CONTROLLING THE INTERPOLATION OF NURBS CURVES AND SURFACES

by

PETER STEPHEN LOCKYER

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

Geometric Modelling Group
Mechanical and Manufacturing Engineering
School of Engineering
The University of Birmingham
Edgbaston
Birmingham B15 2TT

October 2006

Synopsis

The primary focus of this thesis is to determine the best methods for controlling the interpolation of NURBS curves and surfaces. The various factors that affect the quality of the interpolant are described, and existing methods for controlling them are reviewed. Improved methods are presented for calculating the parameter values, derivative magnitudes, data point spacing and twist vectors, with the aim of producing high quality interpolants with minimal data requirements.

A new technique for obtaining the parameter values and derivative magnitudes is evaluated, which constructs a C^1 cubic spline with orthogonal first and second derivatives at specified parametric locations. When this data is used to create a C^2 spline, the resulting interpolant is superior to those constructed using existing parameterisation and derivative magnitude estimation methods.

Consideration is given to the spacing of data points, which has a significant impact on the quality of the interpolant. Existing methods are shown to produce poor results with curves that are not circles. Three new methods are proposed that significantly reduce the positional error between the interpolant and original geometry.

For constrained surface interpolation, twist vectors must be estimated. A method is proposed that builds on the Adini method, and is shown to have improved error characteristics. In numerical tests, the new method consistently outperforms Adini.

Interpolated surfaces are often required to join together smoothly along their boundaries. The constraints for joining surfaces with parametric and geometric continuity are discussed, and the problem of joining N patches to form an N-sided region is considered. It is shown that regions with odd N can be joined with G^1 continuity, but those with even N or requiring G^2 continuity can only be obtained for specific geometries.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr R. J. Cripps, who first motivated my interest in CAE, and whose guidance, feedback and support has been invaluable. I would also like to thank Professor A. A. Ball, whose expertise in research and in the field of CAGD has been of tremendous benefit. I am pleased to acknowledge the financial support of EPSRC (grant reference GR/R35735/01), and The University of Birmingham. Finally, my deepest gratitude goes to my wife, Bex, for her constant love, support and encouragement.

Nomenclature

Symbol	Chapter	Description
α_k	7	Constant for G^2 construction
β_k	7	Constant for G^2 construction
χ_k	3	Angle between tangents $\boldsymbol{t}_k$ and $\boldsymbol{t}_{k+1}$
δ	2	Small angle tolerance
$\boldsymbol{\Delta}$	6	Vector movement of $\boldsymbol{P}_{1,1}^{0,0}$
Δ_k	3	Parametric interval of local segment, $\Delta_k = \left(s_{k+1} - s_k\right)$
ε	7	Maximum normal curvature error for arbitrary topologies
ϕ_k	3	Angle between tangent $\boldsymbol{t}_{k+1}$ and chord $\boldsymbol{s}_k$
φ_k	3	Scalar to ensure orthogonality with circle orthogonal parameterisation
γ_k	7	Constant for G^2 construction
η_k	7	Constant for G^2 construction
κ_0, κ_1	4	Start and end curvature values for GCS curve, $\boldsymbol{G}(\xi)$
κ_{min}, κ_{max}	4	Minimum and maximum normal curvature values
$\kappa(u)$	4	Normal curvature of parametric curve
$\kappa(\xi)$	4	Normal curvature of GCS curve
λ	6	Constant used to specify the relationship between twists
μ	4	Local geometric parameter
θ_k	3	Angle between tangent $\boldsymbol{t}_k$ and chord $\boldsymbol{s}_k$
ρ	4	Small curvature tolerance
σ_k	7	Constant for G^2 construction
ς	4	Parametric tolerance for geometric parameterisation

τ_k	1-3	Derivative magnitude for NURBS curve, $\tau_k = \left\Vert \boldsymbol{C}_u\left(s_k\right)\right\Vert$
υ_k	7	Constant for G^2 construction
ω	5	Total winding angle of curve
ω_k^*	5	Winding angle parameter from start of curve to $\boldsymbol{Q}_k^*$, where $*$ is a placeholder for the method used to space the data points
ξ	4-5	Arc length parameter of curve
ξ_k^*	5	Arc length parameter from start of curve to $\boldsymbol{Q}_k^*$, where $*$ is a placeholder for the method used to space the data points
ψ_k	3	Scalar to ensure orthogonality with circle orthogonal parameterisation
a_k	2-3	Start derivative magnitude for the local segment
b_k	2-3	End derivative magnitude for the local segment
$\boldsymbol{B}^k\left(u\right)$	6	The *k*th cubic Ball curve segment, with parameter *u*
$\boldsymbol{B}_{[*]}^{k,l}\left(u,v\right)$	6	The $\left(k,l\right)$th cubic Ball sub-patch, taken from the surface $\boldsymbol{S}\left(u,v\right)$, where $*$ indicates the twist estimation method used to generate the surface
c_k	2-3	Magnitude of chord length, $c_k = \left\Vert \boldsymbol{Q}_{k+1} - \boldsymbol{Q}_k\right\Vert$
C^0, C^1, C^2	Multiple	Levels of parametric continuity
$\boldsymbol{C}\left(u\right)$	Multiple	NURBS curve, with parameter *u*
$\boldsymbol{C}_u\left(u\right)$, $\boldsymbol{C}_{uu}\left(u\right)$	Multiple	First and second derivative of $\boldsymbol{C}\left(u\right)$, respectively
$\boldsymbol{C}^{k,l}\left(u,v\right)$	6	Bilinearly-blended Coons patch, having the same boundaries as $\boldsymbol{B}_{[T]}^{k,l}\left(u,v\right)$.
$\left[\boldsymbol{C}\right]$	2	Matrix holding a single *x*, *y* or *z* component for each data point used to interpolate a curve; also contains derivative info if constrained
d_k	5	Projected distance between chord $\boldsymbol{c}_k$ and tangent $\boldsymbol{t}_{k+1}$ at $\boldsymbol{Q}_k$

$\boldsymbol{d}_i$	3	Chord, $\boldsymbol{d}_i = \boldsymbol{Q}_k - \boldsymbol{Q}_{k-1}$
e_k	5	Projected distance between chord $\boldsymbol{c}_k$ and tangent $\boldsymbol{t}_k$ at $\boldsymbol{Q}_{k+1}$
$\boldsymbol{e}_i$	3	Chord, $\boldsymbol{d}_i = \boldsymbol{Q}_{k+1} - \boldsymbol{Q}_{k-1}$
$\boldsymbol{E}^{k,l}_{[*]}$	6	The vector difference between $\boldsymbol{S}^{k,l}_{[*]}(\tfrac{1}{2},\tfrac{1}{2})$ and $\boldsymbol{S}^{k,l}_{[T]}(\tfrac{1}{2},\tfrac{1}{2})$ where $*$ indicates the twist estimation method used to generate the surface
f_s	6	Constant that relates the change between adjacent internal control points for a cubic Ball segment
G^0, G^1, G^2	7	Levels of geometric continuity
$\boldsymbol{G}(\xi)$	4	GCS curve, with arc length parameter ξ
l_k	2	Length of circular arc between $\boldsymbol{Q}_k$ and $\boldsymbol{Q}_{k+1}$
m	Multiple	Number of control points, minus one, in u direction for NURBS curve (or surface)
$[\boldsymbol{M}]$	2	Array of $N_{i,p}(u)$ basis functions for all parameter values when interpolating a NURBS curve (or surface)
n	Multiple	Number of control points, minus one, in v direction for NURBS surface
N	5	Number of points used when interpolating a curve; $m+1$ when unconstrained, $m-1$ when constrained
	7	Number of sides to an arbitrary topology region
$N_{i,p}(u)$	1-2	ith basis function of degree p
$N'_{i,p}(u)$, $N''_{i,p}(u)$	1-2	First and second derivatives of $N_{i,p}(u)$
$[\boldsymbol{N}]$	2	Array of $N_{j,q}(v)$ basis functions for all parameter values when interpolating a NURBS surface
p	Multiple	Degree of NURBS curve (or surface) in u
$\boldsymbol{P}_i$	Multiple	ith Control point for NURBS curve
$\boldsymbol{P}_{i,j}$	Multiple	(i,j)th Control point for NURBS surface
$\boldsymbol{P}_{i,j[C]}$	6	(i,j)th Control point for a bilinearly-blended Coons surface.

$\boldsymbol{P}_i^k$	6	*i*th control point for *k*th cubic Ball segment $\boldsymbol{B}^k\left(u\right)$
$\boldsymbol{P}_{i,j}^k$	7	$\left(i,j\right)$ th control point of the *k*th Bezier patch to join in a closed loop creating an *N*-sided region.
$\boldsymbol{P}_{i,j}^{k,l}$	6	$\left(i,j\right)$ th control point for cubic Ball patch $\boldsymbol{B}_{[*]}^{k,l}\left(u,v\right)$
$\left[\boldsymbol{P}\right]$	2	Matrix holding a single *x*, *y* or *z* component of each control point following NURBS curve (or surface) interpolation
q	Multiple	Degree of NURBS surface in *v*
$\boldsymbol{Q}_k$	Multiple	Data point, $\boldsymbol{Q}_k = \boldsymbol{C}\left(s_k\right)$, used to interpolate NURBS curve
$\boldsymbol{Q}_k^*$	5	Data point, where $*$ is a placeholder denoting the method used to space the points
$\boldsymbol{Q}_{k,l}$	Multiple	Data point, $\boldsymbol{Q}_{k,l} = \boldsymbol{S}\left(s_k,t_l\right)$, used to interpolate a NURBS surface
r	5	Shape factor of GCS, $r > -1$
r_k	2	Radius used for *k*th span – circular arc parameterisation
$\boldsymbol{R}^c\left(u,v\right)$, $\boldsymbol{R}^d\left(u,v\right)$	2	Ruled surface in *u* and *v* respectively
$\boldsymbol{R}_u^c\left(u,v\right)$, $\boldsymbol{R}_v^d\left(u,v\right)$	2	Differentiated ruled surface in *u* and *v* respectively
$\boldsymbol{R}^{cd}\left(u,v\right)$	2	Bilinear surface
s_k	Multiple	*k*th parameter value in *u* corresponding to $\boldsymbol{Q}_k$ or $\boldsymbol{Q}_{k,l}$
$\boldsymbol{s}_k$	3	Unit direction of chord
S	4	Total arc length of GCS curve
$\left[\boldsymbol{S}\right]$	2	Matrix holding a single *x*, *y* or *z* component for each data point used to interpolate a surface; also contains derivative and twist data if constrained
$\boldsymbol{S}\left(u,v\right)$	Multiple	NURBS surface, with parameters *u* and *v*
$\boldsymbol{S}^k\left(u,v\right)$	7	*k*th surface to join in a loop to create an *N*-sided region.

