} - \frac{\delta a}{l} = 0$$

Hence

$$\delta w_- = \frac{w_-(r_- \delta a - l \delta a_-)}{2r_- l + w_- l + w_- r_-}$$

Similarly, the change in w_+ at the right hand end of the central edge is

$$\delta w_+ = \frac{w_+(r \delta a_+ - l_+ \delta a)}{2rl_+ + w_+ l_+ + w_+ r}$$

These changes cause the residual in the continuity of curvature equation to depart from zero again. In the situation where a is now being held constant, we have

$$\delta R = \frac{\delta w_-}{w_-} + \frac{2\delta r}{r} - \frac{\delta w_+}{w_+} - \frac{2\delta l}{l}$$

where

$$\delta r = -\delta w_+$$

and

$$\delta l = -\delta w_-$$

so that

$$\delta R = \delta w_- \left(\frac{1}{w_-} + \frac{2}{l} \right) - \delta w_+ \left(\frac{1}{w_+} + \frac{2}{r} \right)$$

Substituting into this equation the values of δw_- and δw_+ just derived we get

$$\delta R = \frac{w_-(r_- \delta a - l \delta a_-)}{(2r_- l + w_- l + w_- r_-)} \left(\frac{1}{w_-} + \frac{2}{l} \right) - \frac{w_+(r \delta a_+ - l_+ \delta a)}{(2rl_+ + w_+ l_+ + w_+ r)_+} \left(\frac{1}{w_+} + \frac{2}{r} \right)$$

Make the further substitution of the values of the δa in terms of the original residuals, and separate out the separate terms coming from those original residuals.

$$\delta R = \frac{w_-(r_-\delta a)}{2r_-l + w_-l + w_-r_-}\left(\frac{1}{w_-} + \frac{2}{l}\right)$$
$$-\frac{w_-(l\delta a_-)}{2r_-l + w_-l + w_-r_-}\left(\frac{1}{w_-} + \frac{2}{l}\right)$$
$$-\frac{w_+(r\delta a_+)}{2rl_+ + w_+l_+ + w_+r_+}\left(\frac{1}{w_+} + \frac{2}{r}\right)$$
$$+\frac{w_+(l_+\delta a)}{2rl_+ + w_+l_+ + w_+r_+}\left(\frac{1}{w_+} + \frac{2}{r}\right)$$

$$= \delta a\left(\frac{w_-r_-}{2r_-l + w_-l + w_-r_-}\left(\frac{1}{w_-} + \frac{2}{l}\right) + \frac{w_+l_+}{2rl_+ + w_+l_+ + w_+r_+}\left(\frac{1}{w_+} + \frac{2}{r}\right)\right)$$
$$-\delta a_-\frac{w_-l}{2r_-l + w_-l + w_-r_-}\left(\frac{1}{w_-} + \frac{2}{l}\right)$$
$$-\delta a_+\frac{w_+r}{2rl_+ + w_+l_+ + w_+r_+}\left(\frac{1}{w_+} + \frac{2}{r}\right)$$

$$= \frac{Rrl}{2(r+l)}\left(\frac{w_-r_-}{2r_-l + w_-l + w_-r_-}\left(\frac{1}{w_-} + \frac{2}{l}\right) + \frac{w_+l_+}{2rl_+ + w_+l_+ + w_+r_+}\left(\frac{1}{w_+} + \frac{2}{r}\right)\right)$$
$$-\frac{R_-r_-l_-}{2(r_- + l_-)}\frac{w_-l}{2r_-l + w_-l + w_-r_-}\left(\frac{1}{w_-} + \frac{2}{l}\right)$$
$$-\frac{R_+r_+l_+}{2(r_+ + l_+)}\frac{w_+r}{2rl_+ + w_+l_+ + w_+r_+}\left(\frac{1}{w_+} + \frac{2}{r}\right)$$

Thus the vector of residuals may be regarded as being multiplied by a triband matrix at each iteration consisting of a continuity correction followed by a rational speed correction.

Consider these terms of this matrix one at a time.

