Curvature combs and harmonized paths in METAPOST

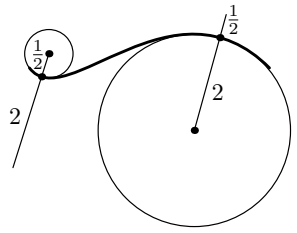
Linus Romer

Abstract

Most font editors offer curvature related tools. One of these tools is the visualization of curvature via *curvature combs*. Another tool is the so-called *harmonization*, which makes the curvature continuous along paths. An implementation of both tools in METAPOST will be presented. Curvature optimized paths already play a significant role in METAFONT and METAPOST and therefore some exemplary METAPOST paths will be examined for their curvature behavior.

1 Curvature

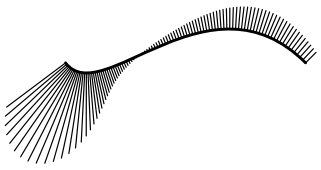
The curvature in a point of a curve is the inverse of the radius of the osculating circle at this point (depicted here as a "curvature vector" on the opposite side of the radius):



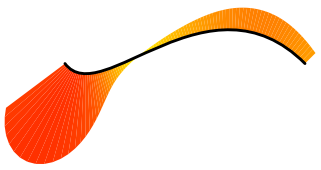
For straight segments, the curvature is constantly zero, since the radius of the osculating circle is infinitely large. Vice versa, the curvature becomes infinitely large when the radius of the osculating circle tends to zero.

2 Curvature combs in METAPOST

We can assemble these curvature vectors into a curvature comb:



The curvature may be additionally mapped to a color and the gaps may be filled:



A figure as above can be achieved by

`path p; p =` ⟨*path*⟩ `; comb(p,300); draw p;`

using the `comb` macro that will be presented here (the `300` scales the curvature comb). We start by defining the macro `crossprod` that returns the cross product between two given vectors $\vec{w}$ and $\vec{z}$:

```
primarydef w crossprod z =
 (xpart w * ypart z - ypart w * xpart z)
enddef;
```

Then the macro `curv` shall be applied to a path `p` in order to return a "curvature vector" that is orthogonal to the path in its initial point. The length of the vector is proportional to the initial curvature of the path:

```
vardef curv expr p =
 save v,w,l; pair v,w;
 v = direction 0 of p;
 l = length v;
 v := v/l;
 w = (point 0 of p - 2*postcontrol 0 of p
                   + precontrol 1 of p)/l;
 2/3*(v crossprod w)/l*(v rotated -90)
enddef;
```

Here is the math behind this macro. A cubic Bézier segment can be described by:

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = t^3(3\vec{Q} - \vec{P} + \vec{S} - 3\vec{R}) + 3t^2(\vec{P} - 2\vec{Q} + \vec{R}) + 3t(\vec{Q} - \vec{P}) + \vec{P}$$

The initial derivatives are then:

$$\begin{pmatrix} \dot{x}(0) \\ \dot{y}(0) \end{pmatrix} = 3\underbrace{(\vec{Q} - \vec{P})}_{=:\vec{v}} \qquad \begin{pmatrix} \ddot{x}(0) \\ \ddot{y}(0) \end{pmatrix} = 6\underbrace{(\vec{P} - 2\vec{Q} + \vec{R})}_{=:\vec{w}}$$

The signed curvature is calculated by $\dfrac{\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \times \begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix}}{\left|\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix}\right|^3}$.

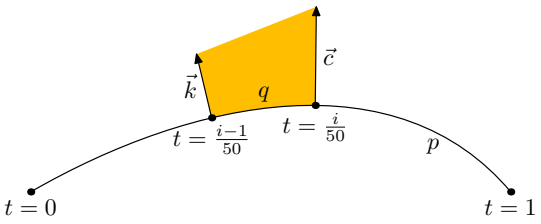
Using $l := |\vec{v}|$ we finally have the formula used in the macro for the initial curvature:

$$\frac{3\vec{v} \times 6\vec{w}}{(3l)^3} = \frac{2}{3}\frac{\vec{v} \times \vec{w}}{l^3} = \frac{2}{3l} \cdot \left(\frac{1}{l}\vec{v} \times \frac{1}{l}\vec{w}\right)$$

The divisions by l are necessary to prevent arithmetic overflows. The special case $\left|\begin{pmatrix} \dot{x}(0) \\ \dot{y}(0) \end{pmatrix}\right| = 0$ is not handled here. The curvature then would diverge to $\pm\infty$ (or be 0 if the cubic Bézier segment is a line segment).