$\boldsymbol{S}_u(u,v)$, $\boldsymbol{S}_v(u,v)$	Multiple	Derivatives of $\boldsymbol{S}(u,v)$ in *u* and *v* respectively
$\boldsymbol{S}_{uv}(u,v)$	Multiple	Mixed partial derivative, or twist vector, of $\boldsymbol{S}(u,v)$
$\boldsymbol{S}_{uv[*]}^{k,l}(u,v)$	6	Estimated twist vector for NURBS surface, where $*$ indicates the method used to generate the estimate. Superscript denotes the (k,l)th sub-patch if used
t	3	Local parameter for curve segment, $t=(1-u)/\Delta_k=(1-u)/(s_{k+1}-s_k)$
$\boldsymbol{t}_k$	Multiple	Tangent direction at $\boldsymbol{Q}_k$, $\boldsymbol{t}_k=\boldsymbol{C}_u(s_k)/c_k$
t_l	Multiple	*l*th parameter value in *v* corresponding to $\boldsymbol{Q}_{k,l}$
T	4	Number of values used for numerical representation of geometric parameterisation
u	Multiple	Parameter value for NURBS curve (or surface)
u_i	Multiple	*i*th knot in $\boldsymbol{U}$
$\boldsymbol{U}$	Multiple	Knot vector in the *u* direction of a NURBS curve (or surface)
v	Multiple	Parameter value for NURBS or surface
v_i	Multiple	*i*th knot in $\boldsymbol{V}$
$\boldsymbol{V}$	Multiple	Knot vector in the *v* direction of a NURBS surface
w_i, $w_{i,j}$	1	Rational weights for a NURBS curve or surface respectively
W	3-4	Index indicating where the orthogonality conditions are satisfied for each C^1 curve segment

Contents

Chapter 1

Introduction

1.1 Introduction to NURBS interpolation

Interpolation, in the context of Computer Aided Geometric Design (CAGD), is the process of constructing a freeform parametrically defined curve, or surface, that passes through a set of data points exactly. Additional constraints are often imposed, which leads to constrained interpolation, e.g. for a curve, derivative vectors can be prescribed at specified points, usually to control the tangent direction at the start and end. Figure 1.1 illustrates a curve, $\boldsymbol{C}(u)$, that interpolates five data points, $\boldsymbol{C}(s_k)$, $0 \leq k \leq 4$, and two tangent directions, $\boldsymbol{t}_0$ and $\boldsymbol{t}_4$.

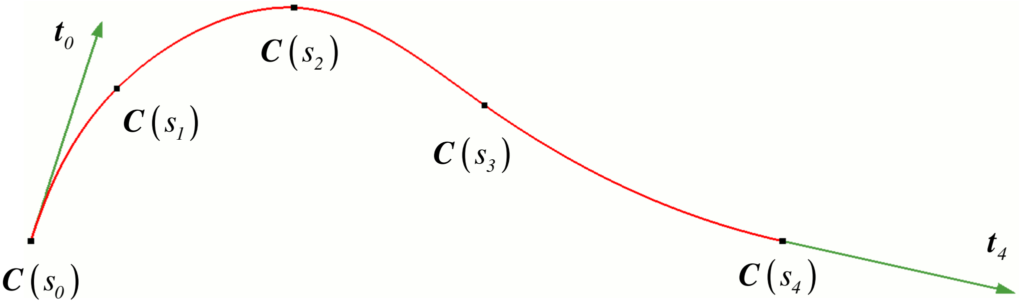


Figure 1.1 – A curve interpolated with five points and two tangent directions

Interpolation is distinguished from approximation, which is another common process for fitting curves or surfaces to geometric data; the resulting entity does not necessarily interpolate the data, but strives to produce a 'best fit', e.g. least-squares approximation. Figure 1.2 illustrates a curve approximating eleven data points, $\boldsymbol{Q}_k$, $0 \leq k \leq 10$, with each point having a corresponding error E_k. Approximation is often used when the data points contain 'noise', e.g. from sampling points using a contact probe, whereas data used for interpolation is assumed to be exact.

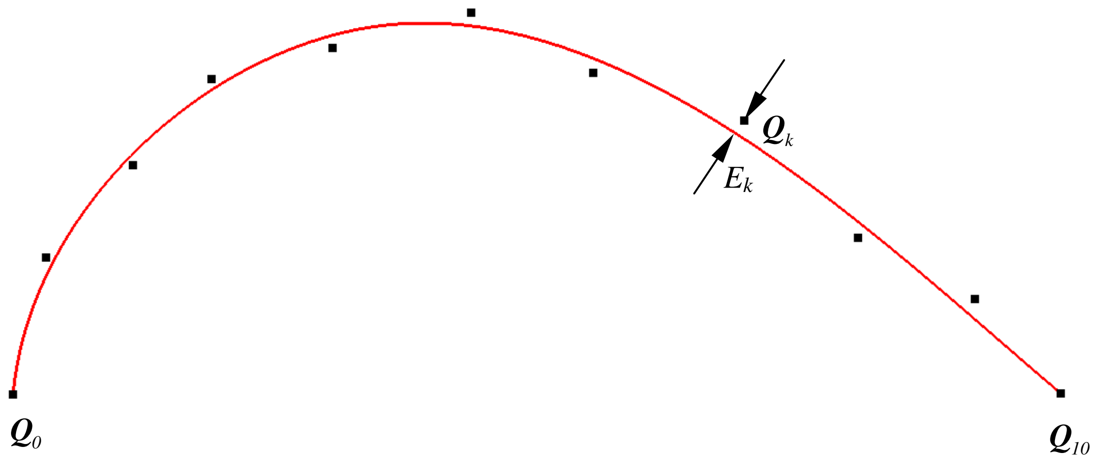


Figure 1.2 – A curve approximating eleven data points

A parametrically defined curve or surface resulting from an interpolation process often has multiple spans or sub-patches respectively. Such entities can be conveniently represented using a single Non-Uniform Rational B-Spline (NURBS) curve or surface.

There are a significant number of factors that affect the shape of the resulting NURBS entity. This thesis is primarily concerned with assessing the effect these factors have, reviewing published methods for controlling them, and developing new ways to improve this control. Section 1.2 provides an overview of NURBS entities, before Section 1.3 outlines these factors in more detail. Section 1.4 motivates the need for improved control over the interpolation process, and Section 1.5 provides an outline of this thesis.

1.2 Overview of NURBS

NURBS curves and surfaces are widely used in many Computer Aided Design (CAD) applications for representing the form of cars, aircraft, ships, shoes and numerous other items with sculptured features [Rogers, 2001]. They are parametrically defined polynomial entities, thus well suited for software implementation, whilst their rational nature provides additional degrees of freedom compared to their non-rational counterparts. This enables them to exactly reproduce conic surfaces [Rogers, 2001]; however, the additional versatility incurs the penalty of increased complexity and, frequently, a reduced understanding for the end user.

The development of NURBS entities can be traced back to Schoenberg [1946], who first introduced the B-Spline basis for statistical data smoothing. Cox [1972] and de Boor [1972] independently discovered a recursive definition, allowing the basis functions to be evaluated more efficiently, and provided greater numerical stability. Gordon and Riesenfeld [1974] introduced the B-spline basis to the domain of CAGD in the context of parametrically defined curves, and Versprille [1975] first discussed the idea of rational B-splines.

NURBS have remained a very popular representation for curves and surfaces in CAD software. Many factors may have contributed to their success including their versatility, ability to represent the Bézier form, and adoption into the IGES specification, thereby allowing CAD models to be transferred between different systems easily. Other spline forms have been published subsequently, but none have surpassed NURBS in terms of their universal support, despite any advantages they may offer. Barsky [1981] introduced the concept of the Beta-spline, which contains the B-spline as a subset. In general, adjacent spans within a Beta-spline do not meet with parametric continuity, rather with geometric continuity, and allow shape to be manipulated using bias and tension parameters [Barsky and Beatty, 1983]. Other spline forms build on the concept of maintaining geometric continuity, using the additional degrees of freedom as tension parameters; these include the Nu-spline [Nielson, G., 1984], the Tau-spline [Hagen, 1985], and the LT-spline [Cohen, 1987]. However, interpolation of these splines often result in a non-linear system of equations, and the majority of CAD systems do not support them.

1.2.1 B-spline basis functions

The *p*th degree B-spline basis functions, $N_{i,p}(u)$, $0 \le i \le m$, can be obtained recursively [Cox, 1972, and de Boor, 1972] in terms of a knot vector ***U***, which takes the form:

$$\boldsymbol{U} = \{u_0, \ldots, u_{m+p+1}\} \tag{1.1}$$

where the knots, u_i, are a sequence of non-decreasing numbers, i.e. $u_i \le u_{i+1}$, $0 \le i \le m+p$. It is noted that this condition allows multiple (i.e. repeated) knots, which have special significance – see Section 1.3.3. The *i*th basis function, $N_{i,p}(u)$, is given by:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \le u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \qquad 0 \le i \le m \tag{1.2}$$

noting that the quotient $\frac{0}{0}$ is defined as zero.

The *i*th basis function can be differentiated with respect to the parameter *u*. The first derivative, $N'_{i,p}(u)$, is found by differentiating (1.2) using the product rule, $(fg)' = f'g + fg'$:

$$N'_{i,0}(u) = 0$$

$$N'_{i,p}(u) = \frac{N_{i,p-1}(u) + (u - u_i)N'_{i,p-1}(u)}{u_{i+p} - u_i} + \frac{(u_{i+p+1} - u)N'_{i+1,p-1}(u) - N_{i+1,p-1}(u)}{u_{i+p+1} - u_{i+1}}$$

$$0 \le i \le m \qquad (1.3)$$

The second derivative, $N''_{i,p}(u)$, is found by differentiating (1.3) using the product rule:

$$N''_{i,0}(u) = N''_{i,1}(u) = 0$$

$$N''_{i,p}(u) = \frac{2N'_{i,p-1}(u) + (u - u_i)N''_{i,p-1}(u)}{u_{i+p} - u_i} + \frac{(u_{i+p+1} - u)N''_{i+1,p-1}(u) - 2N'_{i+1,p-1}(u)}{u_{i+p+1} - u_{i+1}}$$

$$0 \le i \le m \qquad (1.4)$$

Higher derivatives can be found analogously using the product rule. Piegl and Tiller [1997] prove, by induction, that (1.3) is equivalent to:

$$N'_{i,0}(u) = 0$$

$$N'_{i,p}(u) = \frac{p}{u_{i+p} - u_i}N_{i,p-1}(u) - \frac{p}{u_{i+p+1} - u_{i+1}}N_{i+1,p-1}(u) \qquad 0 \le i \le m \qquad (1.5)$$

Letting $N^{(k)}_{i,p}(u)$ denote the *k*th derivative of $N_{i,p}(u)$, Piegl and Tiller [1997] give the general formula:

$$N^{(k)}_{i,p}(u) = p\left(\frac{N^{(k-1)}_{i,p-1}(u)}{u_{i+p} - u_i} - \frac{N^{(k-1)}_{i+1,p-1}(u)}{u_{i+p+1} - u_{i+1}}\right) \qquad 0 \le i \le m \qquad (1.6)$$

1.2.2 B-spline curves

The pth degree Non-Uniform B-Spline (NUBS), i.e. non rational, curve, $\boldsymbol{C}(u)$, can be defined in terms of (1.2), and has $m+1$ control points, $\boldsymbol{P}_i$:

$$\boldsymbol{C}(u)=\sum_{i=0}^{m} N_{i,p}(u)\boldsymbol{P}_i \ , \qquad u_p \le u \le u_{m+1} \tag{1.7}$$

The curve consists of $(m+1-p)$ spans, which are C^{p-1} continuous [Piegl and Tiller, 1997] assuming there are no repeated interior knots. Figure 1.3 is an example of a cubic B-spline curve with three spans; it has an open clamped knot vector (i.e. $p+1$ repeated knots at either end), which causes the curve to start and end at $\boldsymbol{P}_0$ and $\boldsymbol{P}_m$ respectively.

