Curve Construction via Local Interpolation

Given $\{\boldsymbol{Q}_k\}$, $k = 0, \ldots, n$, construct a curve using local curve interpolation. A local curve interpolation scheme constructs *n* polynomial or rational segments, $\boldsymbol{C}_i(u)$, $i = 0, \ldots, n-1$, such that $\mathbf{Q}_i$ and $\mathbf{Q}_{i+1}$ are the end points of $\boldsymbol{C}_i(u)$. Neighboring segments are joined with some prescribed level of continuity.

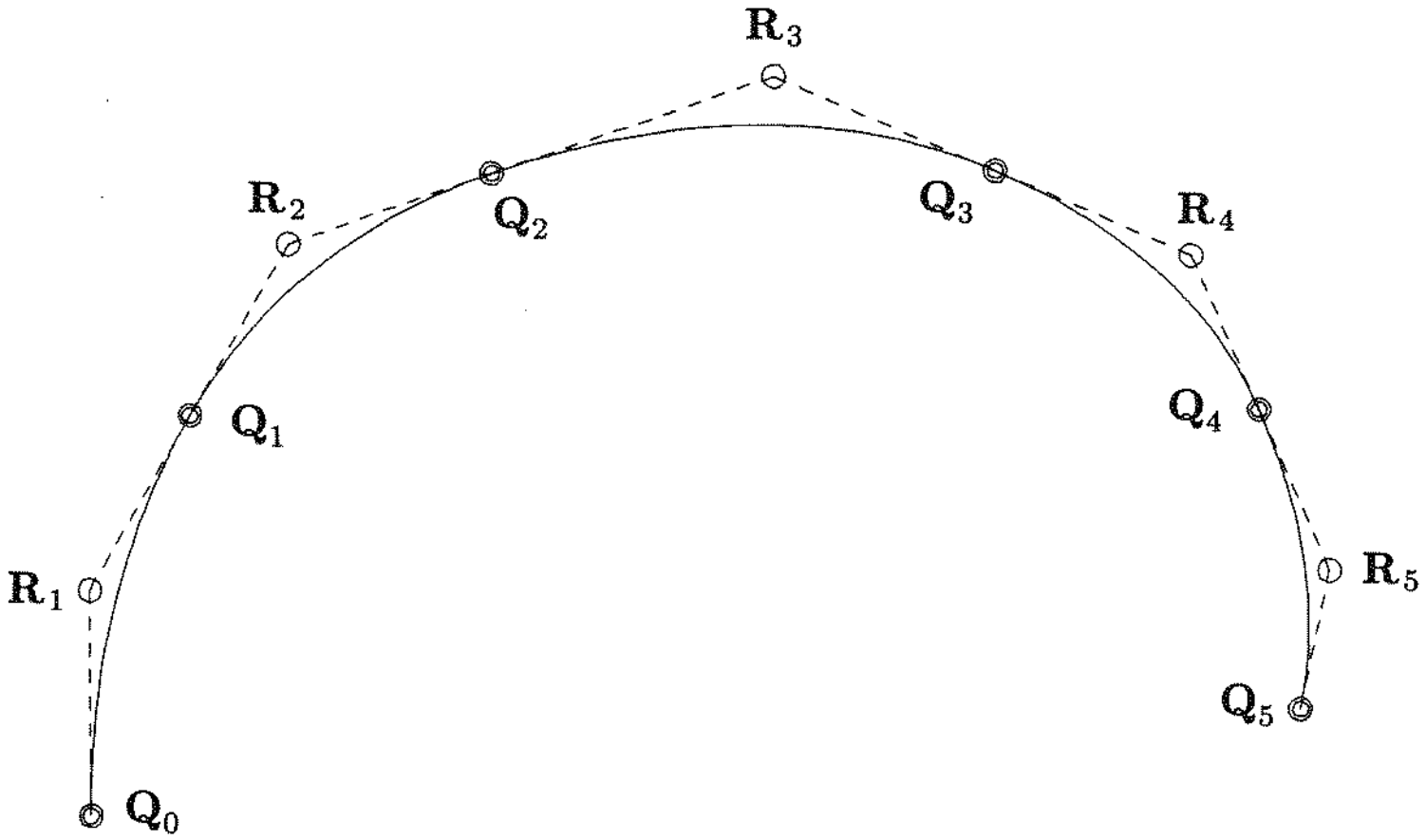
Polynomial or rational Bezier curves are used to construct the segments, then a NURBS curve is obtained by selecting a suitable knot vector. The segments may meet with G^1 or C^1 continuity.

Obtaining the Bezier segments, $\boldsymbol{C}_i(u)$, requires the computation of the inner Bezier control points. These control points lie on the lines which are tangent to the curve at the $\mathbf{Q}_k$.

These tangent vectors, $\mathbf{T}_k$, are required at each $\mathbf{Q}_k$. The tangent vectors may be input with $\mathbf{Q}_k$, if not they must be calculated as part of the interpolation algorithm. There are a number of methods for obtaining $\mathbf{T}_k$.

Local Parabolic Curve Interpolation

Given $\{\boldsymbol{Q}_k, \boldsymbol{T}_k\}$, let L_k be a directed line defined by ($\mathbf{Q}_k$, $\mathbf{T}_k$), and $\mathbf{R}_k$ the intersection point of L_{k-1} and L_k.


$\mathbf{R}_3$
$\mathbf{R}_2$
$\mathbf{Q}_2$
$\mathbf{Q}_3$
$\mathbf{R}_4$
$\mathbf{Q}_1$
$\mathbf{Q}_4$
$\mathbf{R}_1$
$\mathbf{R}_5$
$\mathbf{Q}_5$
$\mathbf{Q}_0$

Assume for the moment that the intersection exists and that

$$\Upsilon_{k-1} > 0 \qquad \Upsilon_k < 0$$

where

$$\boldsymbol{R}_k = \boldsymbol{Q}_{k-1} + \Upsilon_{k-1}\boldsymbol{T}_{k-1}$$

$$\boldsymbol{R}_k = \boldsymbol{Q}_k + \Upsilon_k\boldsymbol{T}_k$$

These restrictions will be removed later.

The needed control points are

$$\boldsymbol{Q}_0, \boldsymbol{R}_1, \boldsymbol{Q}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_n, \boldsymbol{Q}_n$$

Let $\bar{u}_0 = 0$ and $\bar{u}_n = 1$. If $\{\bar{u}_i\}$, $i = 1, \ldots, n-1$, is a sequence of numbers satisfying $\bar{u}_{i-1} < \bar{u}_i < \bar{u}_{i+1}$, then the control points along with the knot vector

$$U = \{0, 0, 0, \bar{u}_1, \bar{u}_1, \ldots, \bar{u}_{n-1}, \bar{u}_{n-1}, 1, 1, 1\}$$

define a nonrational, G^1 continuous, quadratic B-spline curve interpolating $\{\boldsymbol{Q}_k\}$.

The choice of internal knots does not affect the shape of the curve, only the parameterization. It is possible to select knots such that the resulting curve is C^1 continuous. As a result the control points $\mathbf{Q}_1$, ..., $\mathbf{Q}_{n-1}$ and one occurrence of each of the interior knots can be removed.

The C^1 continuous curve interpolating the $\{\boldsymbol{Q}_k\}$ is then defined by the control points:

$$\boldsymbol{Q}_0, \boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_{n-1}, \boldsymbol{R}_n, \boldsymbol{Q}_n$$

and the knot vector:

$$U = \left\{ 0, 0, 0, \frac{\bar{u}_1}{\bar{u}_n}, \frac{\bar{u}_2}{\bar{u}_n}, \ldots, \frac{\bar{u}_{n-2}}{\bar{u}_n}, \frac{\bar{u}_{n-1}}{\bar{u}_n}, 1, 1, 1 \right\}$$

where,

$$\bar{u}_0 = 0$$

$$\bar{u}_1 = 1$$

$$\bar{u}_k = \bar{u}_{k-1} + (\bar{u}_{k-1} + \bar{u}_{k-2}) \frac{\left|\boldsymbol{R}_k - \boldsymbol{Q}_{k-1}\right|}{\left|\boldsymbol{Q}_{k-1} - \boldsymbol{R}_{k-1}\right|}$$

$$where \;\; k = 2, \ldots, n$$

Reparameterizing the curve to obtain C^1 continuity may result in a non-uniform parameterization.