Now we define the curvature comb macro of a path p by subdividing each segment in 50 parts and filling an area over each part. Each part of the comb is made of two subsequent "curvature" vectors $\vec{k}$, $\vec{c}$ that are scaled by a constant factor s given by the user. The color depends on the average length of them.

xnotdoi.org/10.47397/tb/0-0/curvaturecombharmonize




```
vardef comb(expr p,s) =
 save q,c,k; path q; pair c,k;
 for n = 0 upto length(p)-1:
  c := s * curv subpath(n,n+1) of p;
  for i = 1 upto 50:
   k := c;
   c := s * curv subpath(n+i/50,
             n if i<25: +1 fi) of p;
   q := subpath(n+(i-1)/50,n+i/50) of p;
   fill q -- point 1 of q + c
          -- point 0 of q + k
          -- cycle withcolor
      (1,1/(1+.1*.5[length c,length k]),0);
  endfor
 endfor
enddef;
```

The condition `if i<25: +1 fi` makes the subpath as large as possible to get better accuracy.

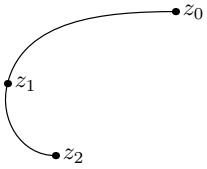
A curvature of 0 is mapped to yellow and an infinitely large curvature is mapped to red. This is done by changing the green value between 1 and 0. If you increase the `.1`, the green value tends faster to 0.

3 Harmonize paths in METAPOST

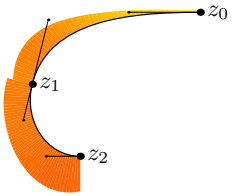
In METAPOST, the code

```
z0 = (70,60); z1 = (0,30); z2 = (20,0);
draw z0{left} .. z1 .. z2{right};
```

produces the following curve:

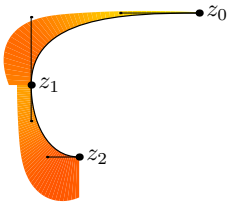


While the directions at the start and the end of the path were set by the user, METAPOST has chosen the angle of the path in z_1 to equalize the so-called *mock curvature* on both sides. The mock curvature is a Taylor approximation of the real curvature (Hobby, 1986). After that, two cubic Bézier segments that nearly minimize the curve energy have been drawn between the given points.



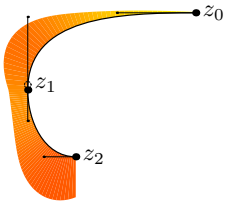

```
z0{left}
.. z1
.. z2{right}
```

The curvature comb in the preceding picture shows that the curvature in z_1 indeed is not continuous but only near continuous. When a user sets the direction in the joining knot, METAPOST has no possibility to optimize the curvature in the joint and the curvature often changes more abruptly in the joint (see the following picture). However, this case is frequent in type design because knots at horizontal and vertical extrema are preferred over knots with arbitrary direction.




```
z0{left}
.. z1{down}
.. z2{right}
```

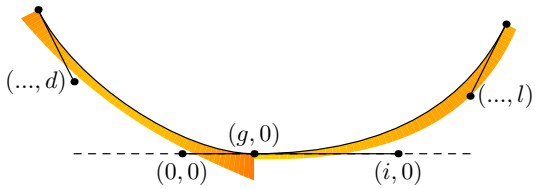
Fortunately, a METAPOST path can be modified to a curvature continuous curve by the later defined `harmonize` macro that moves the joining knot along its tangent:




```
harmonize z0{left}
        .. z1{down}
        .. z2{right}
```

4 The math of harmonization

Assume two adjoint cubic Bézier segments that have the same direction in their joint and do not have zero-handles. Furthermore, assume the joining knot to not be a point of inflection. By translation and rotation we can force one control point next to the joining knot to lie on the origin of the coordinate system and the joining knot tangent to lie on the x-axis:



We want to choose g such that the curvature is continuous. So the curvature on both sides of $(g, 0)$ must