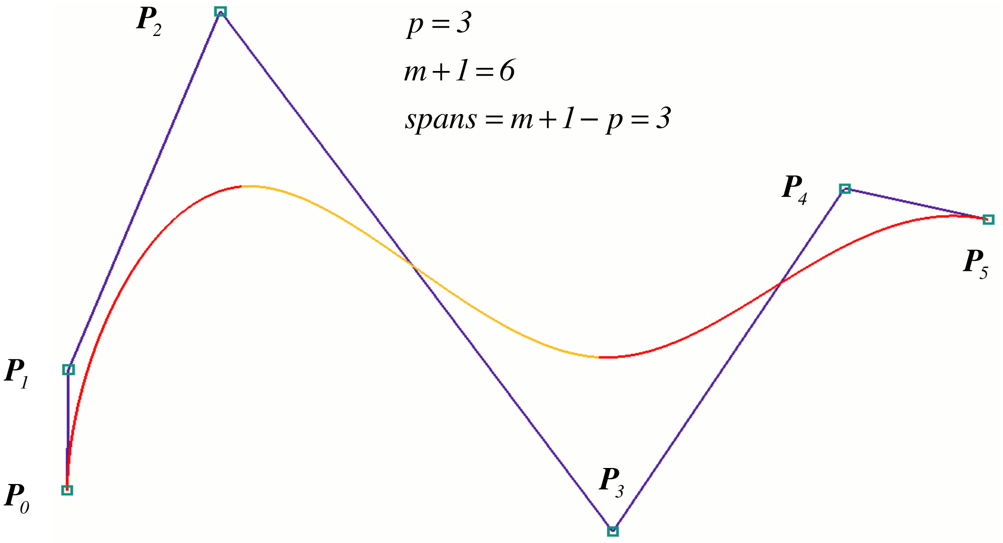


Figure 1.3 – B-spline curve, with 3 spans, and control polygon

The curve can be differentiated with respect to the parameter u. The first and second derivatives, $\boldsymbol{C}_u(u)$, and $\boldsymbol{C}_{uu}(u)$, are given by:

$$\boldsymbol{C}_u(u)=\sum_{i=0}^{m} N'_{i,p}(u)\boldsymbol{P}_i \ , \qquad u_p \le u \le u_{m+1} \tag{1.8}$$

$$\boldsymbol{C}_{uu}(u)=\sum_{i=0}^{m} N''_{i,p}(u)\boldsymbol{P}_i \ , \qquad u_p \le u \le u_{m+1} \tag{1.9}$$

The *k*th derivative of the curve, $\boldsymbol{C}_{(k)}(u)$, is given analogously:

$$\boldsymbol{C}_{(k)}(u) = \sum_{i=0}^{m} N_{i,p}^{(k)}(u)\boldsymbol{P}_i \ , \qquad u_p \le u \le u_{m+1} \tag{1.10}$$

The *p*th degree NURBS curve has $m+1$ weights, $w_i > 0$, $0 \le i \le m$, that can be used to bias the curve towards (or away from) their corresponding control points. The curve is defined as:

$$\boldsymbol{C}(u) = \frac{\sum_{i=0}^{m} N_{i,p}(u) w_i \boldsymbol{P}_i}{\sum_{i=0}^{m} N_{i,p}(u) w_i} \ , \qquad u_p \le u \le u_{m+1} \tag{1.11}$$

Derivatives of NURBS curves are not used in this thesis; the interested reader is referred to [Rogers, 2001] for definitions.

1.2.3 B-spline surfaces

A NUBS surface has a bi-directional control net, consisting of control points $\boldsymbol{P}_{i,j}$, $0 \le i \le m$, $0 \le j \le n$. The surface has an additional knot vector ***V*** in the form of (1.1):

$$\boldsymbol{V} = \{v_0, \ldots, v_{n+q+1}\} \tag{1.12}$$

where the degree of the surface is *p* in the *u*-direction and *q* in the *v*-direction. The surface requires two basis functions in the form of (1.2), and is given by:

$$\boldsymbol{S}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,p}(u) N_{j,q}(v) \boldsymbol{P}_{i,j}, \qquad u_p \le u \le u_{m+1},\ v_q \le v \le v_{n+1} \tag{1.13}$$

The surface consists of $(m+1-p)\times(n+1-q)$ sub-patches, which are C$^{p\text{-}1}$ and C$^{q\text{-}1}$ continuous in the *u*- and *v*-directions respectively, assuming there are no repeated interior knots. An example of a cubic B-spline surface is given in Figure 1.4; both knot vectors ***U*** and ***V*** are open clamped.

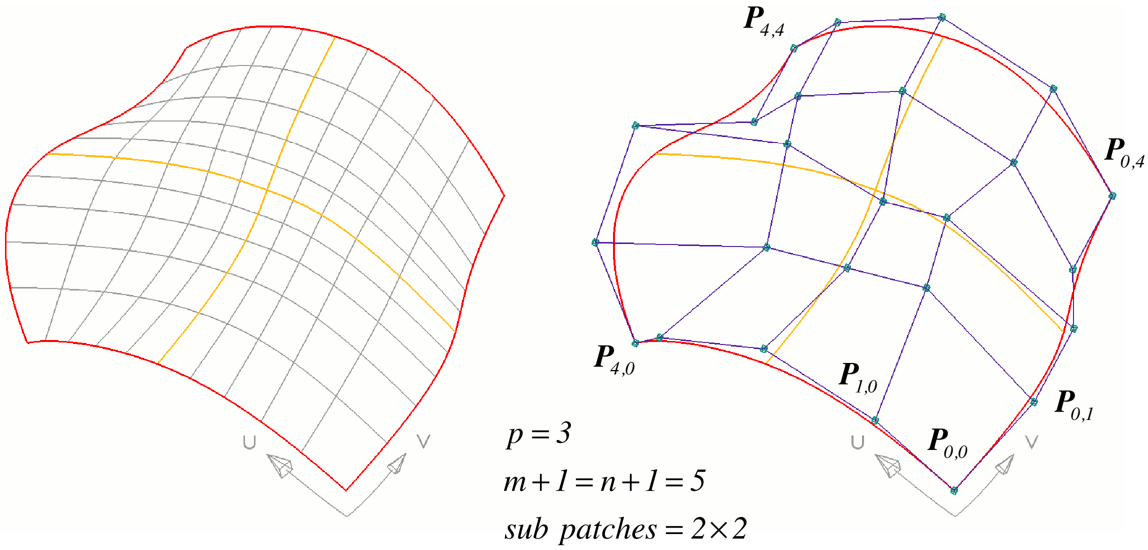


Figure 1.4 – B-spline surface with 2x2 sub-patches, and control polygon

The surface can be differentiated with respect to both *u* and *v*. The first derivatives, $\boldsymbol{S}_u(u,v)$, and $\boldsymbol{S}_v(u,v)$, are given by:

$$\boldsymbol{S}_u(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n}N'_{i,p}(u)N_{j,q}(v)\boldsymbol{P}_{i,j}, \qquad u_p\le u\le u_{m+1},\ v_q\le v\le v_{n+1} \qquad (1.14)$$

$$\boldsymbol{S}_v(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n}N_{i,p}(u)N'_{j,q}(v)\boldsymbol{P}_{i,j}, \qquad u_p\le u\le u_{m+1},\ v_q\le v\le v_{n+1} \qquad (1.15)$$

The mixed partial derivatives, $\boldsymbol{S}_{uv}(u,v)$, are given by:

$$\boldsymbol{S}_{uv}(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n}N'_{i,p}(u)N'_{j,q}(v)\boldsymbol{P}_{i,j}, \qquad u_p\le u\le u_{m+1},\ v_q\le v\le v_{n+1} \qquad (1.16)$$

The second derivatives, $\boldsymbol{S}_{uu}(u,v)$, and $\boldsymbol{S}_{vv}(u,v)$, are given by:

$$\boldsymbol{S}_{uu}(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n}N''_{i,p}(u)N_{j,q}(v)\boldsymbol{P}_{i,j}, \qquad u_p \le u \le u_{m+1},\ v_q \le v \le v_{n+1} \tag{1.17}$$

$$\boldsymbol{S}_{vv}(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n}N_{i,p}(u)N''_{j,q}(v)\boldsymbol{P}_{i,j}, \qquad u_p \le u \le u_{m+1},\ v_q \le v \le v_{n+1} \tag{1.18}$$

In general, the (k,l)th derivative, $\boldsymbol{S}_{(k)(l)}(u,v)$, is given by:

$$\boldsymbol{S}_{(k)(l)}(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n}N^{(k)}_{i,p}(u)N^{(l)}_{j,q}(v)\boldsymbol{P}_{i,j}, \quad u_p \le u \le u_{m+1},\ v_q \le v \le v_{n+1} \tag{1.19}$$

A NURBS surface requires an additional $(m+1)\times(n+1)$ weights, $w_{i,j}>0$, $0\le i\le m$, $0\le j\le n$, and is given by:

$$\boldsymbol{S}(u,v)=\frac{\sum_{i=0}^{m}\sum_{j=0}^{n}N_{i,p}(u)N_{j,q}(v)\boldsymbol{P}_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n}N_{i,p}(u)N_{j,q}(v)w_{i,j}}, \qquad u_p \le u \le u_{m+1},\ v_q \le v \le v_{n+1} \tag{1.20}$$

Derivatives of NURBS surfaces are not used in this thesis; the interested reader is referred to [Rogers, 2001] for definitions.

1.3 Factors affecting interpolation

A variety of factors affect the interpolation process, including:

- Type of interpolation
- Parameterisation
- Knot vector generation
- Derivative constraints
- Derivative magnitude estimation
- Degree
- Rational weightings
- Spacing and number of data points
- Reconciliation of parameter values (surfaces only)
- Twist vector estimation (surfaces only)

An overview of each factor is given in this section, highlighting their influences on the interpolated curve or surface. Methods for controlling many of these factors are detailed in subsequent chapters, and are therefore omitted in this overview.

1.3.1 Type of interpolation

Interpolation can be performed globally or locally. Global interpolation produces a linear system of equations in terms of all the data points and derivatives – resulting directly in a set of control points. Local interpolation algorithms tend to be more geometric in nature, constructing the curve or surface segment-wise, using only local data for each step. Local interpolation processes frequently do not guarantee any form of continuity between segments, and can result in multiple internal knots within the knot vector [Piegl and Tiller, 1997]. Consequently, this thesis deals only with global interpolation methods, as they do not require further manipulation to satisfy continuity constraints.

1.3.2 Parameterisation

Interpolation produces a parametrically defined curve (or surface) that passes through the defining data points. The data points on a curve, $\boldsymbol{C}(s_k)$, therefore have corresponding parameter values, s_k, $0 \le k \le m$. Parameterisation is the process that assigns a particular parameter value to each point. Generally, the curve is parameterised over [0,1], therefore $s_0 = 0$ and $s_m = 1$. Consider a planar quadratic curve interpolating three data points, as illustrated in Figure 1.5. The three different curves are obtained by simply varying the parameter value s_1. It is evident that parameterisation has a very significant effect on the interpolated curve.