Now the earlier restrictions will be handled by two special cases which arise when computing $\mathbf{R}_k$.

1. T_{k-1} and T_k are parallel, thus R_k can not be computed by intersection; this can indicate collinear segments, an inflection point, or a $180°$ turn in the curve

2. R_k can be computed, but Υ_{k-1} and Υ_k do not satisfy

$$\Upsilon_{k-1} > 0 \qquad \Upsilon_k < 0$$

this indicates either an inflection point or a turn of more than $180°$.

The collinear segments case applies if $\mathbf{T}_{k-1}$ and $\mathbf{T}_k$ are both parallel to the chord $\mathbf{Q}_{k-1}\mathbf{Q}_k$. This is handled by setting

$$\boldsymbol{R}_k = \frac{1}{2}(\boldsymbol{Q}_{k-1} + \boldsymbol{Q}_k)$$

All other special cases are handled by creating two parabolic segments between $\mathbf{Q}_{k-1}$ and $\mathbf{Q}_k$, instead of one. Three points must be computed

$$\boldsymbol{R}_k' = \boldsymbol{Q}_{k-1} + \Upsilon_k \boldsymbol{T}_{k-1}$$

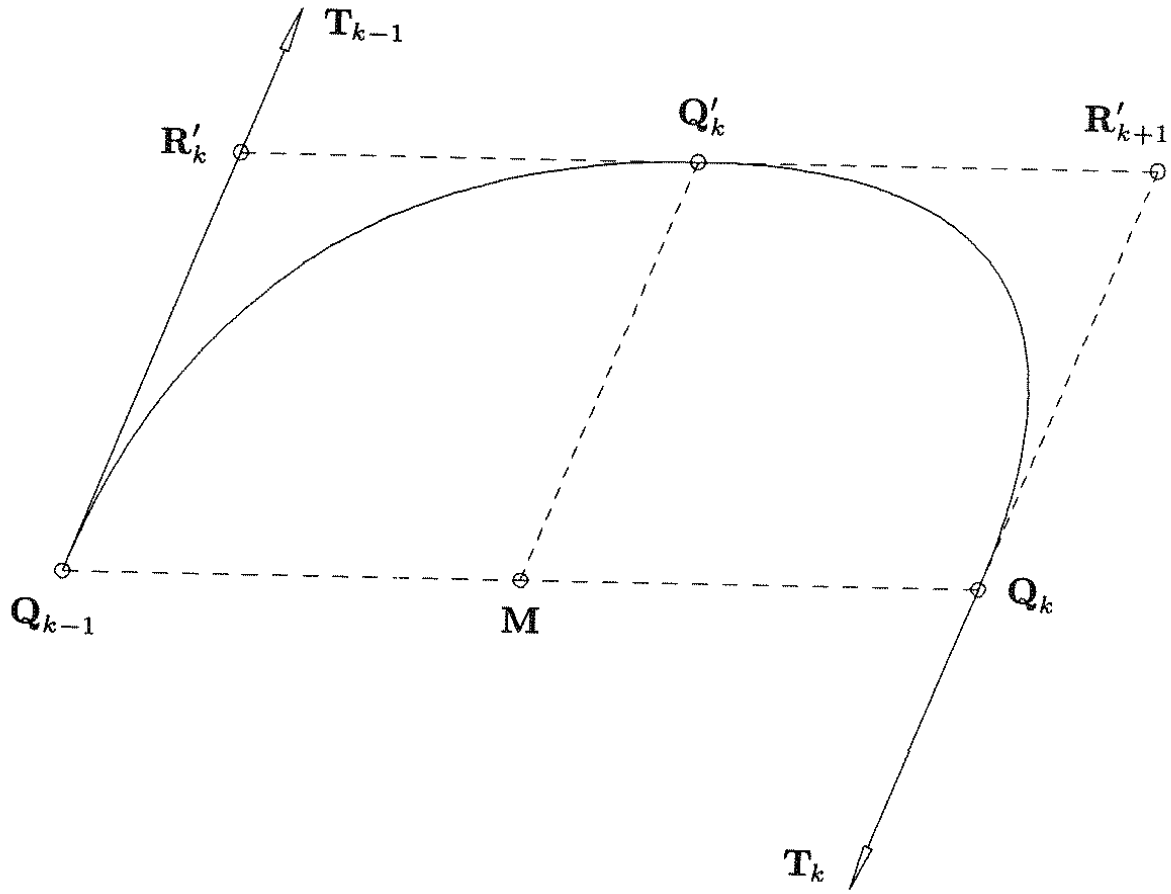
$$\boldsymbol{R}_{k+1}' = \boldsymbol{Q}_k - \Upsilon_{k+1} \boldsymbol{T}_k$$

$$\boldsymbol{Q}_{k+1}' = \frac{\Upsilon_k \boldsymbol{R}_{k+1}' + \Upsilon_{k+1} \boldsymbol{R}_k'}{\Upsilon_k + \Upsilon_{k+1}}$$

It remains to determine reasonable choices for Υ_k and Υ_{k+1}. If $\mathbf{T}_{k-1}$ and $\mathbf{T}_k$ are parallel (but not to $\mathbf{Q}_{k-1}\mathbf{Q}_k$), then set

$$\Upsilon_k = \Upsilon_{k+1} = \frac{1}{2}\left|\boldsymbol{Q}_{k-1}\boldsymbol{Q}_k\right|$$

This is illustrated in the following example.


$\mathbf{T}_{k-1}$
$\mathbf{R}'_k$
$\mathbf{Q}'_k$
$\mathbf{R}'_{k+1}$
$\mathbf{Q}_{k-1}$
$\mathbf{M}$
$\mathbf{Q}_k$
$\mathbf{T}_k$

Now consider the case when $\mathbf{T}_{k-1}$ and $\mathbf{T}_k$ are not parallel. Intuitively, Υ_k and Υ_{k+1} should depend on the angles θ_{k-1} and θ_k subtended by the chord $\mathbf{Q}_{k-1}\mathbf{Q}_k$ and the tangents $\mathbf{T}_{k-1}$ and $\mathbf{T}_k$ $(0 \le \theta_{k-1}, \theta_k \le 90°)$.

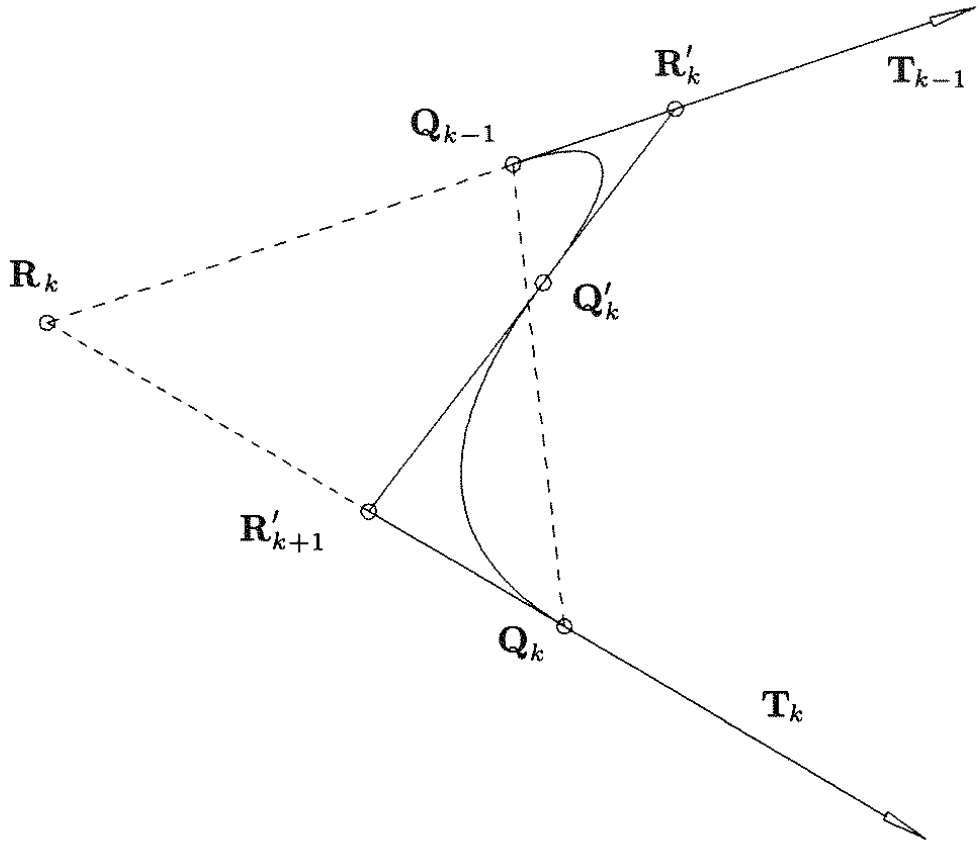
For this case:

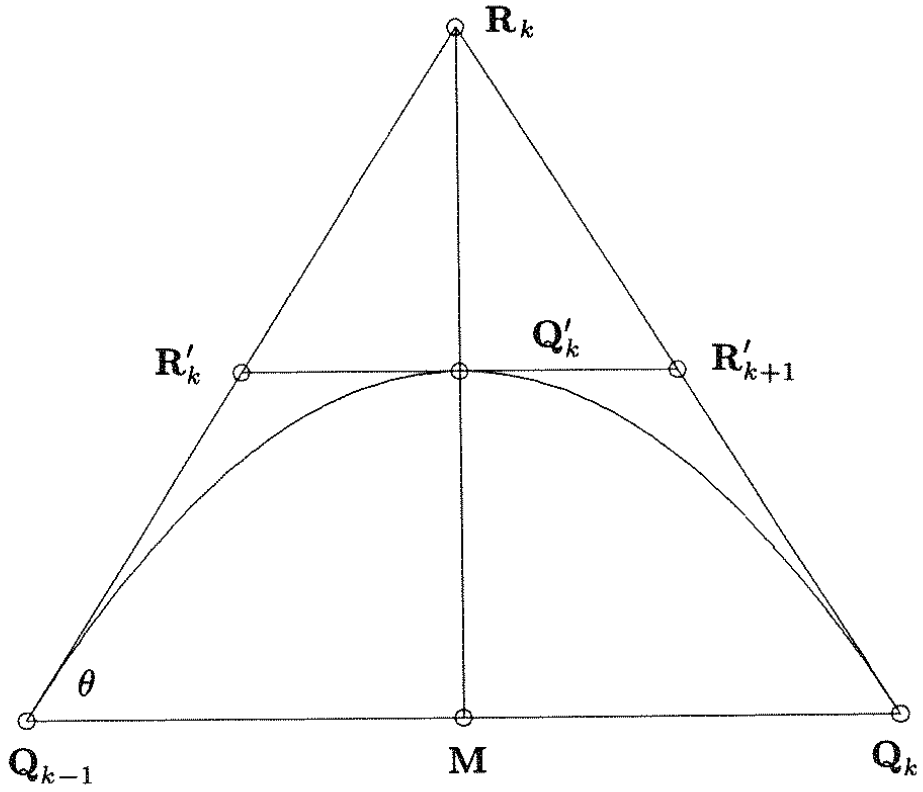
$$\Upsilon_k = \frac{1}{4} \frac{\left|\boldsymbol{Q}_{k-1}\boldsymbol{Q}_k\right|}{(\alpha \cos\theta_k + (1-\alpha)\cos\theta_{k-1})}$$

and

$$\Upsilon_k = \frac{1}{4} \frac{|\boldsymbol{Q}_{k-1}\boldsymbol{Q}_k|}{(\alpha \cos\theta_{k-1} + (1-\alpha)\cos\theta_k)}$$

where α is some constant between 0 and 1. Based on experimental observation, we choose $\alpha = 2/3$.


$\mathbf{R}'_k$
$\mathbf{T}_{k-1}$
$\mathbf{Q}_{k-1}$
$\mathbf{R}_k$
$\mathbf{Q}'_k$
$\mathbf{R}'_{k+1}$
$\mathbf{Q}_k$
$\mathbf{T}_k$


$\mathbf{R}_k$
$\mathbf{R}'_k$
$\mathbf{Q}'_k$
$\mathbf{R}'_{k+1}$
θ
$\mathbf{Q}_{k-1}$
$\mathbf{M}$
$\mathbf{Q}_k$

The preceeding method can be modified to produce rational quadratic curves. All weights at the $\mathbf{Q}_k$ are set to 1; weights at the $\mathbf{R}_k$ can be freely chosen. For example, some applications may require that the segments be more circular rather than parabolic.

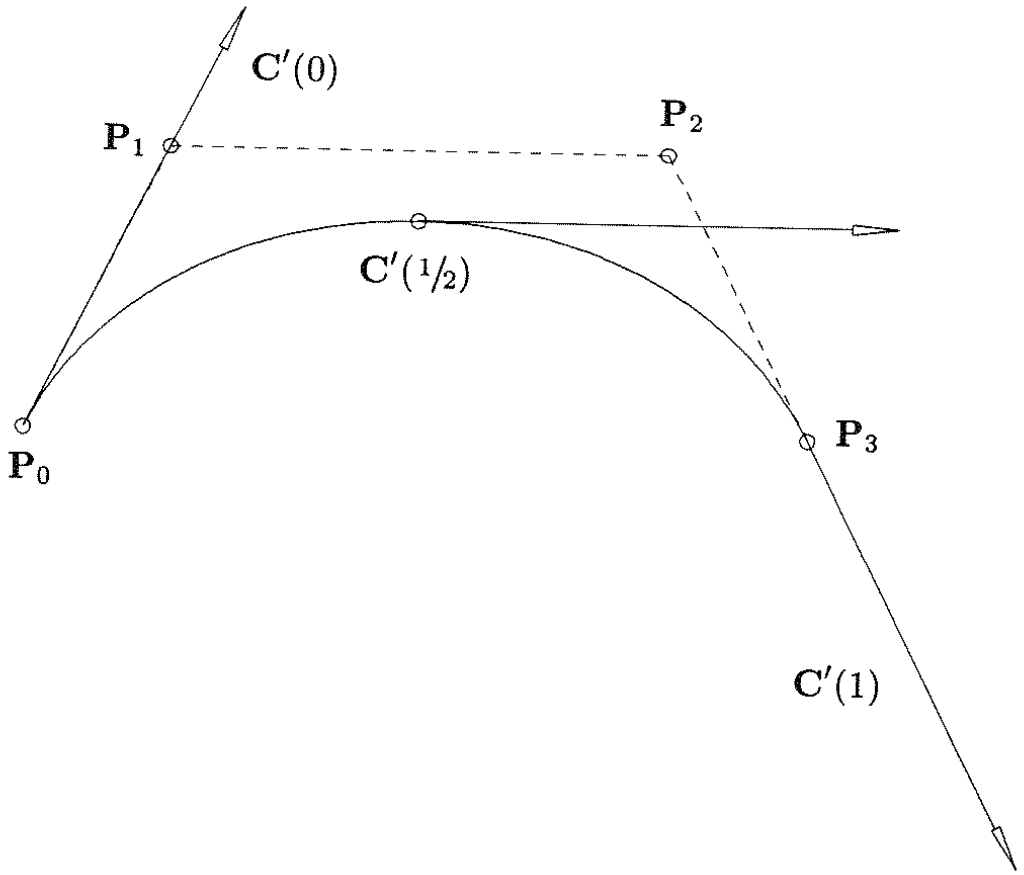
Local Cubic Curve Interpolation

Cubics easily handle three-dimensional data and inflection points without special treatment.

