Interpolation, Fitting and Cross-Sectional Design

Two commonly used curve and surface design techniques are interpolation and fitting. Given a set of constraints, typically points and sometimes derivative or tangent (i.e., unit length) vectors then,

- interpolation means construct a curve (or surface) which passes through the points, and assumes the specified derivative constraints, i.e., the constraints are satisfied precisely.

- fitting means construct a curve (or surface) which approximates (by some measure) the constraints to some tolerance.

Interpolation and fitting methods can be either global or local:

- Global. Generally, a global system of equations is set up and solved. A change in any one constraint may change the entire curve (or surface) shape (although the magnitude of the constraint generally falls off with increasing distance from the affected constraint).

- Local. These algorithms are generally more constructive (geometric) in nature, constructing the curve (or surface) segment-wise, using only local constraints for each step. A change in (or addition of) a constraint only changes the curve (or surface) locally.

Local methods are usually computationally less expensive than global methods. They can also deal with cusps and other local anomalies better; however, achieving desired levels of continuity at segment boundaries can be difficult.

Another issue, when using NURBS, is whether to use only nonrational curves (all weights set to 1), or rational (and if rational, how to set the weights?)

Curve Construction via Global Interpolation

Given a set of data points, $\boldsymbol{Q}_k$, $k = 0, ..., n$, we seek a p-th degree nonrational B-spline curve $\boldsymbol{C}(u)$, which satisfies the $n + 1$ constraints:

$$\boldsymbol{Q}_k = \boldsymbol{C}(\bar{u}_k) = \sum_{i=0}^{n} N_{i,p}(\bar{u}_k)\boldsymbol{P}_i$$

for some parameter values $\bar{u}_k$, $k = 0, ..., n$.

Some remarks about this equation:

- This equation results in a $(n + 1) \times (n + 1)$ system of linear equations.

- There are, of course, infinitely many curves satisfying this equation; some may be “okay” and some may not be.

- We use a nonrational curve. Trying to use a rational curve in a global interpolation is difficult, as it leads to nonlinear constraints.

- This equation is independent of the number of coordinates; it can interpolate 2D, 3D or even 4D (homogeneous) data.

There are actually four unknowns in the equation: the degree p, the data point parameter assignments $\bar{u}_k$, the knot vector U, and the control points $\boldsymbol{P}_i$. Clearly, p, $\bar{u}_k$ and U must all be determined (chosen) before the system of equation can be set up and solved for the $\boldsymbol{P}_i$; and their choice has a very definite effect on the shape of the resulting curve.

There are a number of (mostly heuristic) methods for choosing the $\bar{u}_k$ and U; here are a few:

1. Equally spaced $\bar{u}_k$ and equally spaced knots u_j. In particular,

$$\bar{u}_k = \frac{k}{n} \quad , \quad k = 0, \ldots, n$$

$$U = \{\underbrace{0, \ldots, 0}_{p+1}, u_{p+1} = \frac{1}{n-p+1},$$

$$u_{p+2} = \frac{2}{n-p+1}, \ldots,$$

$$u_n = \frac{n-p}{n-p+1}, \underbrace{1, \ldots, 1}_{p+1}\}$$

This method is notoriously bad, especially if the $\boldsymbol{Q}_k$ are even a little bit unevenly spaced. Furthermore, the system of equations may be singular.

2. Compute the $\bar{u}_k$ according to accumulated chord length, and the u_j by averaging the $\bar{u}_k$. Specifically: first compute,

$$d = \sum_{k=1}^{n} \left| \boldsymbol{Q}_k - \boldsymbol{Q}_{k-1} \right|$$

Then,

$$\bar{u}_0 = 0, \quad \bar{u}_n = 1$$

$$\bar{u}_k = \bar{u}_{k-1} + \frac{1}{d} \left| \boldsymbol{Q}_k - \boldsymbol{Q}_{k-1} \right|$$

$$, k = 1, \ldots, n-1$$

and,

$$U = \{0, \ldots, 0, u_1, \ldots, u_{n-p}, 1, \ldots, 1\}$$

, where

$$u_j = \frac{1}{p} \sum_{i=j}^{j+p-1} \bar{u}_i, \qquad \text{for} \qquad j = 1, \ldots, n-p$$

This is probably the most widely used method, and is generally adequate.