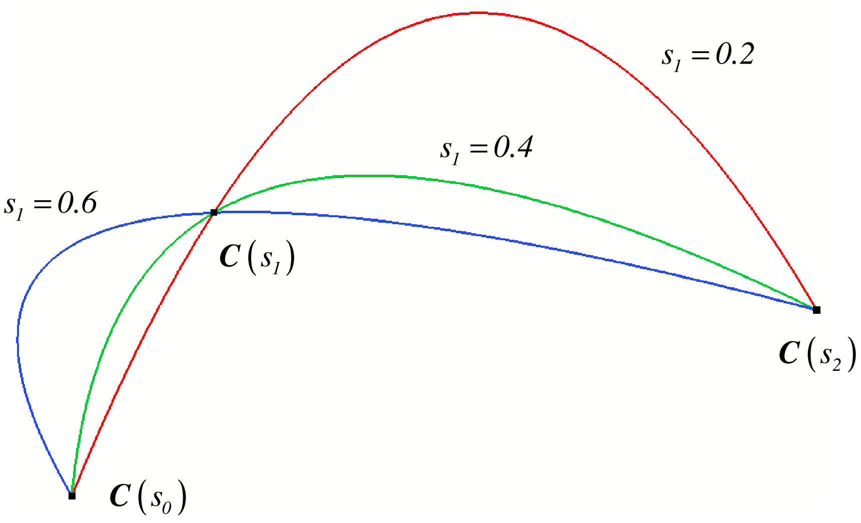
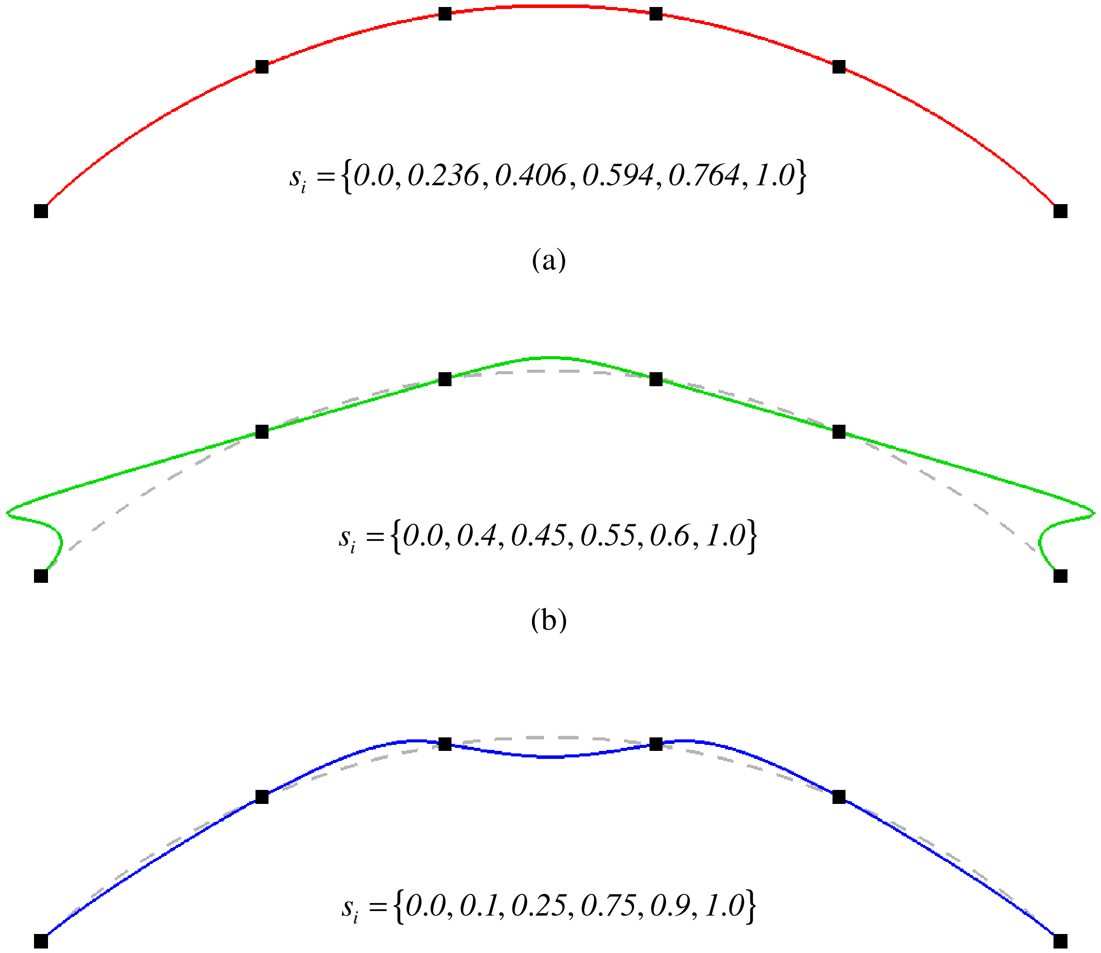


Figure 1.5 – Effect of parameterisation on a quadratic curve

It is noted that all three curves in Figure 1.5 are mathematically correct, and satisfy the data constraints. However, it could be argued that only the curve corresponding to $s_1 = 0.4$ 'characterises' the data points.

The purpose of any parameterisation algorithm is to automatically assign parameter values to points such that the resulting curve is characteristic of the data (see Figure 1.6a), i.e. does not exaggerate geometric features (Figure 1.6b), or cause the curve to undulate through smooth data points (Figure 1.6c).


$s_i = \{0.0, 0.236, 0.406, 0.594, 0.764, 1.0\}$
$s_i = \{0.0, 0.4, 0.45, 0.55, 0.6, 1.0\}$
$s_i = \{0.0, 0.1, 0.25, 0.75, 0.9, 1.0\}$

(a)

(b)

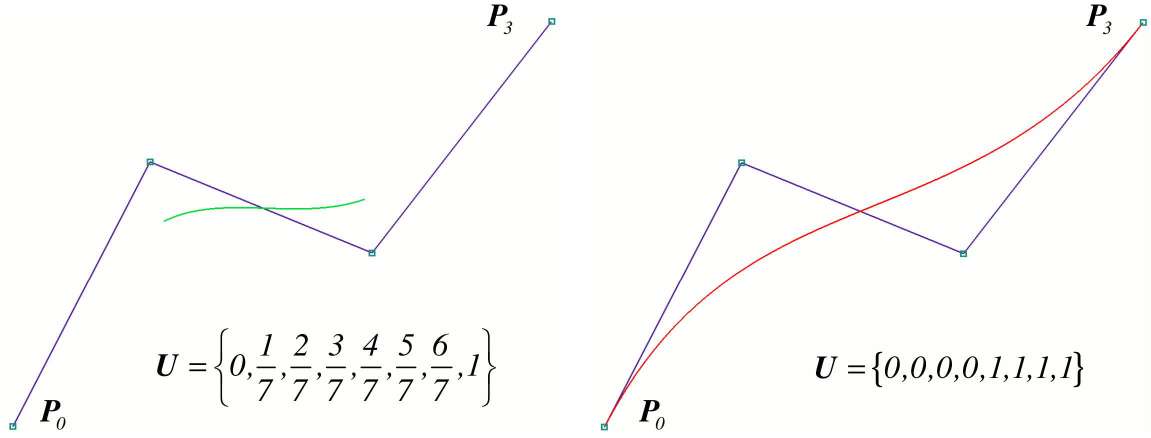
(c)

Figure 1.6 – Effect of poor parameterisation

1.3.3 Knot vector generation

The spacing of knots in the knot vector has a significant effect on the shape of a B-spline curve. The knot vector always has $m+p+2$ monotonically increasing knots, i.e. for a curve, $u_i \leq u_{i+1}$, $0 \leq i \leq m+p$. It is the relative spacing between knots that affects the B-spline basis, therefore it is usual for the knots to lie in the range $u_i \in [0,1]$, $0 \le i \le m+p+1$. The simplest way to distribute the knots is to evenly space them. However, the result is that the curve does not start and end at $\boldsymbol{P}_0$ and $\boldsymbol{P}_m$ respectively, Figure 1.7. The solution is to use an open clamped knot vector, which uses $p+1$ repeated knots at the ends, i.e.:

$$\boldsymbol{U} = \{u_0 = 0, \ldots, u_p = 0, u_{p+1}, \ldots, u_m, u_{m+1} = 1, \ldots, u_{m+p+1} = 1\} \tag{1.21}$$



$\boldsymbol{P}_3$
$\boldsymbol{P}_0$
$\boldsymbol{U} = \left\{0, \frac{1}{7}, \frac{2}{7}, \frac{3}{7}, \frac{4}{7}, \frac{5}{7}, \frac{6}{7}, 1\right\}$
$\boldsymbol{P}_3$
$\boldsymbol{P}_0$
$\boldsymbol{U} = \{0,0,0,0,1,1,1,1\}$

Figure 1.7 – B-spline curves with evenly spaced knots, and open clamped knots

For the *i*th span, the parameter value u varies in the range $u_{p+i} \le u \le u_{p+i+1}$, $0 \le i \le m-p$. Where the curve contains multiple spans, i.e. $m+1-p > 1$, the internal knots, u_i, $p+1 \le i \le m$, need to be distributed appropriately. Again, they could be evenly spaced, although Piegl and Tiller [1997] warn that this can lead to a singular system of equations. Intuitively, a single span, of degree *p*, can only satisfy a maximum of $p+1$ data constraints such as points or derivatives; if the distribution of knots causes a single span to be over-specified, then the interpolation cannot be successful. Most algorithms ensure that each span is not over-specified by distributing the knots in conjunction with the parameter values. Any repeated internal knots reduce the level of continuity between adjacent spans by one.

1.3.4 Derivative constraints

The interpolation process is constrained if additional information is supplied that the resulting entity must satisfy. This is usually the start and end derivatives for curves, and the cross boundary derivatives for surfaces. The purpose of supplying such data is to control the tangent directions at boundaries, allowing greater control of the interpolation process. The system of equations remains linear when constrained by derivative information; other constraints can be imposed, such as curvature conditions, but these result in non-linear systems of equations.

Figure 1.8 shows two curves interpolating three points sampled from a unit circle; one curve uses derivative constraints, the other does not. The derivatives shown are scaled down by approximately a factor of ten for clarity.