Take the diagonal terms first

$$\frac{rl}{2(r+l)}\left(\frac{w_-r_-}{2r_-l + w_-l + w_-r_-}\left(\frac{1}{w_-} + \frac{2}{l}\right) + \frac{w_+l_+}{2rl_+ + w_+l_+ + w_+r_+}\left(\frac{1}{w_+} + \frac{2}{r}\right)\right)$$

$$= \frac{rl}{2(r+l)} \frac{w_-r_-}{2r_-l + w_-l + w_-r_-} \left(\frac{l + 2w_-}{lw_-} \right) + \frac{rl}{2(r+l)} \frac{w_+l_+}{2rl_+ + w_+l_+ + w_+r_+} \left(\frac{r + 2w_+}{rw_+} \right)$$

$$= \frac{rlw_-r_-(l + 2w_-)}{2lw_-(r+l)(2r_-l + w_-l + w_-r_-)} + \frac{rlw_+l_+(r + 2w_+)}{2rw_+(r+l)(2rl_+ + w_+l_+ + w_+r)}$$

$$= \frac{rr_-(l + 2w_-)}{2(r+l)(2r_-l + w_-l + w_-r_-)} + \frac{ll_+(r + 2w_+)}{2(r+l)(2rl_+ + w_+l_+ + w_+r)}$$

$$= \left(\frac{r}{r+l} \right) \left(\frac{(r_-l + 2w_-r_-)}{2(2r_-l + w_-l + w_-r_-)} \right) + \left(\frac{l}{r+l} \right) \left(\frac{(rl_+ + 2w_+l_+)}{2(2rl_+ + w_+l_+ + w_+r)} \right)$$

All diagonal terms are positive, because all the variables included here are themselves positive.

The subdiagonal terms

$$-\frac{r_-l_-}{2(r_- + l_-)} \frac{w_-l}{2r_-l + w_-l + w_-r_-} \left(\frac{l + 2w_-}{w_-l} \right)$$

$$= -\frac{r_-l_-w_-l(l + 2w_-)}{2w_-l(r_- + l_-)(2r_-l + w_-l + w_-r_-)}$$

$$= -\frac{r_-l_-(l + 2w_-)}{2(r_- + l_-)(2r_-l + w_-l + w_-r_-)}$$

$$= -\frac{1}{2} \left(\frac{l_-}{r_- + l_-} \right) \left(\frac{r_-l + 2w_-r_-}{2r_-l + w_-l + w_-r_-} \right)$$

All subdiagonal terms are negative.

The superdiagonal terms are similarly

$$-\frac{1}{2} \left(\frac{r_+}{r_+ + l_+} \right) \left(\frac{l_+r + 2w_+l_+}{2l_+r + w_+r + w_+l_+} \right)$$

These are also all negative.

Consider the sum of magnitudes of the terms in any column of this matrix.

$$= \left(\frac{r}{r+l}\right)\left(\frac{(r_-l+2w_-r_-)}{2(2r_-l+w_-l+w_-r_-)}\right) + \left(\frac{l}{r+l}\right)\left(\frac{(rl_++2w_+l_+)}{2(2rl_++w_+l_++w_+r)}\right)$$

$$+\frac{1}{2}\left(\frac{l}{r+l}\right)\left(\frac{rl_++2w_+r}{2rl_++w_+l_++w_+r}\right)$$

$$+\frac{1}{2}\left(\frac{r}{r+l}\right)\left(\frac{lr_-+2w_-l}{2lr_-+w_-r_-+w_-l}\right)$$

$$= \frac{1}{2}\left[\left(\frac{r}{r+l}\right)\left(\frac{(r_-l+2w_-r_-)}{(2r_-l+w_-l+w_-r_-)}\right) + \left(\frac{l}{r+l}\right)\left(\frac{(rl_++2w_+l_+)}{(2rl_++w_+l_++w_+r)}\right)\right.$$

$$+\left(\frac{l}{r+l}\right)\left(\frac{rl_++2w_+r}{2rl_++w_+l_++w_+r}\right)$$

$$\left.+\left(\frac{r}{r+l}\right)\left(\frac{lr_-+2w_-l}{2lr_-+w_-r_-+w_-l}\right)\right]$$

$$= \frac{1}{2}\left[\left(\frac{r}{r+l}\right)\left(\frac{(2r_-l+2w_-l+2w_-r_-)}{(2r_-l+w_-l+w_-r_-)}\right) + \left(\frac{l}{r+l}\right)\left(\frac{(2rl_++2w_+l_++2w_+r)}{(2rl_++w_+l_++w_+r)}\right)\right]$$

$$< 1$$

Consider the sum of the moduli of the residuals.

$$\sum_i |R'_i| = \sum_i |\sum_j a_{ij}R_j|$$

$$\leq \sum_i \sum_j |a_{ij}R_j|$$

$$= \sum_i \sum_j |a_{ij}||R_j|$$

$$= \sum_j \sum_i |a_{ij}||R_j|$$

But we just showed that $\sum_i |a_{ij}| < 1$, and therefore

$$\sum_i |R'_i| < \sum_j |R_j|$$

This norm therefore strictly decreases at every step.