3. The centripetal method for computing the $\bar{u}_k$. Instead of assigning parameters according to accumulated chord length, use a fractional power, e.g., let,

$$a = \sum_{k=1}^{n} \sqrt{\left|\boldsymbol{Q}_k - \boldsymbol{Q}_{k-1}\right|}$$

then,

$$\bar{u}_0 = 0, \quad \bar{u}_n = 1$$

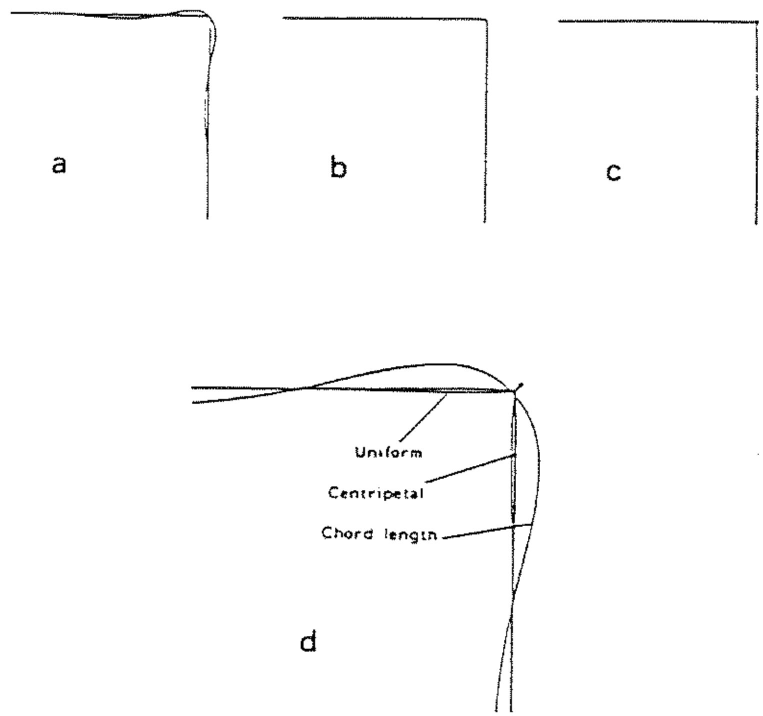
$$\bar{u}_k = \bar{u}_{k-1} + \frac{1}{a}\sqrt{\left|\boldsymbol{Q}_k - \boldsymbol{Q}_{k-1}\right|}$$

$$, k = 1, \ldots, n-1$$

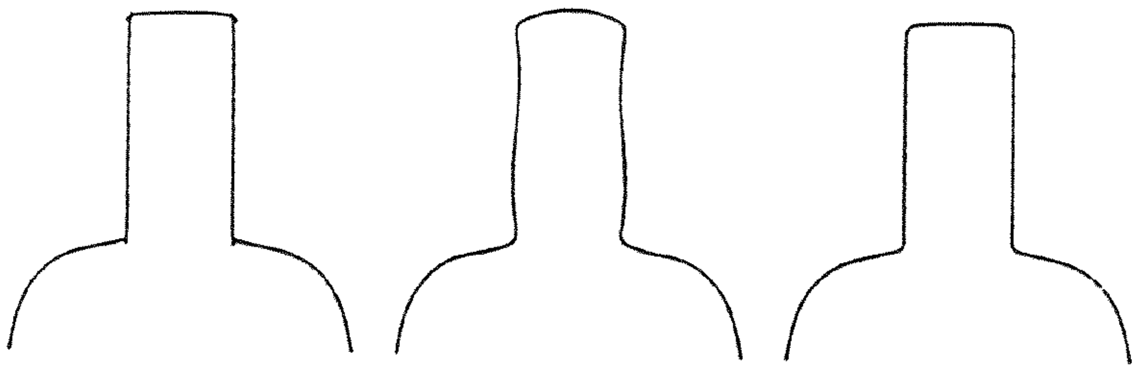
The knots u_j are then computed via parameter averaging, as before. The centripetal method has an advantage over the chord length method when the curve turns sharply.