Let $\mathbf{P}_0$ and $\mathbf{P}_3$ be two endpoints, and $\mathbf{T}_0$ and $\mathbf{T}_3$ be the corresponding tangent directions with unit length. It is possible to construct a cubic Bezier curve, $\boldsymbol{C}(u), u \in [0,1]$, with these endpoints and tangent directions that satisfies:

$$\alpha = \left|\boldsymbol{C}'(0)\right| = \left|\boldsymbol{C}'\left(\frac{1}{2}\right)\right| = \left|\boldsymbol{C}'(1)\right|$$

As shown in the following figure.


$\mathbf{C}'(0)$
$\mathbf{P}_1$
$\mathbf{P}_2$
$\mathbf{C}'(1/2)$
$\mathbf{P}_3$
$\mathbf{P}_0$
$\mathbf{C}'(1)$

This implies:

$$\boldsymbol{P}_1 = \boldsymbol{P}_0 + \frac{1}{3}\alpha \boldsymbol{T}_0 \qquad \boldsymbol{P}_2 = \boldsymbol{P}_3 - \frac{1}{3}\alpha \boldsymbol{T}_3$$

Now applying the deCasteljau Algorithm at $u = 1/2$.

$$\boldsymbol{P}_1{}^2 - \boldsymbol{P}_0{}^3 = \frac{1}{8}(\boldsymbol{P}_3 + \boldsymbol{P}_2 - \boldsymbol{P}_1 - \boldsymbol{P}_0)$$

where, $\boldsymbol{P}_0^{\,3} = \boldsymbol{C}(1/2)$

It follows that:

$$\boldsymbol{C}'\left(\frac{1}{2}\right) = 6\left(\boldsymbol{P}_1^{\,2} - \boldsymbol{P}_0^{\,3}\right)$$

Therefore,

$$\frac{8}{6}\alpha = \left|\boldsymbol{P}_3 + \boldsymbol{P}_2 - \boldsymbol{P}_1 - \boldsymbol{P}_0\right|$$

$$= \left|\boldsymbol{P}_3 + \left(\boldsymbol{P}_3 - \frac{1}{3}\alpha\boldsymbol{T}_3\right) - \left(\boldsymbol{P}_0 + \frac{1}{3}\alpha\boldsymbol{T}_0\right) - \boldsymbol{P}_0\right|$$

which leads to

$$16\alpha^2 = \alpha^2 \left| \boldsymbol{T}_0 + \boldsymbol{T}_3 \right| - 12\alpha \left(\boldsymbol{P}_3 - \boldsymbol{P}_0 \right) \cdot \left(\boldsymbol{T}_0 + \boldsymbol{T}_3 \right) + 36 \left| \boldsymbol{P}_3 - \boldsymbol{P}_0 \right|^2$$

This equation has two real solutions for α, one positive and one negative. Substituting the positive solution into:

$$\boldsymbol{P}_1 = \boldsymbol{P}_0 + \frac{1}{3}\alpha \boldsymbol{T}_0 \qquad \boldsymbol{P}_2 = \boldsymbol{P}_3 - \frac{1}{3}\alpha \boldsymbol{T}_3$$

yields the desired $\mathbf{P}_1$ and $\mathbf{P}_2$.

Let $\{\mathbf{Q}_k\}$, k = 0, ..., *n*, be a set of three-dimensional data points. If tangent vectors are not given, compute them.

Construct a cubic Bezier curve segment, $\boldsymbol{C}_k(u)$, between each pair, $\mathbf{Q}_k$, $\mathbf{Q}_{k+1}$. Denote the Bezier control points by

$$\boldsymbol{P}_{k,0} = \boldsymbol{Q}_k \quad \boldsymbol{P}_{k,1} \quad \boldsymbol{P}_{k,2} \quad \boldsymbol{P}_{k,3} = \boldsymbol{Q}_{k+1}$$

We must determine suitable locations for $\mathbf{P}_{k,1}$ and $\mathbf{P}_{k,2}$ along $\mathbf{T}_k$ and $\mathbf{T}_{k+1}$. It is possible to obtain a C^1 continuous cubic and to achieve a good approximation to a uniform parameterization.

True uniform parameterization means constant speed over the entire parameter range. We construct a curve with equal speed at each $\mathbf{Q}_k$ and at the midpoint of each Bezier segment. Set $\bar{u}_0 = 0$. Now for k = 0, ..., *n* - 1, the $\bar{u}_{k+1}$ and the inner control points of $\boldsymbol{C}_k(u)$ are computed as follows:

1. Compute α to compute $\mathbf{P}_{k,1}$ and $\mathbf{P}_{k,2}$

2. Set $\bar{u}_{k+1} = \bar{u}_k + 3\left|\boldsymbol{P}_{k,1} - \boldsymbol{P}_{k,0}\right|$

This algorithm yields *n* Bezier segments, each having speed equal to 1 at their end- and midpoints with respect to their parameter ranges, $[\bar{u}_k, \bar{u}_{k+1}]$.

Thus a C^1 continuous cubic B-spline curve interpolating the $\mathbf{Q}_k$ is defined by the control points

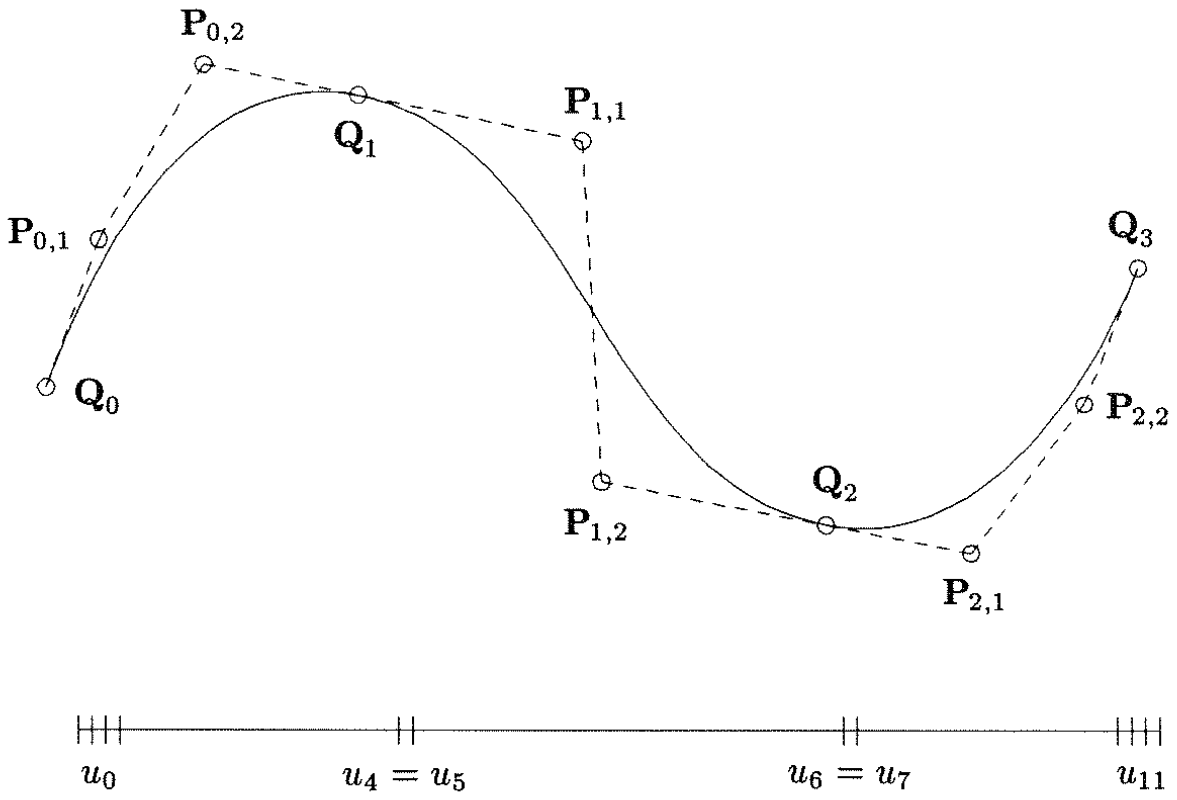
$$\boldsymbol{Q}_0, \boldsymbol{P}_{0,1}, \boldsymbol{P}_{0,2}, \boldsymbol{P}_{1,1}, \boldsymbol{P}_{1,2},$$
$$\ldots, \boldsymbol{P}_{n-2,2}, \boldsymbol{P}_{n-1,1}, \boldsymbol{P}_{n-1,2}, \boldsymbol{Q}_0$$

