How to Halve

a Bezier or a Bspline curve.

Halving a Bezier curve.

One way to produce the Bezier representations of the two halves of a Bezier curve is to apply a change of basis. For example, we can convert the Bezier representation back to monomial form, apply the scaling

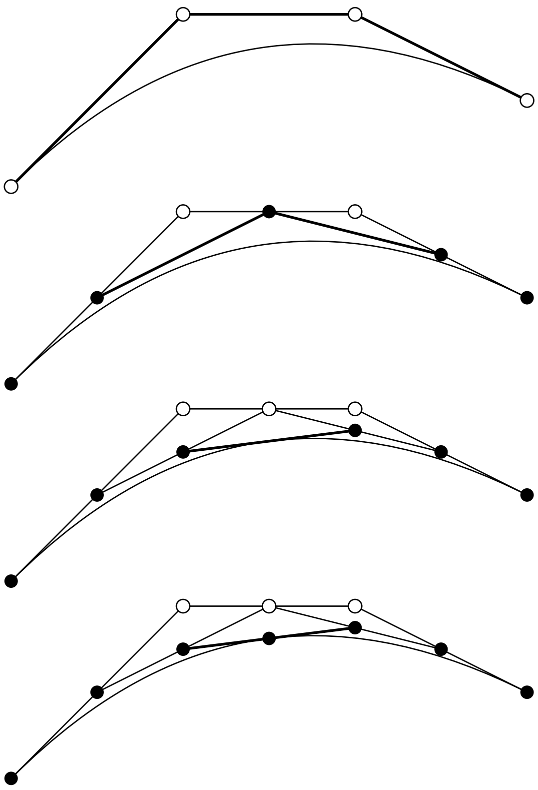
$$u = u'/2$$

and then reconvert to Bezier. At each stage we get a new set of coefficients, and at the last stage the coefficients are the Bezier control points of the first half of the original curve. The scaling

$$u = (u' + 1)/2$$

gives the control points of the second half.

Better, however, is to apply the **de Casteljau** algorithm.



A degree n curve has a control polygon with n edges. Call this polygon 0.

Make a new polygon of $n-1$ sides by putting new vertices at the midpoints of the old edges. Call this polygon 1.

Make a new polygon of $n-2$ sides by putting new vertices at the midpoints of the edges of polygon 1. Call this polygon 2.

Repeat this process until we are creating a polygon (polygon n) which has exactly one vertex and no edges. The vertex lies on the Bezier curve of the original polygon.

The points constructed in this way form the required Bezier polygons of the two halves. The de Casteljau algorithm can in fact be applied using division of each edge in the ratio $t : 1-t$ rather than just at the midpoint, to give the point on the curve at parameter t and also the Bezier polygons of the two parts of the original curve meeting there.

Exercise

Confirm that this process does indeed give the same result as the series of basis changes described first. You should find that the expressing of the Bezier basis functions as the terms of the binomial expansion of $(t + (1 - t))^n$ should be relevant.

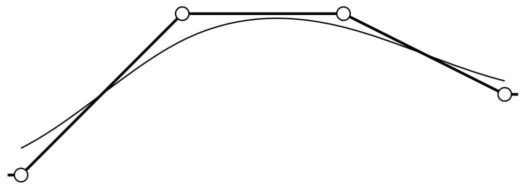
Halving a B-spline curve.

Because the B-spline curve is already a piecewise entity, the initial stages of halving can merely involve identifying that only parts of the sequence of control points are relevant to a particular piece. This means that the hulls of the two halves will definitely overlap, because there are some control points in the middle which are relevant to both halves.

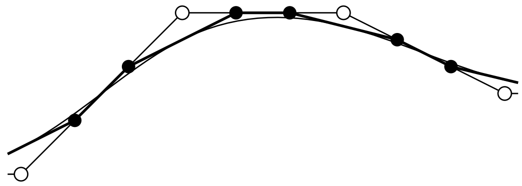
Resist the temptation to make copies of the control points at this stage.

Eventually the individual pieces will be the single polynomial spans. At this point it is definitely worth while converting to Bezier representations. The algorithm for this is closely related to de Casteljau. It is most elegantly derived in terms of knot insertion, which is part of unequal interval spline theory, which we have not covered.

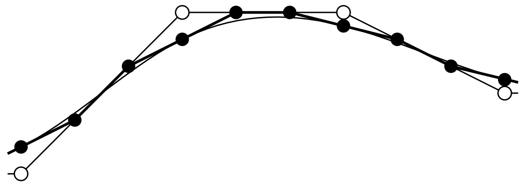
In the case of the cubic B-spline the construction is to start with the original B-spline control polygon.



Divide each edge at the 1/3 and 2/3 points, and construct a new polygon by cutting off the corners.



Finally insert new vertices at the midpoints of the edges which correspond to the original vertices. This now gives a sequence of Bezier pieces which match the original B-spline exactly.



Exercise

Confirm that this construction does indeed match the B-spline curve. Hint: it will be better mathematics if you can use the properties which a B-spline has and show that the piecewise Bezier curve also has them, rather than just ploughing through algebra.

Fast Implementations

In general we make more than one interrogation of any curve. It is therefore worthwhile spending a bit of computer time up front, creating data which allows subsequent enquiries to be made faster. In this case it is possible to build a tree of, for example, bounding boxes relatively quickly. Any subdivision interrogation will not then need to compute the bounding boxes as it goes. This process can be quite fast, since the bounding box for a pair of pieces can be computed from the bounding boxes of those pieces in constant time, regardless of the internal complexity of those pieces.