Using methods 2 or 3 above yields a system of linear equations in $n + 1$ unknowns. The coefficient matrix is totally positive and banded with bandwidth less than p. Hence it can be solved safely by Gaussian elimination without pivoting.

Examples



Natural cubic splines. Data: $\{x, y\} = \{(0, 41),$ $(10, 41), (31, 41), (40.8, 41), (41, 40.8), (41, 31), (41, 10),$ $(41, 0)\}$. (a) shows the chord length parametrization and the centripetal model superimposed; (b) the centripetal model; (c) the uniform parametrization; (d) shows the different behaviours near the corner



Natural cubic splines through 18 points: $\{x, y\} =$ $\{(0, 0), (1.34, 5), (5, 8.66), (10, 10), (10.6, 10.4), (10.7, 12), (10.7, 28.6), (10.8, 30.2), (11.4, 30.6), (19.6, 30.6), (20.2, 30.2), (20.3, 28.6), (20.3, 12), (20.4, 10.4), (21, 10), (26, 8.66), (29.66, 5), (31, 0)\}$. *Note (a) contains several loops; (b) might even be preferred on artistic grounds, but the shape is not the one intended. (a) Uniform, (b) chord length, (c) centripetal*

In addition to points, first derivative or tangent vectors may be given at each point.

Assume derivatives, $\boldsymbol{D}_k$, are given. Then we have $2(n + 1)$ constraints and will have that many control points to solve for. The systems of equations are:

$$\boldsymbol{Q}_k = \boldsymbol{C}(\bar{u}_k) = \sum_{i=0}^{2n+1} N_{i,p}(\bar{u}_k)\boldsymbol{P}_i \qquad \text{(A)}$$

$$\boldsymbol{D}_k = \boldsymbol{C}'(\bar{u}_k)$$

$$= p\sum_{i=0}^{2n} N_{i,p-1}(\bar{u}_k)\frac{\boldsymbol{P}_{i+1}-\boldsymbol{P}_i}{u_{i+p+1}-u_{i+1}} \qquad \text{(B)}$$

The $\bar{u}_k$ can be chosen as above, or we can compute a parametrization even closer to the true arc length by making use of the $\boldsymbol{D}_k$ (e.g., fit a parabolic arc between two neighboring points and approximate its arc length).

The knot vector is obtained as follows. We need $2(n + 1) + p + 1$ knots. We consider the cases $p = 2$ and $p = 3$ (most frequent cases). If $p = 2$, then choose,

$$U = \{0, 0, 0, \frac{\bar{u}_1}{2}, \bar{u}_1, \frac{\bar{u}_1 + \bar{u}_2}{2}, \bar{u}_2, \ldots, 1, 1, 1$$

$$, \ldots, \frac{\bar{u}_{n-1} + 1}{2}, 1, 1, 1 \}$$

if $p = 3$, then,

$$U = \{0, 0, 0, 0, \frac{\bar{u}_1}{2}, \frac{2\bar{u}_1 + \bar{u}_2}{3}, \frac{\bar{u}_1 + 2\bar{u}_2}{3}$$

$$, \ldots, \frac{\bar{u}_{n-2} + 2\bar{u}_{n-1}}{3}, \frac{\bar{u}_{n-1} + 1}{2}, 1, 1, 1, 1\}$$

If we merge equations (A) and (B) in an alternating fashion, the resulting $2(n + 1) \times 2(n + 1)$ system is banded and can be solved without pivoting.

If only the tangent vectors, $\boldsymbol{T}_k$, are given instead of $\boldsymbol{D}_k$, then the magnitudes will have to be computed (to obtain the $\boldsymbol{D}_k$ from the $\boldsymbol{T}_k$). A method to do this follows:

- Compute the $\bar{u}_k$ (e.g., by approximating the arc length of parabolic arcs between the $\boldsymbol{Q}_k$).