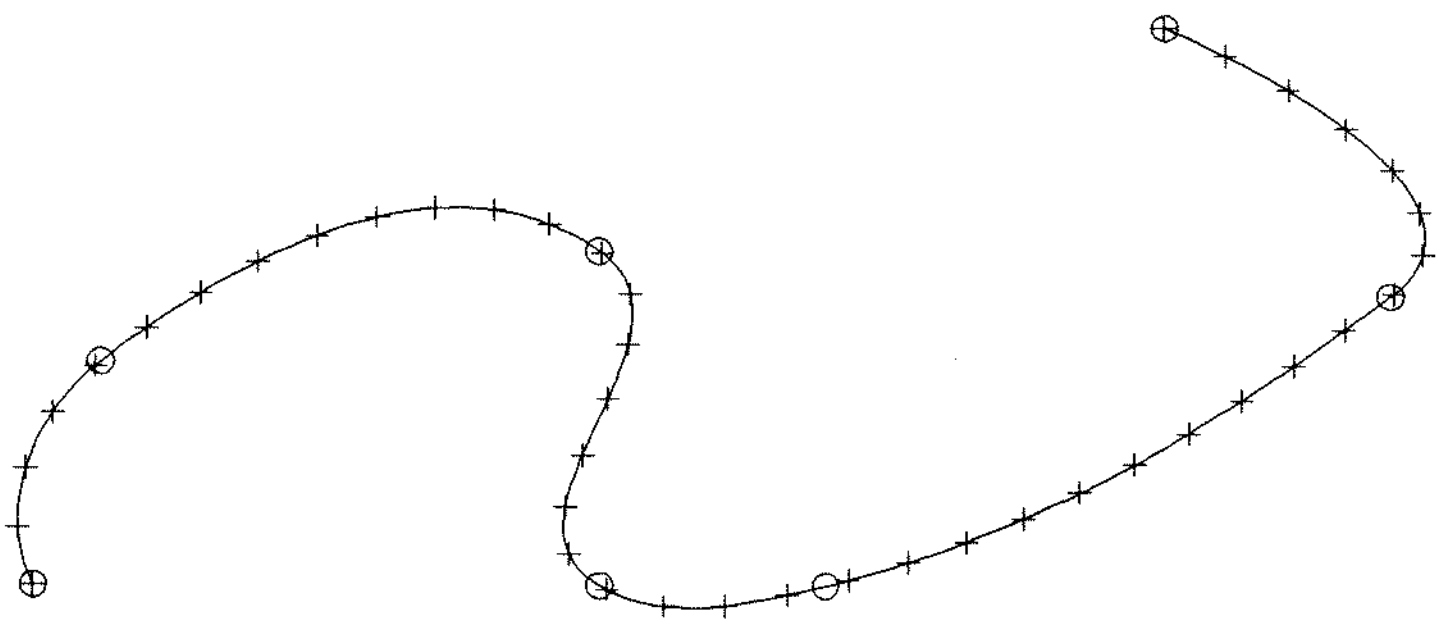
and the knots

$$U = \left\{ 0, 0, 0, 0, \frac{\bar{u}_1}{\bar{u}_n}, \frac{\bar{u}_1}{\bar{u}_n}, \frac{\bar{u}_2}{\bar{u}_n}, \frac{\bar{u}_2}{\bar{u}_n}, \right.$$

$$\left. \ldots, \frac{\bar{u}_{n-1}}{\bar{u}_n}, \frac{\bar{u}_{n-1}}{\bar{u}_n}, 1, 1, 1, 1 \right\}$$

The following figures show examples of cubic curve interpolation.


$\mathbf{P}_{0,2}$
$\mathbf{P}_{1,1}$
$\mathbf{Q}_1$
$\mathbf{P}_{0,1}$
$\mathbf{Q}_3$
$\mathbf{Q}_0$
$\mathbf{P}_{2,2}$
$\mathbf{Q}_2$
$\mathbf{P}_{1,2}$
$\mathbf{P}_{2,1}$
u_0
$u_4 = u_5$
$u_6 = u_7$
u_{11}



Curve Construction via Global Fitting

A well known method is least squares. Suppose $\boldsymbol{Q}_k$, $k = 0, \ldots, m$ are given. We want to produce a p-th degree nonrational B-spline curve with $(n+1)$ control points ($p < n < m$) (how to choose n???), satisfying the following conditions.

1. $\boldsymbol{Q}_0 = \boldsymbol{P}_0$ and $\boldsymbol{Q}_m = \boldsymbol{P}_n$.

2. The curve passes close to $\boldsymbol{Q}_k$,
   k = 1, ..., m -1, in the least squares sense.

The steps are as follows:

1. Compute the data point parameter assignments $\bar{u}_k$, k = 0, ..., m using accumulated chord length or the centripetal method.

2. For the knot vector, choose interior knots so that each knot span contains roughly the same number of $\bar{u}_k$.

3. Set up the constraint equation in partitioned form:

$$\begin{bmatrix} \boldsymbol{Q}_U \\ \boldsymbol{Q}_C \end{bmatrix} = \begin{bmatrix} N_U & N_C \\ 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{P}_U \\ \boldsymbol{P}_C \end{bmatrix}$$

where subscript *U* denotes unconstrained, and *C*, constrained, i.e.,

$$\boldsymbol{Q}_U = [\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{m-1}]^T$$

$$\boldsymbol{Q}_C = [\boldsymbol{Q}_0, \boldsymbol{Q}_m]^T$$

$$\boldsymbol{P}_U = [\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{n-1}]^T$$

$$\boldsymbol{P}_C = [\boldsymbol{P}_0, \boldsymbol{P}_n]^T$$

N_U is the $(m-1) \times (n-1)$

matrix of basis functions $N_{i,p}(\bar{u}_k)$

$i = 1, \ldots, n-1$ and $k = 1, \ldots, m-1$

N_C is the $(m-1) \times 2$

matrix of basis functions $N_{i,p}(\bar{u}_k)$

$i = 0, n$ and $k = 1, \ldots, m-1$

I is the 2×2 identity matrix

4. Now since $\boldsymbol{Q}_C = \boldsymbol{P}_C$, it follows that

$N_U \boldsymbol{P}_U = \boldsymbol{Q}_U - N_C \boldsymbol{Q}_C$ which implies,

$$(N_U^T N_U)\,\boldsymbol{P}_U = N_U^T\,(\boldsymbol{Q}_U - N_C \boldsymbol{Q}_C)$$

5. Solve this equation. It is an (n - 1) $\times$ (n -1) system of linear equations in the unknowns, $\boldsymbol{P}_C$, with (invertible) coefficient matrix $N_U^T N_U$,

and right-hand side $N_U^T (\boldsymbol{Q}_U - N_C \boldsymbol{Q}_C)$

Example: Let $m = 3$ and $n = p = 2$. Then,

$$\boldsymbol{Q}_U = \begin{bmatrix} \boldsymbol{Q}_1 \\ \boldsymbol{Q}_2 \end{bmatrix} \qquad \boldsymbol{P}_U = \begin{bmatrix} \boldsymbol{P}_1 \end{bmatrix}$$

$$N_U = \begin{bmatrix} N_{1,2}(\bar{u}_1) \\ N_{1,2}(\bar{u}_2) \end{bmatrix}$$

$$N_C = \begin{bmatrix} N_{0,2}(\bar{u}_1) & N_{2,2}(\bar{u}_1) \\ N_{0,2}(\bar{u}_2) & N_{2,2}(\bar{u}_2) \end{bmatrix}$$

and from the above formulation we have,

$$\begin{bmatrix} N_{1,2}(\bar{u}_1) & N_{1,2}(\bar{u}_2) \end{bmatrix} \begin{bmatrix} N_{1,2}(\bar{u}_1) \\ N_{1,2}(\bar{u}_2) \end{bmatrix} \boldsymbol{P}_U =$$