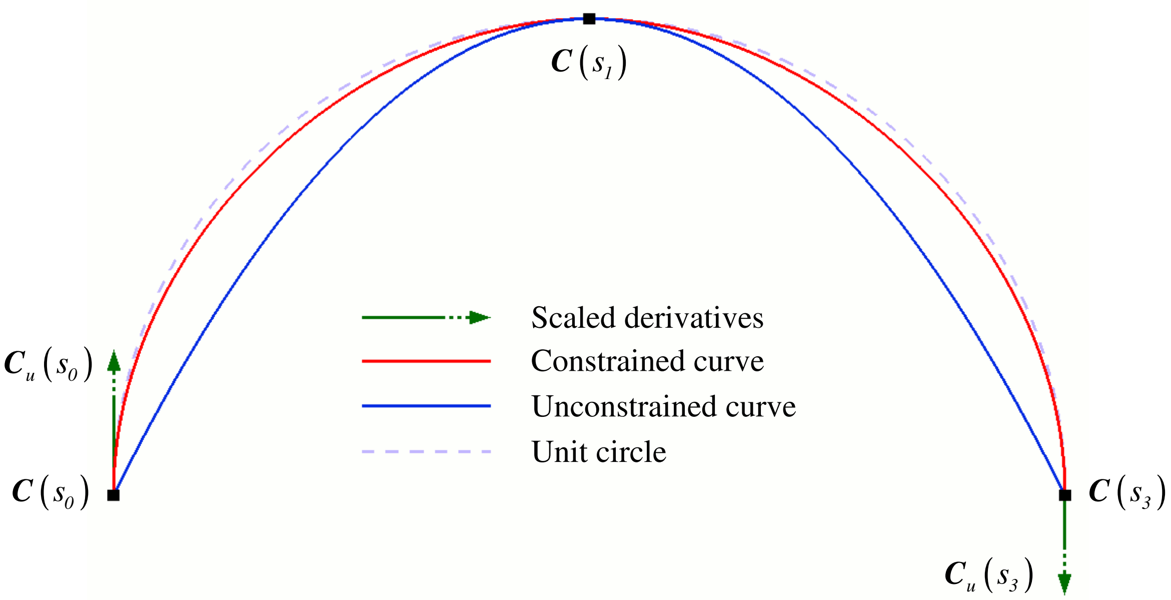


Figure 1.8 – Circle data interpolated with constrained and unconstrained curves

A NUBS curve contains $m+1$ control points, therefore, for unconstrained interpolation, the system of linear equations requires $m+1$ data points. Derivative constraints provide an additional two equations, therefore constrained interpolation requires just $m-1$ data points.

1.3.5 Derivative magnitude estimation

When a curve or surface is interpolated, the designer usually can only provide geometric data, i.e. points, $\boldsymbol{C}(s_i)$, $0 \le i \le m-2$, and end tangent directions, $\boldsymbol{t}_0$ and $\boldsymbol{t}_{m-2}$; a constrained interpolation, however, requires parametric derivatives, thus the derivative magnitudes, τ_0 and τ_{m-2}, need to be estimated, where:

$$\tau_0 \boldsymbol{t}_0 = \boldsymbol{C}_u(s_0)$$
$$\tau_{m-2} \boldsymbol{t}_{m-2} = \boldsymbol{C}_u(s_{m-2})$$

The derivative directions are a function of the geometry, but the magnitudes have no geometric interpretation because the derivatives are formulated with respect to the parameterisation.

To illustrate the effect that derivative magnitudes have, consider the interpolation of two data points and two tangent directions by a planar cubic curve. There are two degrees of freedom which control the end derivatives, τ_0 and τ_1. Figure 1.9 shows the effect of varying τ_0.

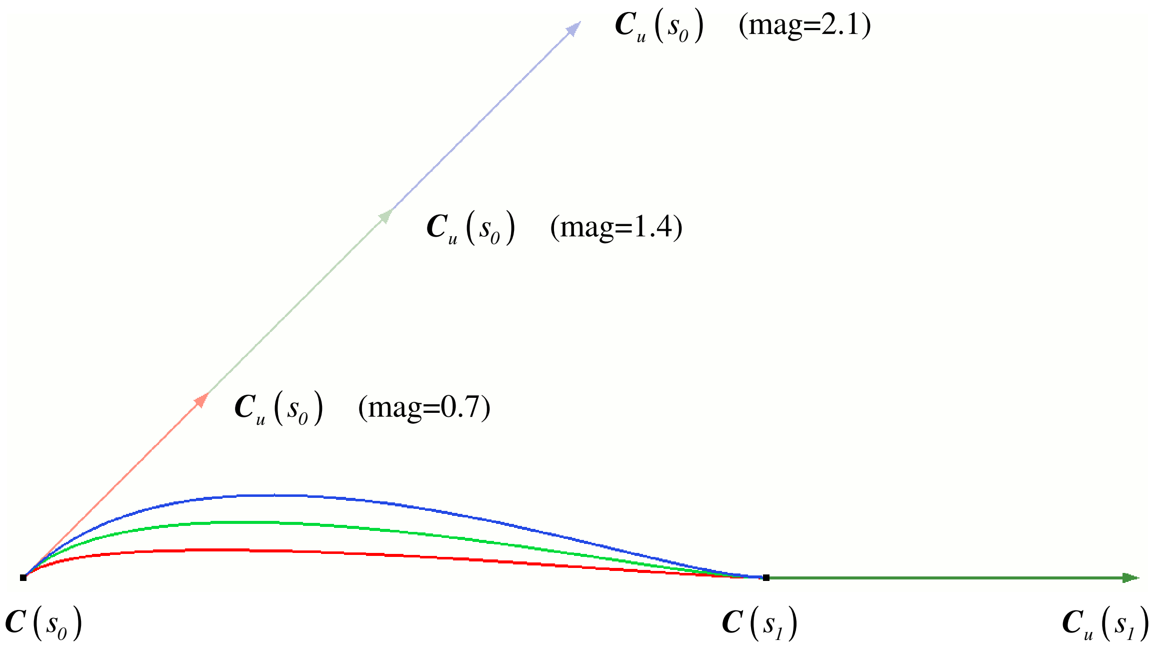


Figure 1.9 – Effect of derivative magnitude

Despite the magnitude being a function of the parameterisation, it is commonplace to approximate it using an estimate of the interpolated curve's length. Several distance-based methods build on this strategy, e.g. summing the chord lengths.

1.3.6 Degree

The degree of a B-spline curve or surface dictates the degree of each span or sub-patch; the maximum possible degree is defined by the number of control points, i.e. for a curve $p \leq m$, and for a surface $p \leq m$, $q \leq n$. Figure 1.10 illustrates the unconstrained interpolation of five data points by four curves of varying degree. When $p = 1$, the curve is piecewise linear, and has four C^0 continuous spans; it is clearly unsuitable for representing general freeform shapes. The quadratic curve, $p = 2$, has three C^1 continuous spans. In general, quadratic curves do not provide sufficient flexibility for design: a single span cannot represent an inflection, nor can the derivative magnitudes at either end of the span be controlled independently.

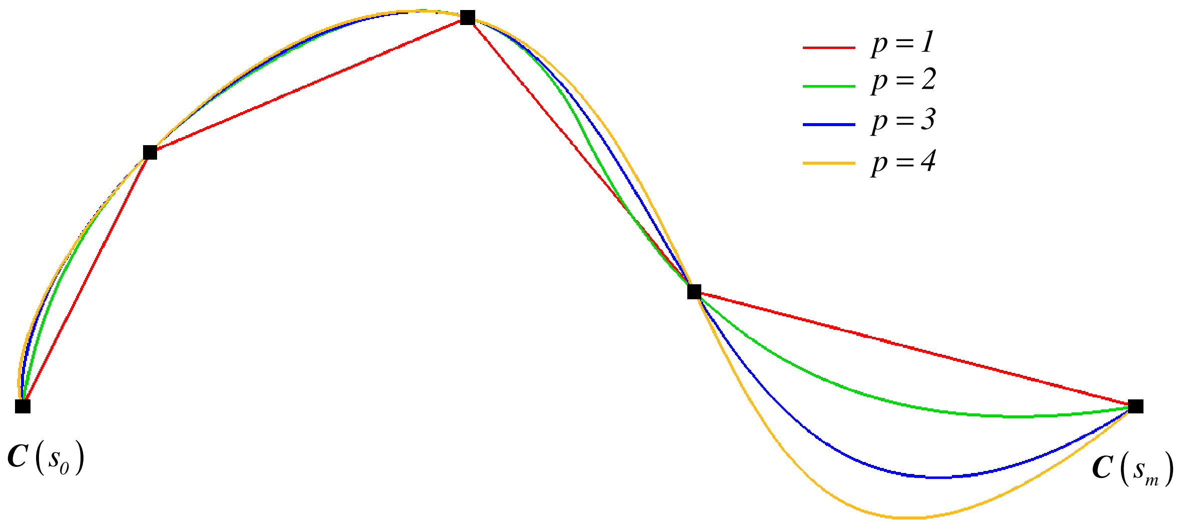


Figure 1.10 – Various degree curves interpolating 5 data points

The cubic curve, $p = 3$, has two C^2 continuous spans, and overcomes the problems encountered by the quadratic case. It is the most frequently used degree for interpolation [Farin, 2002]. The quartic curve, $p = 4$, has a single C^3 continuous span; the additional continuity constraints of polynomials where $p > 3$ often force the curve to 'exaggerate' the features of the interpolated data, and undesirable oscillations may occur. When the flexibility of a higher degree curve is required (e.g. for refinement, manipulation of curvature without affecting tangents, joining to other higher order curves, etc.), it is usual to interpolate with $p = 3$, and then raise to the desired degree (see [Piegl and Tiller, 1997]).

1.3.7 Rational weightings

The control points of a B-spline entity can be represented using homogeneous coordinates, i.e. (wx, wy, wz, w), where the fourth component w is known as the rational weight. The weights can be used to bias the curve or surface towards (or away from) their corresponding control points; this is illustrated in Figure 1.11 where all $w_i = 1$, except for w_2.

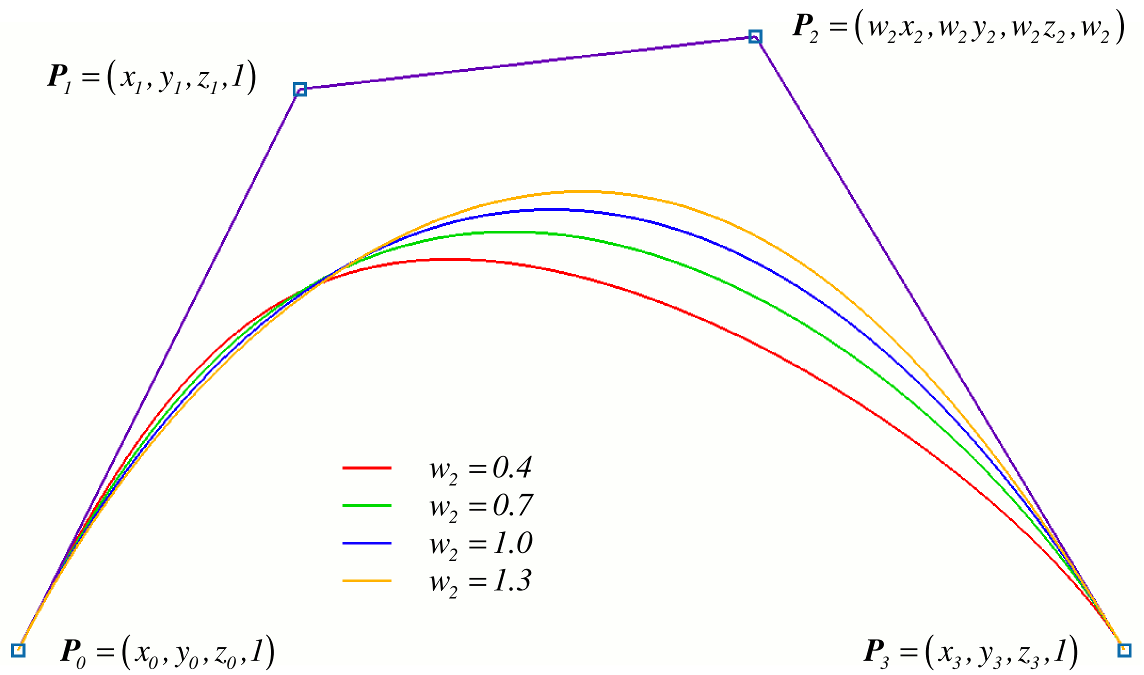

$P_1 = (x_1, y_1, z_1, 1)$
$P_2 = (w_2x_2, w_2y_2, w_2z_2, w_2)$
$w_2 = 0.4$
$w_2 = 0.7$
$w_2 = 1.0$
$w_2 = 1.3$
$P_0 = (x_0, y_0, z_0, 1)$
$P_3 = (x_3, y_3, z_3, 1)$

Figure 1.11 – Effect of various rational weights on a curve

The added degrees of freedom that the rational weights provide can be very useful, as they allow conic entities to be exactly represented, and are beneficial when refining an existing definition. However, when a curve or surface is to be constructed by interpolation, all the weights are set to unity; there is very little advantage in doing otherwise [Piegl and Tiller, 1997], and would result in a non-linear system of equations [Farin, 2002]. One exception to this rule is detailed by Hoschek [1992], and applies only when interpolating piecewise circular arcs.