- Let s be the total computed arc length.

- Then set $\boldsymbol{D}_k = s\boldsymbol{T}_k$ for all k (i.e., this assumes a uniform parametrization of the curve).

- If the resulting curve is not satisfactory, s may be interactively and iteratively adjusted (increasing s produces a “fuller” curve).

Next consider rational interpolation. This is important, for example, in skinning. If we are given weighted control points $\boldsymbol{Q}_k^w$, $k = 0, ..., n$, we apply the above formulation in 4D (since it is independent of the number of coordinates.

Of course, using the chordal arc length approximation for parameter assignment implies that we are computing the curve based on its arc length of its unprojected (4D) counterpart, but in practice this seems to work quite well.

Finally, suppose both $\boldsymbol{Q}_k^w$ and $\boldsymbol{D}_k$ are given (if $\boldsymbol{T}_k$ are given, the $\boldsymbol{D}_k$ must be derived from them, as above). Then to use the derivative constraints above, we must estimate the $\boldsymbol{D}_k^w$; i.e., the 4D derivatives. So, from the

$$\boldsymbol{D}_k = (\dot{x}_k, \dot{y}_k, \dot{z}_k)$$

we need to find the,

$$\boldsymbol{D}_k^w = ((w_k x_k)', (w_k y_k)', (w_k z_k)', w'_k)$$

Since,

$$(w_k x_k)' = \dot{w}_k x_k + w_k \dot{x}_k$$

we need to compute $\dot{w}_k$ only. We interpolate a 1D spline $w(u)$ through the data points w_k, $k = 0, ..., n$, so that $w(\bar{u}_k) = w_k$. From this we have,

$$\dot{w}_k = \dot{w}(u)\big|_{u = \bar{u}_k}$$

Then proceed as above.

Surface Construction via Global Interpolation

Given a set of (n + 1) x (m + 1) data points $\mathbf{Q}_{k,l}$, $k = 0, \ldots, n$ and $l = 0, \ldots, m$, that are to be interpolated to construct a (p, q) th degree nonrational B-spline surface.

$$
\begin{aligned}
\boldsymbol{Q}_{k,l} &= \boldsymbol{S}(\bar{u}_k, \bar{v}_l) \\
&= \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(\bar{u}_k) N_{j,q}(\bar{v}_l) \boldsymbol{P}_{i,j}
\end{aligned}
$$

The first step is to compute reasonable value for $(\bar{u}_k, \bar{v}_l)$ and the knot vectors ***U*** and ***V***.

Using chord length or centripetal method presented in global curve interpolation section compute $\bar{u}_o^{\,l}, \ldots, \bar{u}_n^{\,l}$ for each *l*, and then average these results for $l = 0, \ldots, m$, that is

$$\bar{u}_k = \frac{1}{m+1} \sum_{l=0}^{m} \bar{u}_k^{\,l} \qquad k = 0, \ldots, n$$

The same procedure can be used to find $\bar{v}_l$.

See algorithm A9.3.

Once the $(\bar{u}_k, \bar{v}_l)$ are computed, the knot vectors ***U*** and ***V*** can be obtained using techniques from global curve interpolation.

Now to compute the control points.

This equation,

$$\boldsymbol{Q}_{k,l} = \boldsymbol{S}(\bar{u}_k, \bar{v}_l)$$

$$= \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(\bar{u}_k) N_{j,q}(\bar{v}_l) \boldsymbol{P}_{i,j}$$

clearly represents a (n + 1) x (m + 1) system of linear equations in the unknowns $\boldsymbol{P}_{i,j}$. However, since $\boldsymbol{S}(u, v)$ is a tensor product surface the unknowns can be obtained by a sequence of curve interpolations.