$$\begin{bmatrix} N_{1,2}(\bar{u}_1) & N_{1,2}(\bar{u}_2) \end{bmatrix} \left(\begin{bmatrix} \boldsymbol{Q}_1 \\ \boldsymbol{Q}_2 \end{bmatrix} - \begin{bmatrix} N_{0,2}(\bar{u}_1) & N_{2,2}(\bar{u}_1) \\ N_{0,2}(\bar{u}_2) & N_{2,2}(\bar{u}_2) \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_0 \\ \boldsymbol{Q}_3 \end{bmatrix} \right)$$

which implies,

$$\boldsymbol{P}_U =$$

$$= [N_{1,2}(\bar{u}_1)\,(\boldsymbol{Q}_1 - N_{0,2}(\bar{u}_1)\,\boldsymbol{Q}_0 - N_{2,2}(\bar{u}_1)\,\boldsymbol{Q}_3)$$

$$+ N_{1,2}(\bar{u}_2)\,(\boldsymbol{Q}_2 - N_{0,2}(\bar{u}_2)\,\boldsymbol{Q}_0 - N_{2,2}(\bar{u}_2)\,\boldsymbol{Q}_3)\,] /$$

$$[\,(N_{1,2}(\bar{u}_1))^2 + (N_{1,2}(\bar{u}_2))^2]$$

Convince yourself that it works by choosing some examples points!

Weighted and Constrained Least Squares Curve Fitting

Let $\{\mathbf{Q}_i\}$, i = 0, ..., r, be the points to be approximated. In addition the first derivative $\mathbf{D}_i$ at any $\mathbf{Q}_i$ can be specified. Let $\{\mathbf{D}_{i(j)}\}$, j = 0, ..., s, be the set of derivatives; $-1 \leq s \leq r$, where s = -1, means no derivatives are specified. Any $\mathbf{Q}_i$ or $\mathbf{D}_{i(j)}$ can be constrained.

The unconstrained items are denoted by $\boldsymbol{Q}^{u}_{i(0)}, \ldots, \boldsymbol{Q}^{u}_{i(r_u)}$ and $\boldsymbol{D}^{u}_{i(0)}, \ldots, \boldsymbol{D}^{u}_{i(s_u)}$.
The constrained items are denoted by $\boldsymbol{Q}^{c}_{i(0)}, \ldots, \boldsymbol{Q}^{c}_{i(r_c)}$ and $\boldsymbol{D}^{c}_{i(0)}, \ldots, \boldsymbol{D}^{c}_{i(s_c)}$.

Note that s_u, s_c or r_c equal to -1 means no data corresponding to that index.

In addition note that $r = r_u + r_c + 1$ and $s = s_u + s_c + 1$.

A positive weight can also be assigned to each nonconstrained item, $w^q_{i(0)}, \ldots, w^q_{i(r_u)}$ and $w^d_{i(0)}, \ldots, w^d_{i(s_u)}$.

These weights allow additional influence over the tightness of the approximation to each data item relative to its neighbors. $w^q_i, w^d_i = 1$ is the default.

Increasing the weight increases the tightness of the approximation to that item, decreasing the weight loosens the approximation to that item.

Notice that these weights have nothing to do with weights in the NURBS sense.

Now let $m_u = r_u + s_u + 1$ and $m_c = r_c + s_c + 1$.

We want to approximate the unconstrained data in the least squares sense and interpolate the constrained data with a *p*th degree nonrational curve, **C**(u), with n + 1 control points.

Assume

$$m_c < n \qquad m_c + n < m_u + 1$$

Now let $\mathbf{S}_k$, k = 0, ..., m_u, be the *k*th unconstrained data item. $\mathbf{T}_k$, k = 0, ..., m_c, be the *k*th constrained data item. Finally, w_k, k = 0,... ,m_u, be the *k*th weight.

Define the following vectors/matrices

$$\boldsymbol{S} = [\boldsymbol{S}_k] \quad a\ vector\ of\ m_u + 1\ elements$$

$$\boldsymbol{T} = [\boldsymbol{T}_k] \quad a\ vector\ of\ m_c + 1\ elements$$

$$W = [w_k] \quad an \quad (m_u + 1) \times (m_u + 1)$$

$$diagonal\ matrix$$

$$\boldsymbol{P} = [\boldsymbol{P}_k] \quad the \ vector \ of \ n+1$$

$$unknown \ control \ points$$

$$N = [ND_{i,p}(\bar{u}_k)]$$

where $ND_{i,p}(\bar{u}_k)$ is the *i*th basis function or its first derivative, evaluated at $\bar{u}_k$, which is a parameter value corresponding to an unconstrained data item, this is a $(m_u + 1) \times (n + 1)$ matrix of scalars.

$$M = [MD_{i,p}(\bar{u}_k)]$$

where $MD_{i,p}(\bar{u}_k)$ is the *i*th basis function or its first derivative, evaluated at $\bar{u}_k$, which is a parameter value corresponding to an constrained data item, this is a $(m_c + 1) \times (n + 1)$ matrix of scalars.

Compute the data point parameter assignments $\bar{u}_k$, k = 0, ..., m using accumulated chord length or the centripetal method.

The knots u_j can be found using

$$d = \frac{r+1}{n-p+1}$$

$$i = int(jd) \qquad \alpha = jd - i$$

$$u_{p+j} = (1-\alpha)\,\bar{u}_{i-1} + \alpha \bar{u}_i$$

$$j = 1, \ldots, n-p$$

Determining the curve is a constrained minimization problem, involving n + 1 unknowns and m_c + 1 constraints.

The standard method of solution is to use Lagrange multipliers. This yields a partitioned matrix of the form.

$$\begin{bmatrix} N^T W \boldsymbol{S} \\ \boldsymbol{T} \end{bmatrix} = \begin{bmatrix} N^T W N & M^T \\ M & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{P} \\ \boldsymbol{A} \end{bmatrix}$$

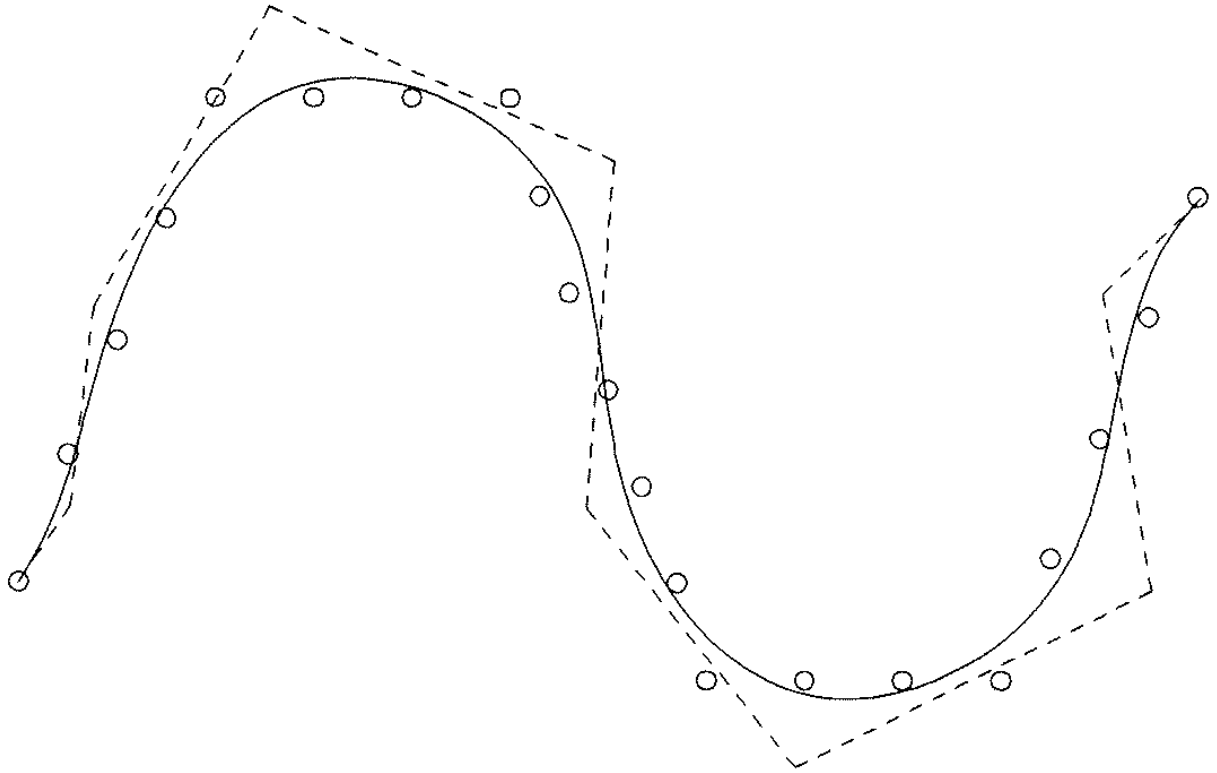
There is a unique solution if $N^T WN$ and $M(N^T WN)^{-1}M^T$ are both invertible.