1.3.8 Number of data points

All of the preceding factors presume that the data points to be interpolated are given; in many circumstances, however, the designer may be able to choose the number of data points to meet a specific quality measure, e.g. positional or curvature error. Supplying additional data points will inevitably reduce the error between the interpolated entity and the sampled geometry. Figure 1.12 illustrates two constrained curves interpolating three ($m = 4$) and five ($m = 6$) evenly spaced points sampled from a unit semi-circle; the resulting curvature profiles are shown as an indication of quality.

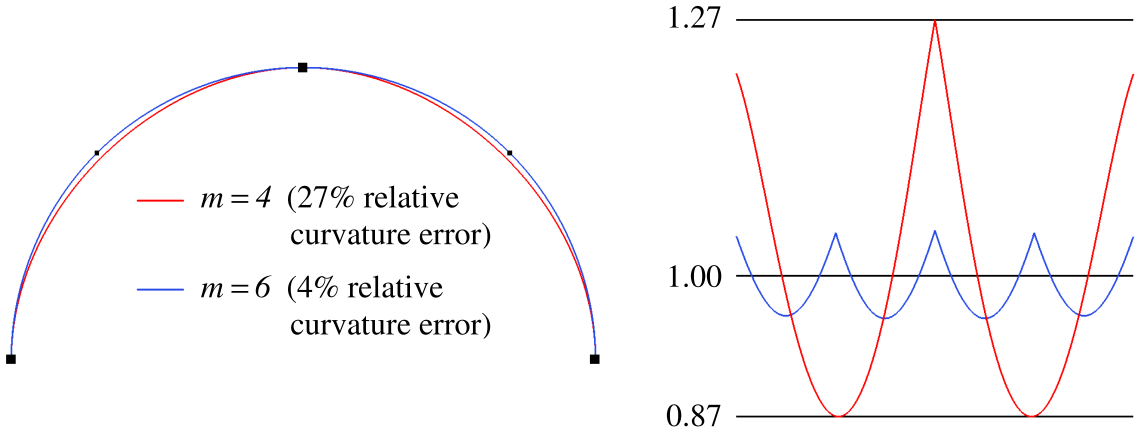


Figure 1.12 – Curvature errors when interpolating 3 and 5 points from a semi circle

The number of data points directly corresponds to the number of control points required to represent the interpolated entity, so it is undesirable to use more points than necessary – doing so can lead to data proliferation and compromise downstream activities [Cripps and Lockyer, 2005]. Also, sampling sparse data points from a poor quality object allows the interpolation process to 'smooth' over undesirable fluctuations in the geometry, whereas a dense point sampling will result in the blemishes being inherited. The aim is therefore to find the minimum number of data points that will satisfy the quality requirements.

1.3.9 Spacing of data points

Closely related to the number of data points is the spacing. Where curvature is constant (lines, planes, circles, cylinders), evenly spaced points yield optimal results. This is not true of entities with varying curvature profiles. Consider the Generalised Cornu Spiral [Ali, *et. al.*, 1999] in Figure 1.13 (denoted by the red dotted line), which has a rational linear curvature profile that monotonically increases over the length of the curve. When points are sampled evenly along the GCS, and then interpolated, it is evident that the region with high curvature does not have a sufficient density of points. The absolute positional error shows this approach is insensitive to curvature variation. An alternative spacing shows far superior positional error across the entire curvature range. It is evident that the method used to space points can have a significant effect on the quality of the interpolated entity; a good algorithm will characterise the geometry of the original shape and minimise positional/curvature errors.

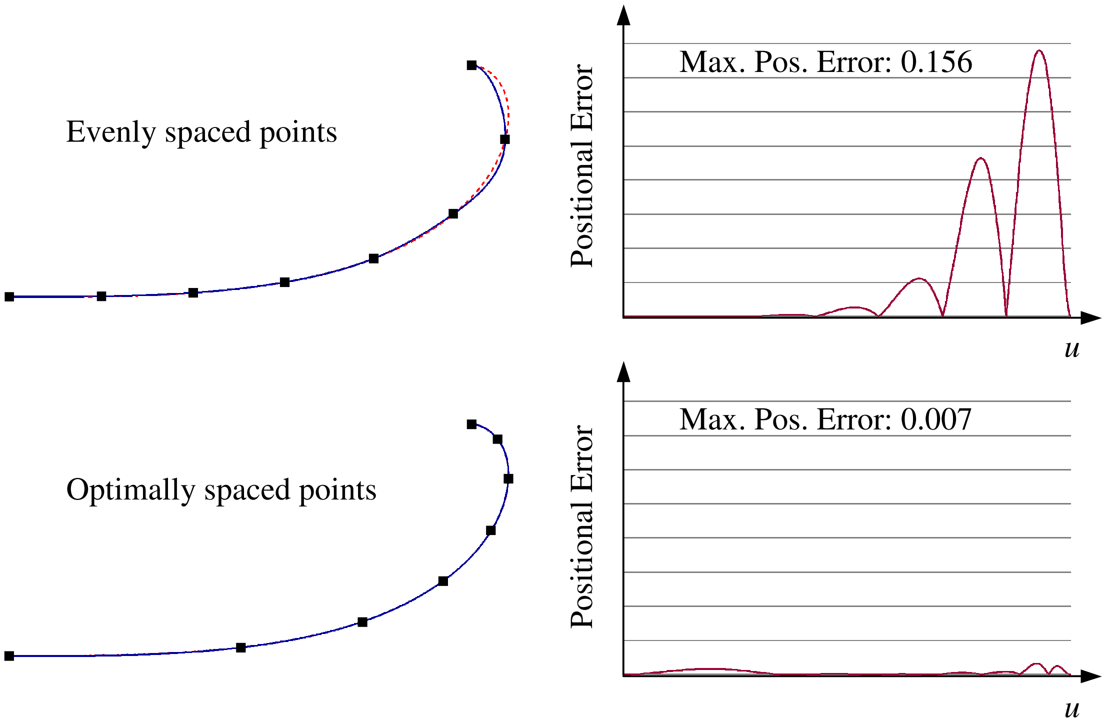


Figure 1.13 – Effect of data point spacing on abs. positional error for constrained curves

1.3.10 Reconciliation of parameter values

When interpolating surfaces, a single parameterisation needs to be obtained for each parametric direction. Consider the ruled surface illustrated in Figure 1.14, where all the data points are sampled from a cylinder. The parameterisation in the *v*-direction is trivial, but the ideal parameterisation for the boundary $\boldsymbol{S}(u,0)$ would be different to that of $\boldsymbol{S}(u,1)$ because the spacing of data points is different. One method to reconcile these is simply to average the corresponding parameter values – the result, however, is that the parameterisation is a compromise for both boundaries, causing distortion and undulations. Unfortunately, some form of parameter reconciliation method is necessary when interpolating surfaces, because global interpolation requires that parallel strips of points have identical parameterisations. The only way to avoid poor quality results is to ensure the reconciliation process causes minimal disruption to the ideal parameter values, i.e. by careful point distribution.

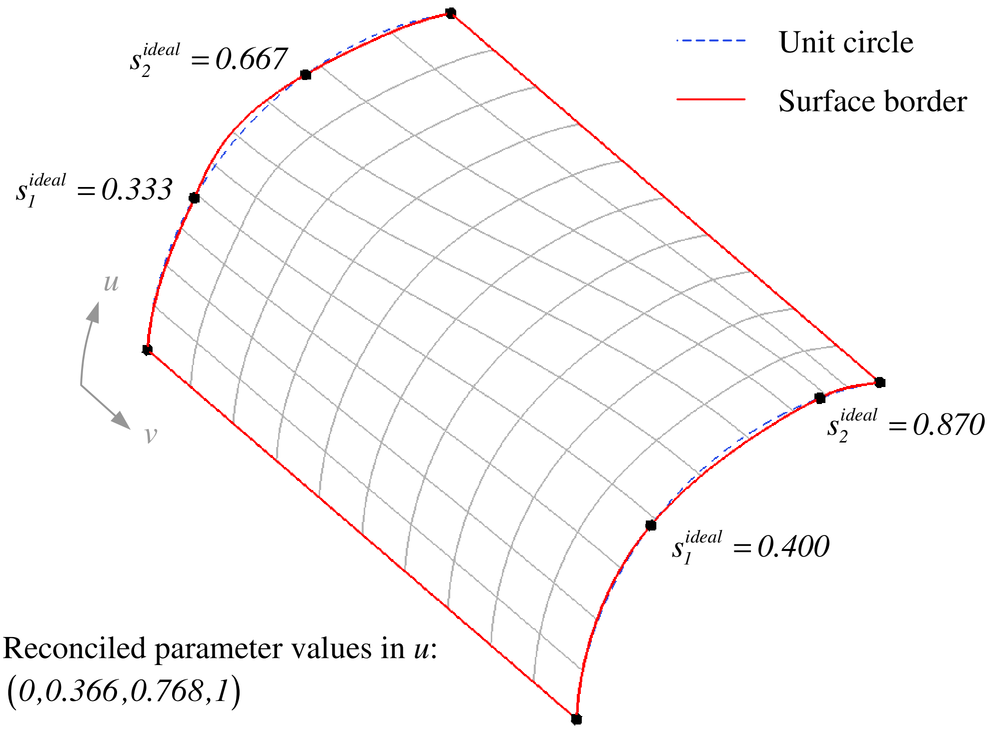


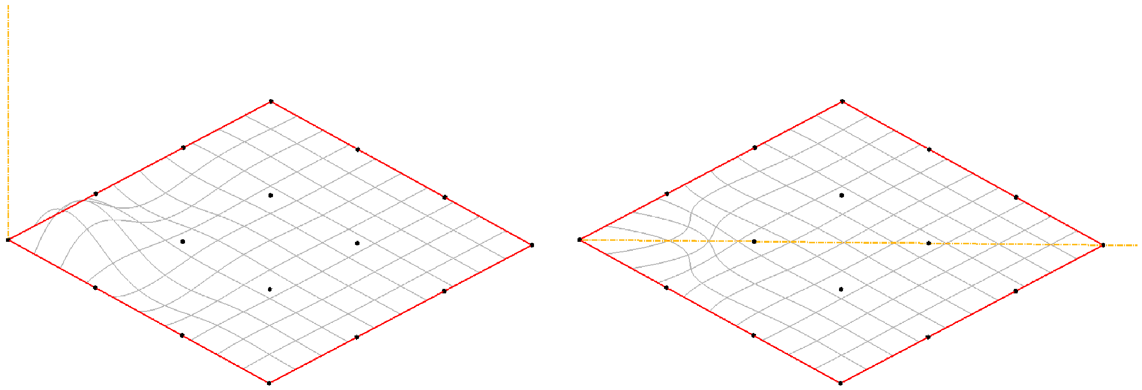
Figure 1.14 – Reconciling parameter values

1.3.11 Twist vector estimation

A NUBS surface contains $(m+1)\times(n+1)=(mn+m+n+1)$ control points, $\boldsymbol{P}_{i,j}$, $0 \le i \le m$, $0 \le j \le n$. When the surface is created using constrained interpolation, one must supply:

- $(m-1)\times(n-1)$ data points, $\boldsymbol{S}(s_k,t_l)$, $0 \le k < m-1$, $0 \le l < n-1$,
- $2(n-1)$ cross-boundary derivatives in the *u*-direction, $\boldsymbol{S}_u(0,t_l)$ and $\boldsymbol{S}_u(1,t_l)$, $0 \le l \le n-2$,
- $2(m-1)$ cross-boundary derivatives in the *v*-direction, $\boldsymbol{S}_v(s_k,0)$ and $\boldsymbol{S}_v(s_k,1)$, $0 \le k \le m-2$.