For fixed *l*,

$$\boldsymbol{Q}_{k,l} = \sum_{i=0}^{n} N_{i,p}(\bar{u}_k)\boldsymbol{R}_{i,l} \qquad \text{(a)}$$

where

$$\boldsymbol{R}_{i,l} = \sum_{j=0}^{m} N_{j,q}(\bar{v}_l)\boldsymbol{P}_{i,j} \qquad \text{(b)}$$

Notice that (a) is just curve interpolation through points $\boldsymbol{Q}_{k,\,l}$, $k = 0, \ldots, n$. The $\boldsymbol{R}_{i,\,l}$ are the control points of the isoparametric curve on $\boldsymbol{S}(u, v)$ at fixed $v = \bar{v}_l$.

Now fixing *i* and letting *l* vary, (b) is curve interpolation through points $\boldsymbol{R}_{i,\,0}, \ldots, \boldsymbol{R}_{i,\,m}$ with $\boldsymbol{P}_{i,\,0}, \ldots, \boldsymbol{P}_{i,\,m}$ as the computed control points.

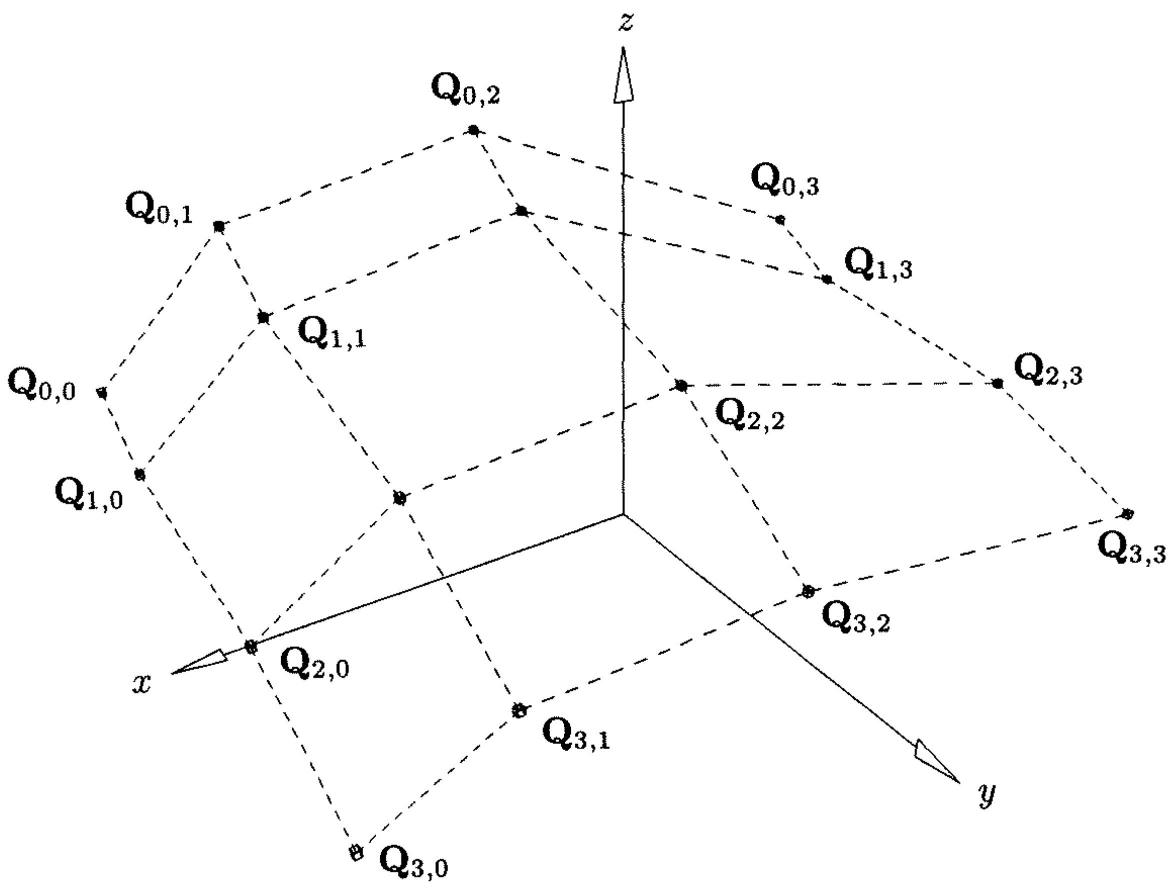
Thus, the algorithm to obtain all $\boldsymbol{P}_{i,j}$ is:

1. using ***U*** and $\bar{u}_k$, do *m* + 1 curve interpolations through $\boldsymbol{Q}_{0,l}, \ldots, \boldsymbol{Q}_{n,l}$ (for $l = 0, \ldots, m$); this yields $\boldsymbol{R}_{i,l}$.