$$\boldsymbol{P} = \left(N^T WN\right)^{-1} N^T W\boldsymbol{S} - \left(N^T WN\right)^{-1} M^T \boldsymbol{A}$$

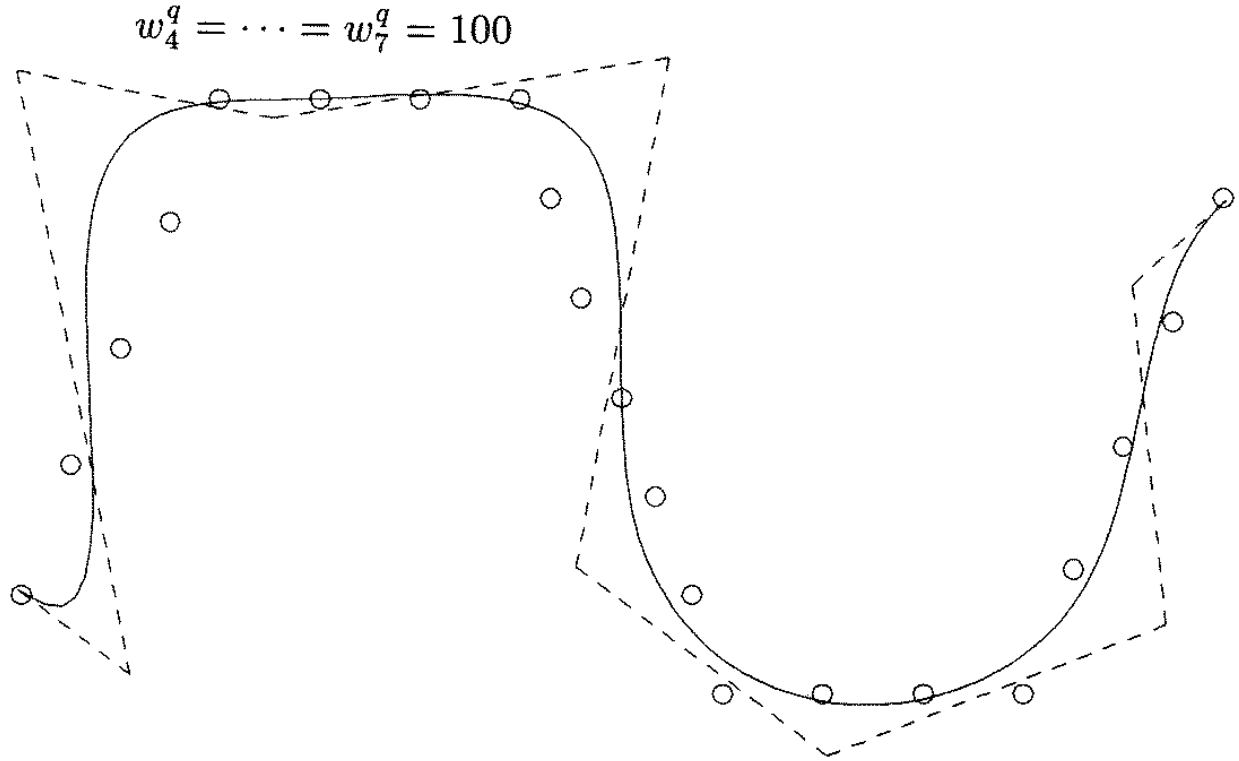
$$\boldsymbol{A} = \left(M\left(N^T WN\right)^{-1} M^T\right)^{-1}\left(M\left(N^T WN\right)^{-1} N^T W\boldsymbol{S} - \boldsymbol{T}\right)$$

See algorithm A9.6.

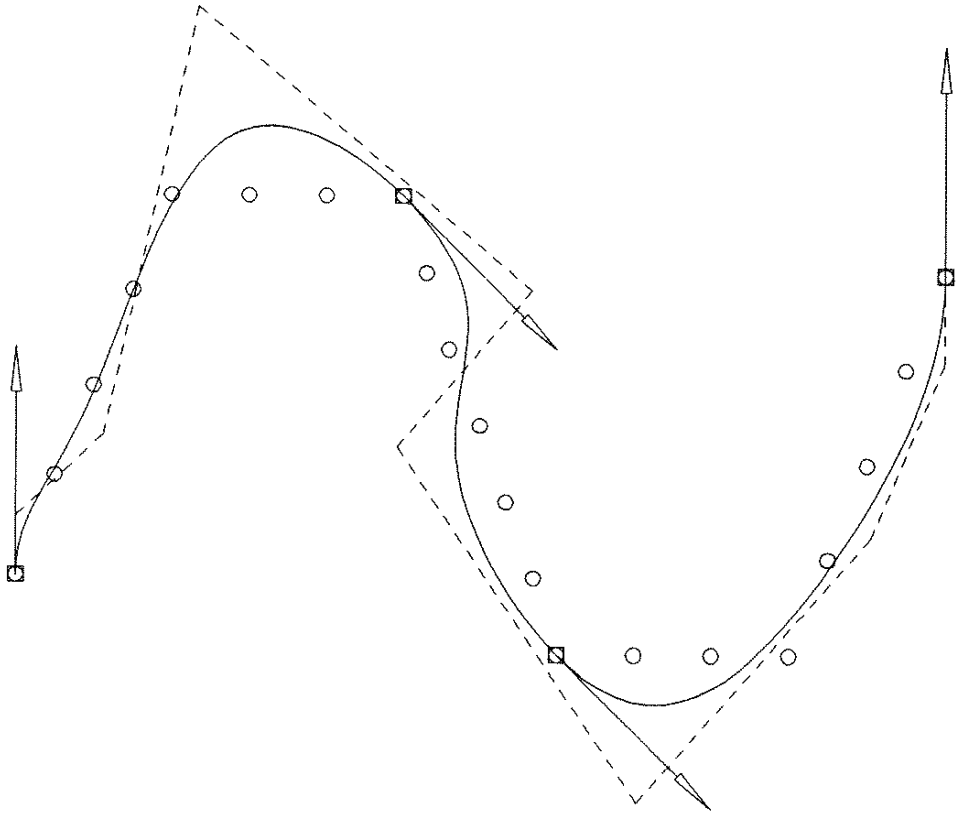
Unconstrained fit with all weights equal to one; note curve does not pass through endpoints.