The data points and cross-boundary derivatives amount to $(mn+m+n-3)$ vectors in total, leaving four outstanding vector-valued pieces of information. These are the mixed partial derivatives from each corner of the interpolated surface, $\boldsymbol{S}_{uv}(a,b)$, $a,b=\{0,1\}$, and are commonly referred to as twist vectors. Mixed partial derivatives are second order derivatives, and being entirely a function of the surface's parameterisation, they have no geometric interpretation. In terms of their effect on the surface, they control the rate of change of the cross boundary derivatives, which must be identical in both parametric directions at the corners. Inappropriate control of the twist vectors can lead to poor surface geometry and/or poor parameterisation. Consider the case of a plane: it should have zero length twist vectors because the rate of change in the cross boundary derivatives is zero. Figure 1.15(a) illustrates the effect of applying a large twist in the normal direction – the interpolated surface is no longer a plane. Figure 1.15(b) illustrates the consequences of a large twist within the plane; the surface remains a plane, but the parameterisation is very poor.



(a) (b)

Figure 1.15 – Effect of poor twists on shape and parameterisation

Estimating an appropriate twist vector for a general freeform surface is very difficult – often the twist vectors are simply set to zero, even though this yields poor results for all but the most basic shapes.

1.4 Motivation of research

It is evident from the numerous factors outlined in Section 1.3 that the interpolation process can yield very poor results if it is not controlled appropriately. The implications of poorly constructed parametric representations affect both subsequent design activities, and analysis/simulation. Smith [1999] reports that poor quality surface representations can also have a serious detrimental effect on downstream machining operations.

To compensate for inadequate control of the interpolation process, designers often increase the amount of data points used to define an entity. This often leads to data proliferation, which is undesirable for storage, transmission and subsequent processing. Cripps and Lockyer [2005] warn of the adverse effects that this can have on follow-on activities. Furthermore, increasing the data set does not always resolve quality related issues; for example, interpolating a smooth data set with a high density of points can induce undulations in the curve if a poor parameterisation is used. Intuitively, one expects that this would be less likely if sparse data points were employed.

Many of the commonly used techniques for controlling the interpolation process have been adopted because they are simple, or have been developed heuristically. A rigorous evaluation of these methods is required to ascertain how well they perform in general, and to investigate if there are more appropriate methods of control. For instance, the total chord length joining a set of data is often used to estimate the derivative magnitudes for a constrained interpolation; however, derivative magnitudes are not a geometric feature, being related to the parameterisation of the curve.

One area where this research will be of considerable benefit is the approximation of curves and surfaces defined using Generalised Cornu Spirals [Ali, *et. al.*, 1999]. GCS entities have many advantageous properties, such as monotonic variation in curvature, but are not supported in existing CAD software. Points can easily be sampled from a GCS definition, but the 'ideal' number and position of those points for interpolation is uncertain. Tangent directions are also easily sampled, but GCS definitions are parameterised by arc length, so derivative magnitudes and twist vectors need to be estimated.

Another application of this work is to construct surfaces with *N*-sides, using standard rectangular surfaces, in order that they can be represented using existing CAD software. The sub-patches that are assembled to form the *N*-sided region are most likely to be constructed using interpolation. Subsequent modification is then required to attain an acceptable level of continuity between sub-patches.

1.5 Outline of thesis

This chapter has provided a background to the interpolation of NUBS curves and surfaces, outlining the factors that need to be controlled for producing quality results. The next chapter details the process for constructing a curve or surface using interpolation, and looks at existing methods for controlling the shape of the interpolant.

Chapter 3 introduces a new parameterisation and derivative magnitude estimation method which strives to maintain orthogonality between the first and second parametric derivatives. The proposed algorithms are assessed analytically for their stability, degeneracy, and sensitivity. Chapter 4 compares the performance of all the algorithms presented in Chapters 2 and 3 using a number of numerical tests. Case studies include interpolating points sampled from a circle, and from GCS definitions.

Chapter 5 evaluates various methods for spacing data points prior to interpolation, and proposes a new method that maintains the sum of projected distances over each span. Chapter 6 proposes an improved twist vector estimation method, based on the Adini algorithm, and is shown to outperform others theoretically and numerically.

Following the examination of factors that control the interpolation process, Chapter 7 considers how interpolated surfaces can be joined together with parametric and geometric continuity. These methods are then applied to a common surface construction problem – creating an *N*-sided surface. Existing solutions to this problem are not generally compatible with most CAD software. However, interpolation can be used to assemble *N* rectangular patches that, after refinement, join with internal geometric continuity, and are compatible with most commercial CAD systems.

Chapter 8 draws conclusions from all aspects of the work detailed in this thesis. Some ideas for future work are also discussed.

Chapter 2

The interpolation process

NUBS interpolation is essentially the process of calculating the solution to a system of linear equations such that the resulting entity satisfies the given data constraints. This chapter reviews the details of this process for both curves and surfaces. Constrained (i.e. using tangent information at boundaries), as well as unconstrained, interpolation is considered. As NUBS interpolation is most commonly implemented in a computer system using matrices, a numerical approach to solving systems of linear equations and inverting matrices is discussed.

The interpolation process requires certain factors to be calculated prior to solving the system of equations. These factors can have significant impact on the quality of the interpolant, and include the parameterisation, derivative magnitude estimation, knot vector generation, parameter reconciliation and twist vector estimation. This chapter presents existing methods for controlling these factors, which are then assessed alongside new methods in subsequent chapters.

2.1 Review of the interpolation process

2.1.1 Unconstrained curve interpolation

To interpolate a set of points with a NUBS curve, the following data is required:

- A set of data points, $\boldsymbol{C}(s_k)$, $0 \le k \le m$
- The parameter values, s_k, $0 \le k \le m$
- The degree, p, where $p \le m$
- An open clamped knot vector, $\boldsymbol{U} = \{u_0, \ldots, u_{m+p+1}\}$, in the form of (1.21)

An $(m+1) \times (m+1)$ system of linear equations can be set up to solve for the (x, y, z) components of $\boldsymbol{P}_i$, $0 \le i \le m$:

$$\boldsymbol{C}(s_k) = \sum_{i=0}^{m} N_{i,p}(s_k)\boldsymbol{P}_i \qquad 0 \le k \le m \tag{2.1}$$

This can be expressed in matrix form:

$$[\boldsymbol{C}] = [\boldsymbol{M}][\boldsymbol{P}] \tag{2.2}$$

$$[\boldsymbol{P}] = [\boldsymbol{M}]^{-1}[\boldsymbol{C}] \tag{2.3}$$

where $[\boldsymbol{C}]$ is a $(m+1) \times 1$ matrix of 3D points, $\boldsymbol{C}(s_k)$, $0 \le k \le m$,

$[\boldsymbol{M}]$ is a $(m+1) \times (m+1)$ matrix containing $N_{i,p}(s_k)$, $0 \le i, k \le m$, and

$[\boldsymbol{P}]$ is a $(m+1) \times 1$ matrix of 3D points that will receive $\boldsymbol{P}_i$, $0 \le i \le m$.

$[\boldsymbol{C}]$ holds a single (x, y, z) data point on each row; $[\boldsymbol{P}]$ has an identical structure, and receives the calculated control points. $[\boldsymbol{M}]$ is constructed such that each row contains a complete set of basis functions corresponding to a single parameter value, i.e.:

$$[\boldsymbol{M}]_{k,i} = N_{i,p}(s_k) \qquad 0 \le i, k \le m \tag{2.4}$$

Note that $\left[\boldsymbol{M}\right]$ is banded if the knot vector is constructed such that each internal interval contains no more than $p+1$ data constraints (see Section 1.3.3, also de Boor [2001]), because the basis functions have only a localised effect:

$$\left[\boldsymbol{M}\right]=\begin{bmatrix} N_{0,p}(s_0)\equiv 1 & 0 & 0 & \cdots & 0 & 0 \\ N_{0,p}(s_1) & N_{1,p}(s_1) & N_{2,p}(s_1) & \cdots & 0 & 0 \\ 0 & N_{1,p}(s_2) & N_{2,p}(s_2) & \cdots & N_{m-1,p}(s_2) & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & N_{1,p}(s_{m-2}) & N_{2,p}(s_{m-2}) & \cdots & N_{m-1,p}(s_{m-2}) & 0 \\ 0 & 0 & N_{2,p}(s_{m-1}) & \cdots & N_{m-1,p}(s_{m-1}) & N_{m,p}(s_{m-1}) \\ 0 & 0 & 0 & \cdots & 0 & N_{m,p}(s_m)\equiv 1 \end{bmatrix} \tag{2.5}$$

The solution to (2.3) is found by inverting (2.5) – see Section 2.1.5.

2.1.2 Constrained curve interpolation

A constrained curve interpolation gives rise to two additional control points. The following data is required:

- A set of data points, $\boldsymbol{C}(s_k)$, $0 \le k \le m-2$
- The parameter values, s_k, $0 \le k \le m-2$
- The degree, *p*, where $p \le m$
- An open clamped knot vector, $\boldsymbol{U}=\left\{u_0,...,u_{m+p+1}\right\}$, in the form of (1.21)
- Start and end derivatives, $\boldsymbol{C}_u(0)$ and $\boldsymbol{C}_u(1)$, both directions and magnitudes

The $(m+1)\times(m+1)$ system of linear equations is constructed using matrix form in exactly the same way as (2.3); however $\left[\boldsymbol{M}\right]$ and $\left[\boldsymbol{C}\right]$ need to be modified to accept the derivative constraints. The data points only provide $(m-1)$ equations – the additional two are provided by considering the derivatives [Piegl and Tiller, 1997]:

$$-\boldsymbol{P}_0+\boldsymbol{P}_1=\frac{u_{p+1}}{p}\boldsymbol{C}_u(0) \tag{2.6}$$

$$-\boldsymbol{P}_{m-1}+\boldsymbol{P}_m=\frac{1-u_m}{p}\boldsymbol{C}_u(1) \tag{2.7}$$

The two equations, (2.6) and (2.7), are inserted into $[\boldsymbol{M}]$ as the second and penultimate rows:

$$[\boldsymbol{M}]=\begin{bmatrix} N_{0,p}(s_0)\equiv 1 & 0 & 0 & \cdots & 0 & 0 \\ -p/u_{p+1} & p/u_{p+1} & 0 & \cdots & 0 & 0 \\ N_{0,p}(s_1) & N_{1,p}(s_1) & N_{2,p}(s_1) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & N_{2,p}(s_{m-3}) & \cdots & N_{m-1,p}(s_{m-3}) & N_{m,p}(s_{m-3}) \\ 0 & 0 & 0 & \cdots & -p/(1-u_m) & p/(1-u_m) \\ 0 & 0 & 0 & \cdots & 0 & N_{m,p}(s_{m-2})\equiv 1 \end{bmatrix} \tag{2.8}$$

$[\boldsymbol{C}]$ needs to be correspondingly modified by inserting the start and end derivatives, $\boldsymbol{C}_u(0)$, and $\boldsymbol{C}_u(1)$, as the second and penultimate rows respectively. The control points are obtained by inverting (2.8) and solving (2.3) as before.