2. using ***V*** and $\bar{v}_l$, do *n* + 1 curve interpolations through $\boldsymbol{R}_{i,0}, \ldots, \boldsymbol{R}_{i,m}$ (for $i = 0, \ldots, n$); this yields $\boldsymbol{P}_{i,j}$

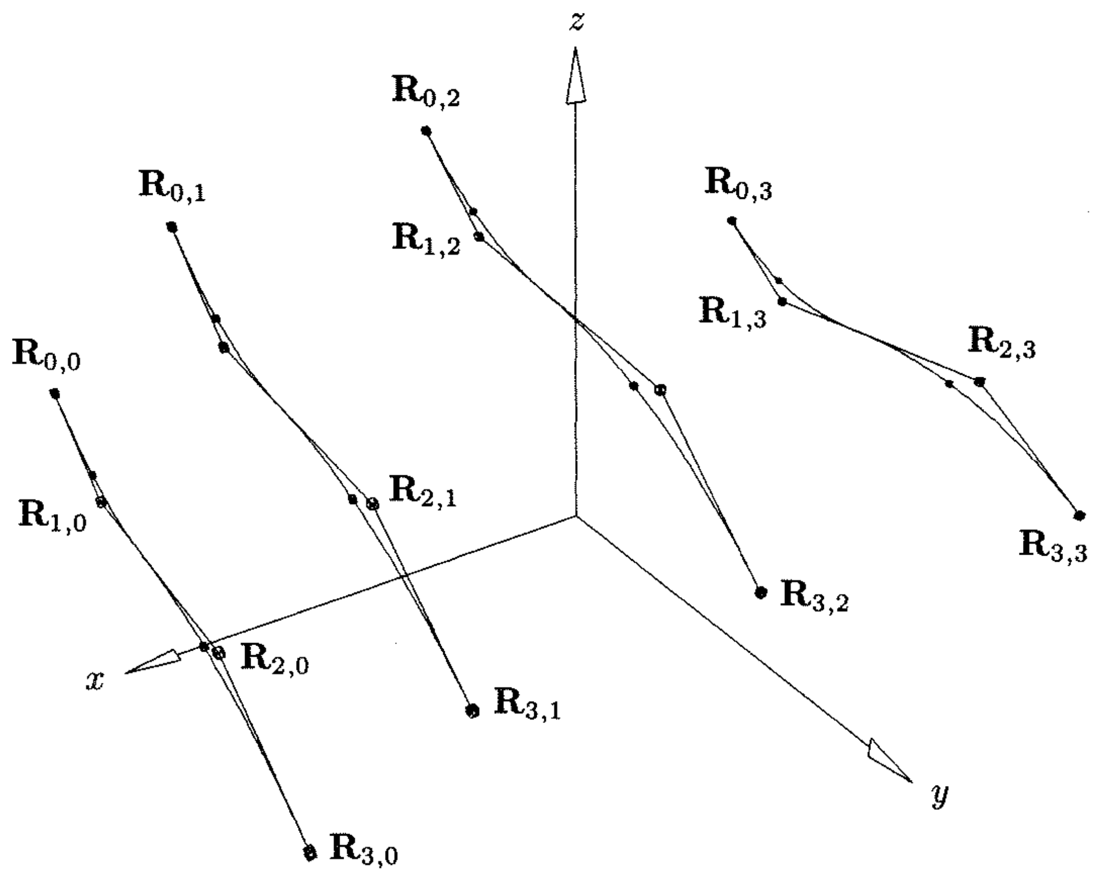
See algorithm A9.4

Data points:

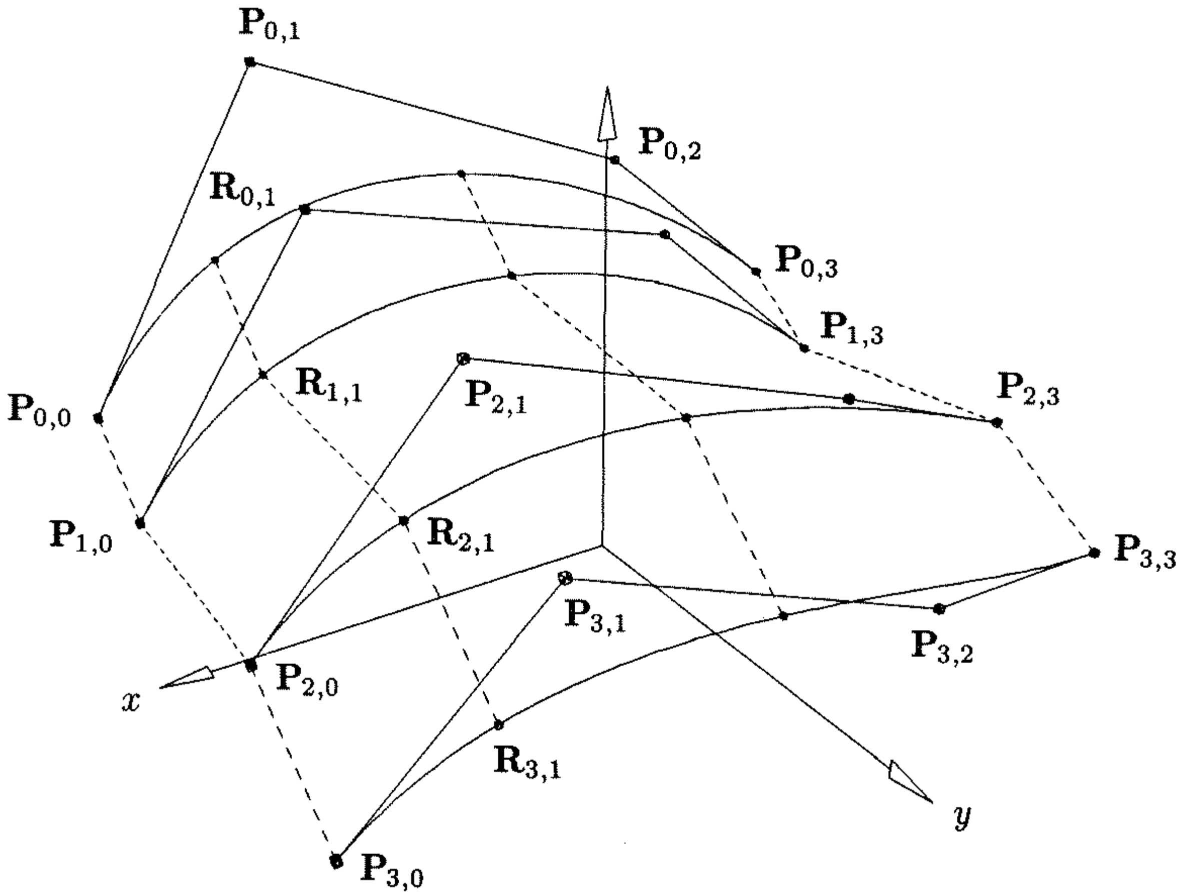


z
$\mathbf{Q}_{0,2}$
$\mathbf{Q}_{0,1}$
$\mathbf{Q}_{0,3}$
$\mathbf{Q}_{1,3}$
$\mathbf{Q}_{1,1}$
$\mathbf{Q}_{0,0}$
$\mathbf{Q}_{2,3}$
$\mathbf{Q}_{2,2}$
$\mathbf{Q}_{1,0}$
$\mathbf{Q}_{3,3}$
$\mathbf{Q}_{3,2}$
$\mathbf{Q}_{2,0}$
x
$\mathbf{Q}_{3,1}$
y
$\mathbf{Q}_{3,0}$