QED.

The worst case is when w is large compared with r and l, which is when the angles turned through approach 180 degrees. In these circumstances, the vector of residuals rapidly becomes alternating in sign, and convergence thereafter is slow. For turning angles of about 90 degrees, the reduction in norm is about a factor of 3/4 per step. For zero turning angles, the reduction is 2/3 per step.

4 Code

The following code is designed for ease of understanding rather than for efficiency. So that left and right neighbours in the curve can consistently be represented by subscript incrementing and decrementing, alternate entries are used in the arrays holding the data. Vertex information uses the odd-valued subscripts, edge data the even-valued subscripts.

The actual iteration is carried out a fixed number of times, to avoid complicating the code by convergence tests.

Faster convergence can almost certainly be achieved by Newton iteration to all of the conditions simultaneously, but this code separates out the two stages to make the variants stand out more clearly.

Serious use of this algorithm will therefore require recoding. The following code can, however, be used for regression testing of any more efficient version.

4.1 I/O Data formats

Input is read from a file containing first the number of polygon control points, then the coordinate pairs representing the control points.

Output is in postscript, with no change of units or origin. The input should therefore be chosen to suit the size of the required image, with coordinates measured in 'points' (units of 1/72 of an inch or 0.353 mm)

4.2 Internal data structures

4.2.1 Vertex data

Number of polygon vertices NPV

Polygon vertex coordinates X(2*NPV-1), Y(2*NPV-1)

As mentioned above, only the odd subscripts are used, starting from 1.

4.2.2 Edge data

Edge components DX(2*NPV-2), DY(2*NPV-2)

Edge lengths EL(2*NPV-2)

Distances from vertices to change points A(2*NPV-2), B(2*NPV-2)

for edge properties only even subscripts are used, starting from 2.

4.2.3 Internal Vertex data

Internal vertex angles S(2*NPV-3), C(2*NPV-3)

Distances from vertices to corner cutting W(2*NPV-3)

The internal vertices use odd subscripts from 3.

4.3 Code

4.3.1 Input

```
READ ( *,*) NPV
DO  IPV = 1, 2*NPV-1, 2
   READ ( *,*) X(IPV), Y(IPV)
ENDDO
```

4.3.2 Determination of constant edge properties

```
      DO IE = 2, 2*NPV-2, 2
         DDX = X(IE+1) - X(IE-1)
         DDY = Y(IE+1) - Y(IE-1)
         EL(IE) = SQRT( DDX*DDX + DDY*DDY )
         DX(IE) = DDX
         DY(IE) = DDY
      ENDDO
```

4.3.3 determination of constant internal vertex properties

```
      DO IIV = 3, 2*NPV-3, 2
         SS = DX(IIV-1)*DY(IIV+1) - DY(IIV-1)*DX(IIV+1)
         CC = DX(IIV-1)*DX(IIV+1) + DY(IIV-1)*DY(IIV+1)
         TT = SQRT(SS*SS + CC*CC)
         SS = SS/TT
         IF( SS .LT. 0.0 ) SS = -SS
C                              clipping for small S
         IF( SS .LT. 0.1 ) SS = 0.1
         S(IIV) = SS
         C(IIV) = CC/TT
      ENDDO
```

4.3.4 initial changepoints(equal interval)

```
      DO IE = 2, 2*NPV-2, 2
         A(IE) = EL(IE)*0.5
         B(IE) = EL(IE) - A(IE)
      ENDDO
```

4.3.5 initial changepoints(biassed by angle)

```
      A(2) = 0.5*EL(2)
      B(2) = EL(2)-A(2)
      A(2*NPV-2) = 0.5*EL(2*NPV-2)
      B(2*NPV-2) = EL(2*NPV-2) - A(2*NPV-2)
      DO IE = 4, 2*NPV-4, 2
C                         AL and AR are cheap estimators of angle
C                         S/(C+1) is half angle, accurate to
C                         angle to the fifth.  Replacing the 1 by
C                         1.1 prevents blowup at angle = pi.
         AL = S(IE-1)/(C(IE-1)+1.1)
         AR = S(IE+1)/(C(IE+1)+1.1)
         A(IE) = EL(IE)*AL/(AL+AR)
         B(IE) = EL(IE) - A(IE)
      ENDDO
```