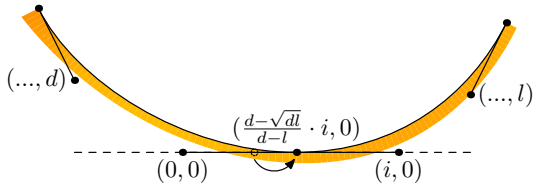
be equal:

$$\frac{2d}{3g^2} = \frac{2l}{3(i-g)^2}$$

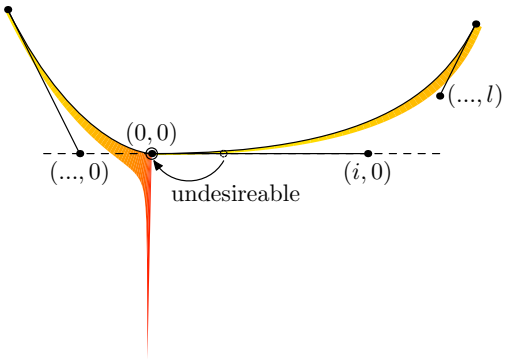
In the special case $d = l$, we get $g = i - g$. Solving for g we get

$$g = \begin{cases} \frac{d \pm \sqrt{dl}}{d-l} \cdot i & \text{if } d \neq l, \\ \frac{i}{2} & \text{else.} \end{cases}$$

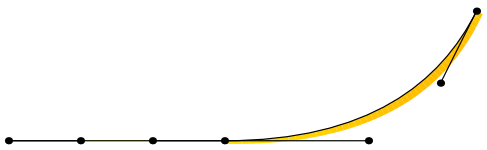
Since $\sqrt{dl}$ is the geometric mean between d and l, the solution $\frac{d-\sqrt{dl}}{d-l} \cdot i$ guarantees the joining knot to lie between its control points.



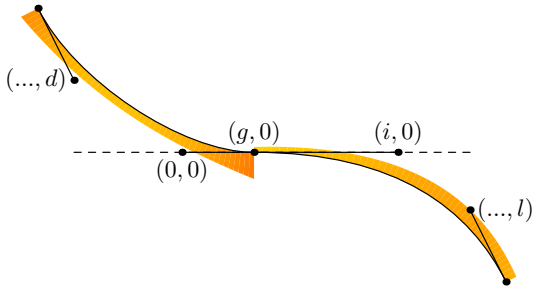
If either d or l is zero, $g = \frac{d-\sqrt{dl}}{d-l} \cdot i$ becomes either 0 or i. That means, that the joining knot will become collocated with one of its control points, which generally should be avoided. One reason for this avoidance is that the curvature might become infinitely large:



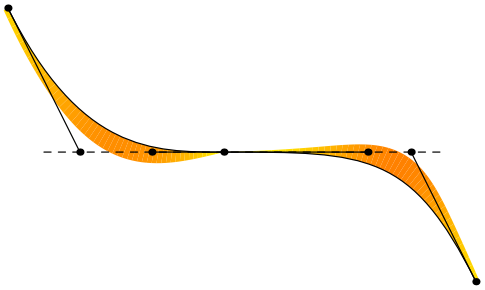
So, we will not alter the paths at all in the case of either d or l being zero. This case occurs also when a straight line goes over to a curve, which is quite frequent in type design:



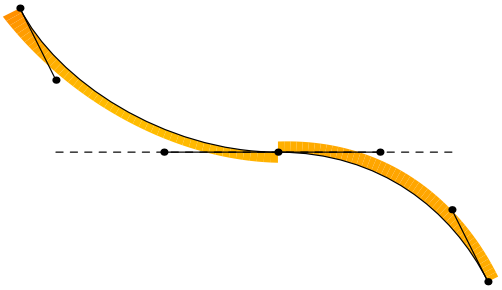
When the joining knot is a point of inflection, the curvatures $\frac{2d}{3g^2}$ and $\frac{2l}{3(i-g)^2}$ must have different signs.



Hence, a curvature-continuous solution forces $d = l = 0$. A curvature continuous solution then only has to satisfy that all control points must lie on one line e.g.:



In this situation, one could also satisfy further conditions like the preservation of area. On the other hand, having all four control points on the same line of the two affected cubic Bézier segments is critical. Due to rounding errors, such a conversion is likely to add new points of inflection. So, instead of this, we will only guarantee the *absolute value* of the curvature to be continuous in the case of points of inflection by moving the joining knot in the same manner as before in between its control points:



Finally, the solution of setting

$$g_{\text{new}} = \begin{cases} g_{\text{old}} & \text{if } d = 0 \text{ or } l = 0, \\ \frac{i}{2} & \text{else if } |d| = |l|, \\ \frac{|d|-\sqrt{|dl|}}{|d|-|l|} \cdot i & \text{else} \end{cases}$$