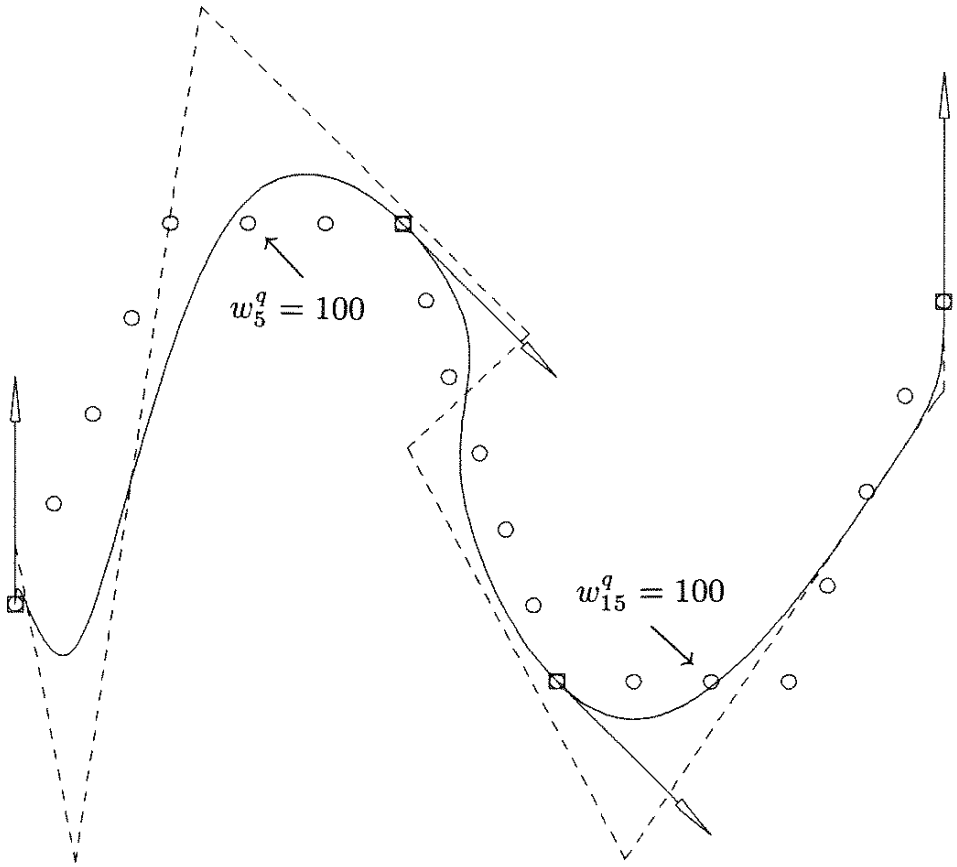
Unconstrained fit.



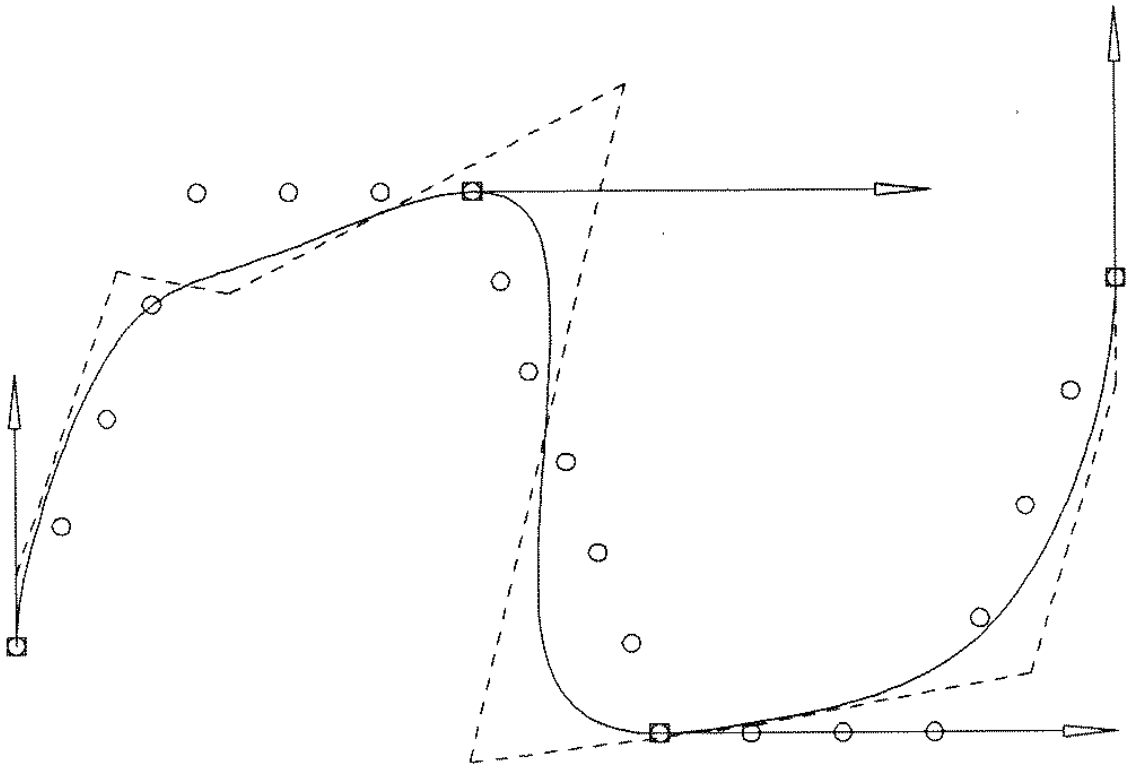
Constrained fit; point constraints marked with squares, tangent constraints by arrows.



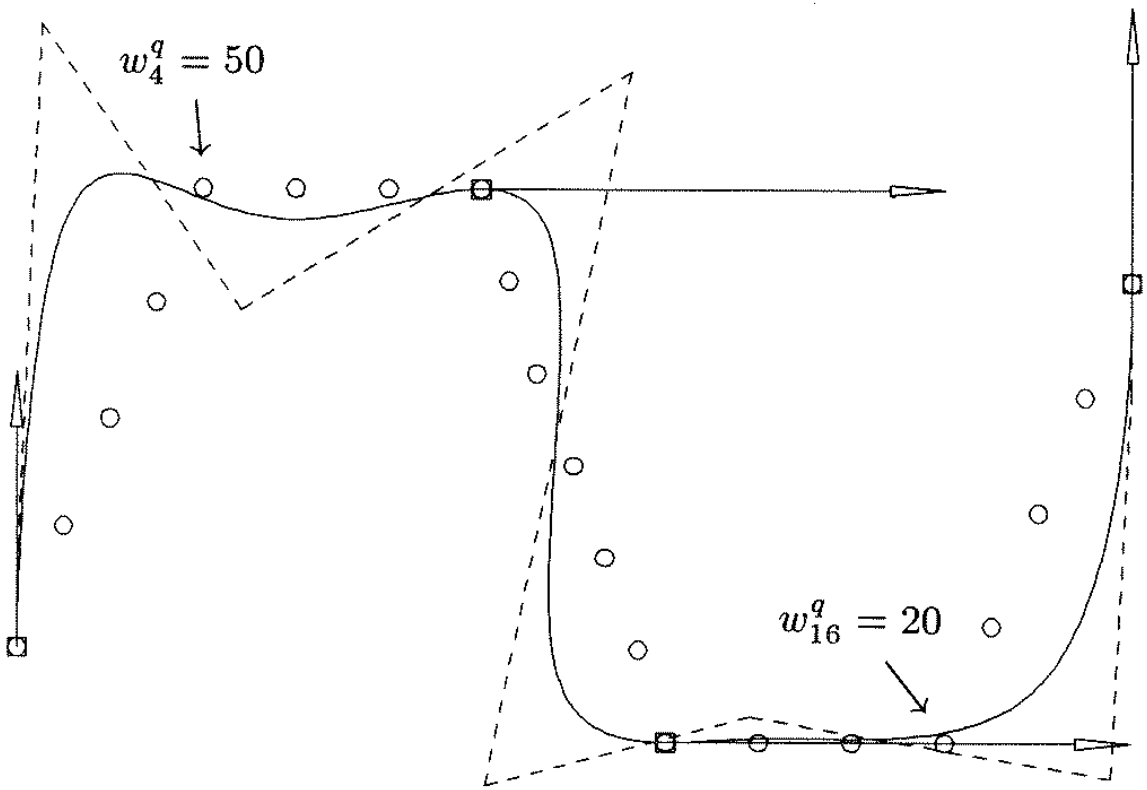
Constrained fit.



Two new tangent constraints.



Two new weights.



Least Squares Surface Approximation

Although it is possible to set up and solve a general least squares surface fitting problem, linear or nonlinear, and with or without weights and constraints, the task is more complex than for curves.

The text presents a surface approximation scheme which builds upon the least squares curve scheme. The approach is simple but adequate for most applications.

See algorithm A9.7.