4.3.6 determination of W

```
      DO IIV = 3, 2*NPV-3, 2
         SIG = A(IIV+1) + B(IIV-1)
         PRD = A(IIV+1)*B(IIV-1)
         W(IIV) = 2*PRD/
     *   (SIG + SQRT(SIG*SIG + 4.0*PRD*(2.0*CC(IIV)+1.0))
      ENDDO
```

4.3.7 determination of changepoints (equal intervals)

```
C                         end spans are left unaltered.
      DO IE = 4, 2*NPV-4, 2
         A(IE) = 0.5*(EL(IE) + W(IE-1) - W(IE+1))
         B(IE) = EL(IE) - A(IE)
      ENDDO
```

4.3.8 determination of changepoints (C2)

The code here gives continuity of magnitude of curvature if the s values are left unclipped in the initialisation above, near C2 if the clipping is carried out.

```
C                       end spans are left unaltered.
      DO IE = 4, 2*NPV-4, 2
         RWSL = SQRT(W(IE-1)*S(IE-1))
         RWSR = SQRT(W(IE+1)*S(IE+1))
         A(IE) = (EL(IE) - W(IE-1) - W(IE+1))*RWSL/
     *                (RWSL+RWSR) + W(IE-1)
         B(IE) = EL(IE) - A(IE)
      ENDDO
```

4.3.9 Output of Preliminary information

This opens a postscript file with a bounding box, suitable for importing into LaTeX files with psfig.

```
      WRITE(*,*) '%!'
      XMAX = X(1)
      YMAX = Y(1)
      XMIN = XMAX
      YMIN = YMAX
      DO IPV = 3, 2*NPV, 2
        IF( X(IPV) .GT. XMAX ) XMAX = X(IPV)
        IF( Y(IPV) .GT. YMAX ) YMAX = Y(IPV)
        IF( X(IPV) .LT. XMIN ) XMIN = X(IPV)
        IF( Y(IPV) .LT. YMIN ) YMIN = Y(IPV)
      ENDDO
      WRITE(*,11) XMIN,YMIN,XMAX,YMAX
11    FORMAT('%%Bounding Box:', 4F6.1)
```

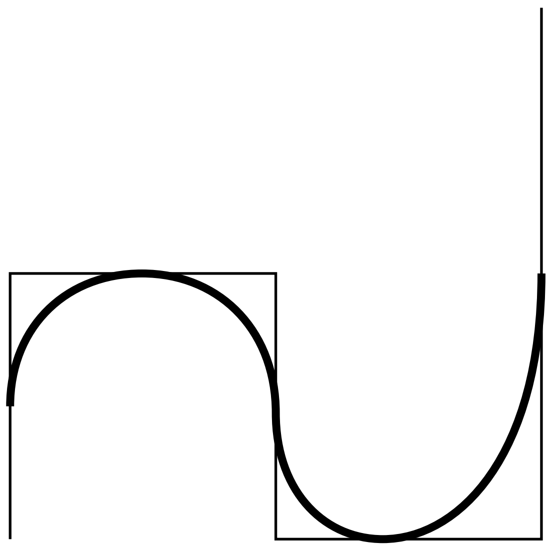
4.3.10 Output Curve

```
      R = B(2)/L(2)
      XL = X(3) - R*DX(2)
      YL = Y(3) - R*DY(2)
      WRITE(*,21) XL,YL
21    FORMAT( 'newpath'/2F6.1, ' moveto')
      DO IIV = 3, 2*NPV - 3, 2
         R = W(IIV)/EL(IIV-1)
         X1 = X(IIV) - R*DX(IIV-1)
         Y1 = Y(IIV) - R*DY(IIV-1)
         R = W(IIV)/EL(IIV+1)
         X2 = X(IIV) + R*DX(IIV+1)
         Y2 = Y(IIV) + R*DY(IIV+1)
         R = A(IIV+1)/EL(IIV+1)
         X3 = X(IIV) + R*DX(IIV+1)
         Y3 = Y(IIV) + R*DY(IIV+1)
         WRITE(*,22) X1,Y1, X2,Y2, X3,Y3
22       FORMAT( 6F6.1, ' curveto')
      ENDDO
      WRITE(*,23)
23    FORMAT('stroke'/'showpage')
```