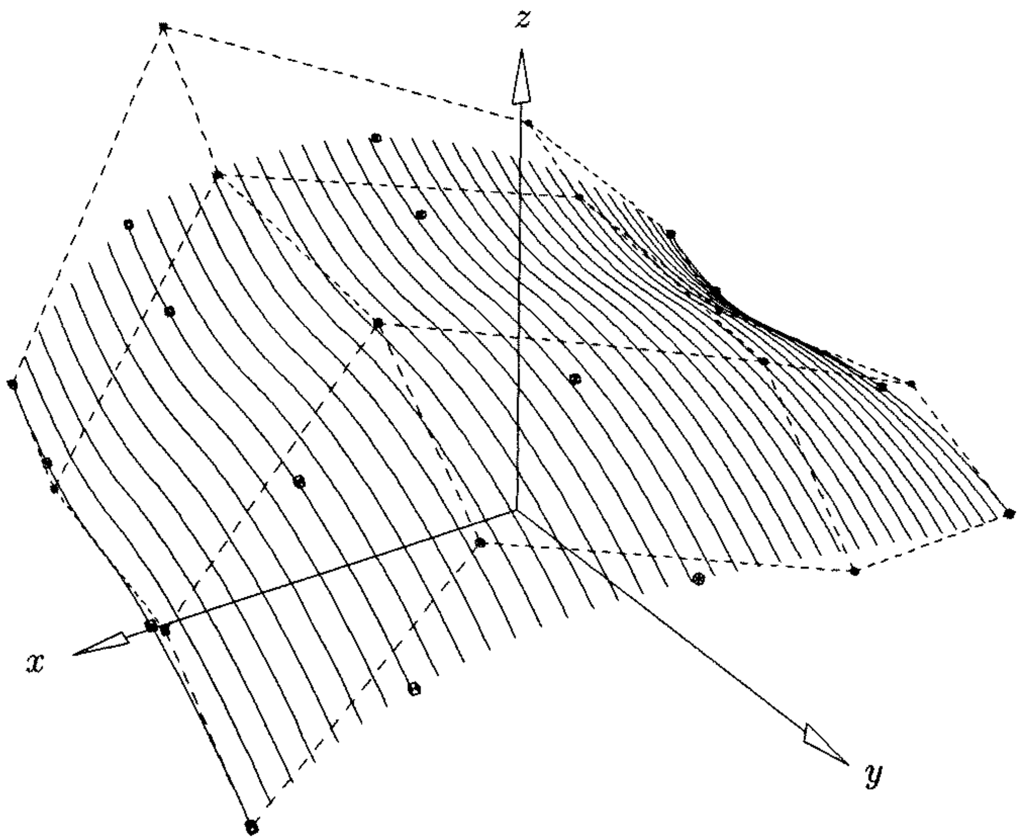
Interpolating the *u*-directional data points:



Interpolating the *v*-direction through control points of *u*-directional interpolants:



Surface interpolant showing control points:



The algorithm is symmetric; the same surface is obtained by:

1. doing n + 1 curve interpolations through the $\boldsymbol{Q}_{0,l}, \ldots, \boldsymbol{Q}_{n,l}$ to obtain the $\boldsymbol{R}_{k,j}$ (control points of isoparametric curves $\boldsymbol{S}(\bar{u}_k, v)$);

2. then doing m + 1 curve interpolations through the $\boldsymbol{R}_{0,j}, \ldots, \boldsymbol{R}_{n,j}$ to obtain the $\boldsymbol{P}_{i,j}$.

Derivative constraints can also be included in global surface interpolation. The derivative formulas can be used to add one additional equation for each derivative constraint.

However, if the number of constraints is not the same for each row or column it becomes difficult to solve for the unknown surface control points.

With clever use of curve knot insertion and surface knot removal, partial derivative constraints at individual data points can be handled.

Local surface interpolation methods (yet to come) are well suited to handling derivative constraints.