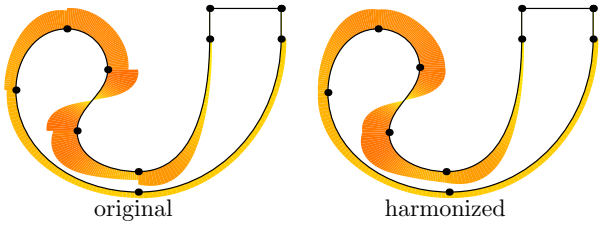
is chosen here and shall be the definition of *harmonization*. We define a corresponding macro `harmonize` that returns a harmonized version of a given path p. In the generic case, the tangent in the joining knot is not the x-axis (as depicted in the preceding figures), so we calculate d and l as the height to the tangent by cross products.

```
vardef harmonize expr p =
 save t,u,d,l,n,q; pair t,u,q[];
 n = length p;
 for j = if cycle p: 0 else: 1 fi upto n-1:
  q[j] = point j of p;
  t := unitvector(direction j of p);
  u := unitvector(point j of p
           - precontrol j of p);
  if eps > abs((u dotprod t) - 1): % smooth
   l := abs(t crossprod
       (precontrol j+1 of p - point j of p) );
   d := abs(t crossprod
       (postcontrol j-1 of p - point j of p) );
   if not ( (l = 0) or (d = 0) ):
    q[j] := if (d = l): .5 else:
         ((d-sqrt(d*l))/(d-l)) fi
         [precontrol j of p,postcontrol j of p];
   fi
  fi
 endfor
 if not cycle p:
  q[0] = point 0 of p;
  q[n] = point n of p;
 fi
 q[0] % start returned path
 for j = 0 upto n-1: % define new path
  .. controls postcontrol j of p
  and precontrol j+1 of p .. if (j = n-1)
  and (cycle p): cycle else: q[j+1] fi
 endfor
enddef;
```

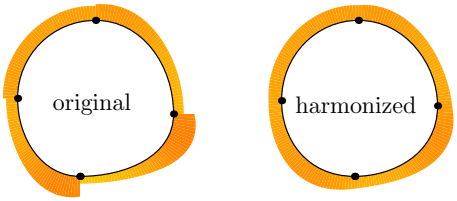
A mostly equivalent algorithm has been published in (Roach, 1990).

5 Examples of harmonization

Should you use harmonization? At least it does not harm to consider it. Most of the time, the changes are subtle as in the following bulb terminal:



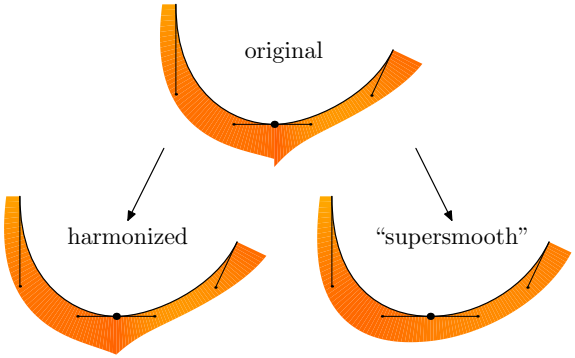
And sometimes they are less subtle as in the following calligraphic dots:



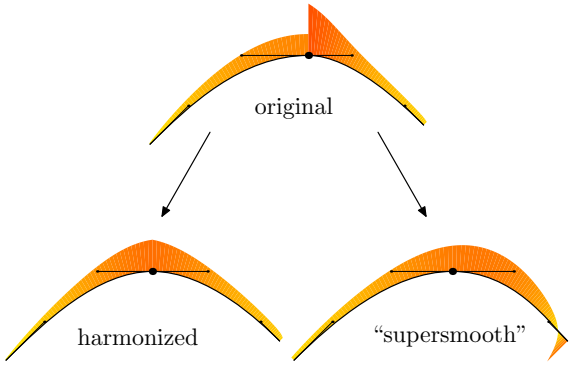
After harmonization, the dot has become more rounded and may have lost its "personality".

6 Smoothing out paths even more

Harmonization does not affect control points nor the curvatures at other joining knots. Hence, it can be easily used over several cubic Bézier segments. Nonetheless, harmonized paths normally no longer interpolate the knots they were originally meant to. The author once thought it might therefore be a good idea to leave the joining knots and move the control points instead. Then we not only can make the curvature continuous but also the the change of curvature. The curve then becomes some kind of "supersmooth".



However, there are some problems that come with this "supersmoothness": It might introduce additional points of inflection (see below). Furthermore, this will normally change the curvature at other knots and break curvature continuity there.



References

Hobby, John D. "Smooth, easy to compute interpolating splines". *Discrete & computational geometry* **1**(2), 123–140, 1986.

Roach, Robert L. "Curvature continuity of cubic Bezier curves in the solid modeling aerospace research tools design software". interim report, NASA Langley Research Center, 1990.

◇ Linus Romer
Ahornstrasse 8
Uznach, 8730
Switzerland