5 Examples

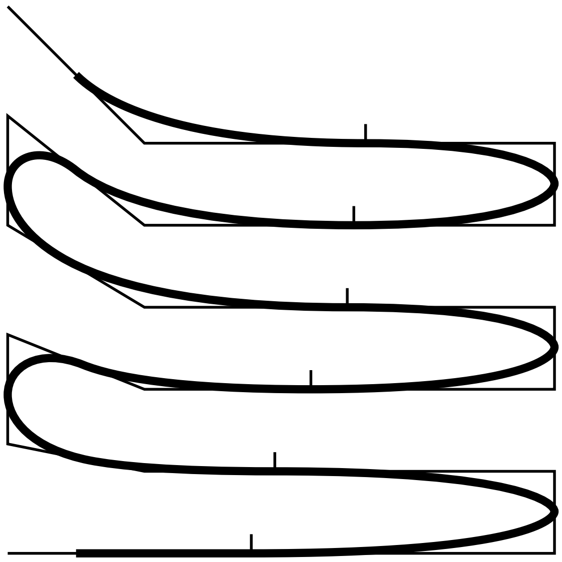
5.1 Example 1

```
6
100 100
100 200
200 200
200 100
300 100
300 300
```

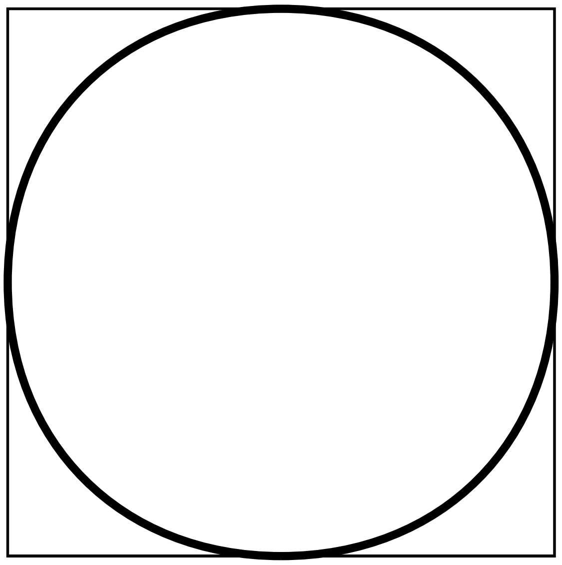


5.2 Example 2

```
18
100 100
150 100
300 100
300 130
150 130
100 140
100 180
150 160
300 160
300 190
150 190
100 220
100 260
150 220
300 220
300 250
150 250
100 300
```

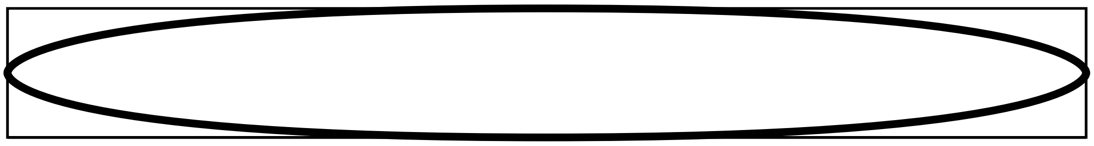


5.3 Example 3



```
6
100 100
300 100
300 300
100 300
100 100
300 100
```

5.4 Example 4



6 References

G. Albrecht and R. T. Farouki,
Construction of C2 Pythagorean-hodograph interpolating splines by the homotopy method.
Advances in Comp Math, to appear

W. L. F. Degen,
High Accuracy Approximation of Parametric Curves.
in Mathematical Methods for Curves and Surfaces. Vanderbilt Univ Press (1995) pp83–98.

R. T. Farouki,
The conformal map $z \mapsto z^2$ of the hodograph plane.
CAGD **10** (1995) pp363–390.

R. T. Farouki and C. A. Neff,
Hermite Interpolation by Pythagorean hodograph quintics.
Math Comp to appear

R. T. Farouki and T. Sakkalis,
Pythagorean Hodographs.
IBM J Res Develop **34**, pp736-752 (1990)

Field, D. and W. H. Frey,
Constructing Bezier Conic Segments with Monotone Curvature.
in Proceedings of 5th IFIP Workshop on Solid Modeling in CAD. Chapman and Hall, 1997 (to appear).

T. Pavlidis,
Curve Fitting with Conic Splines.
ACMToG **2**(1983) pp1–31.

H. Pottmann,
Locally Controllable Conic Splines with Curvature Continuity.
ACMToG **10** (1991) pp366–377.

V. Pratt,
Techniques for Conic Splines.
in Proceedings of ACM SIGGRAPH 1985 pp151–159.