2.1.3 Unconstrained surface interpolation

To interpolate a set of data points with a NUBS surface, the following data is required:

- A grid of data points, $\boldsymbol{S}(s_k,t_l)$, $0\le k\le m$, $0\le l\le n$
- The parameter values, s_k, $0\le k\le m$, and t_l, $0\le l\le n$
- The degrees p and q, where $p\le m$, $q\le n$
- Two open clamped knot vectors, $\boldsymbol{U}=\{u_0,\ldots,u_{m+p+1}\}$ and $\boldsymbol{V}=\{v_0,\ldots,v_{n+q+1}\}$, in the form of (1.21)

As for curves, a system of linear equations can be set up to solve for the (x, y, z) components of $\boldsymbol{P}_{i,j}$, $0 \le i \le m$, $0 \le j \le n$:

$$\boldsymbol{S}(s_k, t_l) = \sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,p}(s_k) N_{j,q}(t_l) \boldsymbol{P}_{i,j} \qquad 0 \le k \le m,\ 0 \le l \le n \qquad (2.9)$$

This can be expressed in matrix form:

$$[\boldsymbol{S}] = [\boldsymbol{M}][\boldsymbol{P}][\boldsymbol{N}] \qquad (2.10)$$

$$[\boldsymbol{P}] = [\boldsymbol{M}]^{-1}[\boldsymbol{S}][\boldsymbol{N}]^{-1} \qquad (2.11)$$

where $[\boldsymbol{S}]$ is a $(m+1)\times(n+1)$ matrix of 3D data points, containing $\boldsymbol{S}(s_k, t_l)$, $0 \le k \le m$, $0 \le l \le n$,

$[\boldsymbol{M}]$ is a $(m+1)\times(m+1)$ matrix containing $N_{i,p}(s_k)$, $0 \le i,k \le m$,

$[\boldsymbol{N}]$ is a $(n+1)\times(n+1)$ matrix containing $N_{j,q}(t_l)$, $0 \le j,l \le m$, and

$[\boldsymbol{P}]$ is a $(m+1)\times(n+1)$ matrix of 3D points, that will receive $\boldsymbol{P}_{i,j}$, $0 \le i \le m$, $0 \le j \le n$.

$[\boldsymbol{M}]$ is constructed such that each row contains a complete set of basis functions corresponding to a single parameter value, i.e.:

$$[\boldsymbol{M}]_{k,i} = N_{i,p}(s_k) \qquad 0 \le i,k \le m \qquad (2.4)$$

$[\boldsymbol{N}]$ is constructed analogously, but is the transpose:

$$[\boldsymbol{N}]_{j,l} = N_{j,q}(t_l) \qquad 0 \le j,l \le n \qquad (2.12)$$

Given that $\boldsymbol{U}$ and $\boldsymbol{V}$ are constructed such that each internal interval contains at most $p+1$ and $q+1$ data constraints respectively, $[\boldsymbol{M}]$ and $[\boldsymbol{N}]$ will be banded [de Boor, 2001]; $[\boldsymbol{M}]$ is of the form (2.5), and $[\boldsymbol{N}]$ is analogous except that it is transposed. They are then inverted to provide a solution to (2.11) – see Section 2.1.5.

2.1.4 Constrained surface interpolation

To interpolate a set of data points and derivative constraints with a NUBS surface, the following data is required:

- A grid of data points, $\boldsymbol{S}(s_k,t_l)$, $0 \le k \le m-2$, $0 \le l \le n-2$
- The parameter values, s_k, $0 \le k \le m-2$, and t_l, $0 \le l \le n-2$
- The degrees p and q, where $p \le m$, $q \le n$
- Two open clamped knot vectors, $\boldsymbol{U} = \{u_0,...,u_{m+p+1}\}$ and $\boldsymbol{V} = \{v_0,...,v_{n+q+1}\}$, in the form of (1.21)
- The start and end cross-boundary derivatives in the *u*-direction, $\boldsymbol{S}_u(0,t_l)$ and $\boldsymbol{S}_u(1,t_l)$, $0 \le l \le n-2$
- The start and end cross-boundary derivatives in the *v*-direction, $\boldsymbol{S}_v(s_k,0)$ and $\boldsymbol{S}_v(s_k,1)$, $0 \le k \le m-2$
- The four corner twist vectors, $\boldsymbol{S}_{uv}(a,b)$, $a,b = \{0,1\}$

The system of linear equations is constructed using matrix form in exactly the same way as (2.11); however, $[\boldsymbol{M}]$, $[\boldsymbol{N}]$ and $[\boldsymbol{S}]$ need to be modified to accept the derivative constraints. $[\boldsymbol{S}]$ remains a $(m+1)\times(n+1)$ array of 3D vectors, although only $(m-1)\times(n-1)$ data points are supplied. The cross boundary derivatives and the twist vectors satisfy the remaining equations. In general, when $p = q = 3$, $[\boldsymbol{S}]$ is constructed as follows:

$$[\boldsymbol{S}]=\begin{bmatrix} \boldsymbol{Q}_{0,0} & \boldsymbol{D}^{v}_{0,0} & \boldsymbol{Q}_{0,1} & \cdots & \boldsymbol{Q}_{0,n-3} & \boldsymbol{D}^{v}_{0,n-2} & \boldsymbol{Q}_{0,n-2} \\ \boldsymbol{D}^{u}_{0,0} & \boldsymbol{T}_{0,0} & \boldsymbol{D}^{u}_{0,1} & \cdots & \boldsymbol{D}^{u}_{0,n-3} & \boldsymbol{T}_{0,n-2} & \boldsymbol{D}^{u}_{0,n-2} \\ \boldsymbol{Q}_{1,0} & \boldsymbol{D}^{v}_{1,0} & \boldsymbol{Q}_{1,1} & \cdots & \boldsymbol{Q}_{1,n-3} & \boldsymbol{D}^{v}_{1,n-2} & \boldsymbol{Q}_{1,n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \boldsymbol{Q}_{m-3,0} & \boldsymbol{D}^{v}_{m-3,0} & \boldsymbol{Q}_{m-3,1} & \cdots & \boldsymbol{Q}_{m-3,n-3} & \boldsymbol{D}^{v}_{m-3,n-2} & \boldsymbol{Q}_{m-3,n-2} \\ \boldsymbol{D}^{u}_{m-2,0} & \boldsymbol{T}_{m-2,0} & \boldsymbol{D}^{u}_{m-2,1} & \cdots & \boldsymbol{D}^{u}_{m-2,n-3} & \boldsymbol{T}_{m-2,n-2} & \boldsymbol{D}^{u}_{m-2,n-2} \\ \boldsymbol{Q}_{m-2,0} & \boldsymbol{D}^{v}_{m-2,0} & \boldsymbol{Q}_{m-2,1} & \cdots & \boldsymbol{Q}_{m-2,n-3} & \boldsymbol{D}^{v}_{m-2,n-2} & \boldsymbol{Q}_{m-2,n-2} \end{bmatrix} \tag{2.13}$$

where the following notation is used for brevity:

$$\boldsymbol{Q}_{k,l}=\boldsymbol{S}\left(s_k,t_l\right),\ \boldsymbol{D}^{u}_{k,l}=\boldsymbol{S}_u\left(s_k,t_l\right),\ \boldsymbol{D}^{v}_{k,l}=\boldsymbol{S}_v\left(s_k,t_l\right),\ \boldsymbol{T}_{k,l}=\boldsymbol{S}_{uv}\left(s_k,t_l\right),$$

$$0\le k\le m-2,\ t_l,\ 0\le l\le n-2$$

When m or n is less than 5, the generality of (2.13) is less apparent, so two specific examples are given to aid clarity. Firstly, when $m=n=3$:

$$[\boldsymbol{S}]=\begin{bmatrix} \boldsymbol{Q}_{0,0} & \boldsymbol{D}^{v}_{0,0} & \boldsymbol{D}^{v}_{0,1} & \boldsymbol{Q}_{0,1} \\ \boldsymbol{D}^{u}_{0,0} & \boldsymbol{T}_{0,0} & \boldsymbol{T}_{0,1} & \boldsymbol{D}^{u}_{0,1} \\ \boldsymbol{D}^{u}_{1,0} & \boldsymbol{T}_{1,0} & \boldsymbol{T}_{1,1} & \boldsymbol{D}^{u}_{1,1} \\ \boldsymbol{Q}_{1,0} & \boldsymbol{D}^{v}_{1,0} & \boldsymbol{D}^{v}_{1,1} & \boldsymbol{Q}_{1,1} \end{bmatrix}$$

and when $m=n=4$:

$$[\boldsymbol{S}]=\begin{bmatrix} \boldsymbol{Q}_{0,0} & \boldsymbol{D}^{v}_{0,0} & \boldsymbol{Q}_{0,1} & \boldsymbol{D}^{v}_{0,2} & \boldsymbol{Q}_{0,2} \\ \boldsymbol{D}^{u}_{0,0} & \boldsymbol{T}_{0,0} & \boldsymbol{D}^{u}_{0,1} & \boldsymbol{T}_{0,2} & \boldsymbol{D}^{u}_{0,2} \\ \boldsymbol{Q}_{1,0} & \boldsymbol{D}^{v}_{1,0} & \boldsymbol{Q}_{1,1} & \boldsymbol{D}^{v}_{1,2} & \boldsymbol{Q}_{1,2} \\ \boldsymbol{D}^{u}_{2,0} & \boldsymbol{T}_{2,0} & \boldsymbol{D}^{u}_{2,1} & \boldsymbol{T}_{2,2} & \boldsymbol{D}^{u}_{2,2} \\ \boldsymbol{Q}_{2,0} & \boldsymbol{D}^{v}_{2,0} & \boldsymbol{Q}_{2,1} & \boldsymbol{D}^{v}_{2,2} & \boldsymbol{Q}_{2,2} \end{bmatrix}$$

It is noted that there is no requirement for $m\equiv n$.

When $p=3$, the matrix $[\boldsymbol{M}]$ has two additional rows for the differentiated basis functions; these are the second and penultimate rows, i.e.:

$$[\boldsymbol{M}] = \begin{bmatrix} N_{0,p}(s_0) & N_{1,p}(s_0) & \cdots & N_{m-1,p}(s_0) & N_{m,p}(s_0) \\ N'_{0,p}(s_0) & N'_{1,p}(s_0) & \cdots & N'_{m-1,p}(s_0) & N'_{m,p}(s_0) \\ N_{0,p}(s_1) & N_{1,p}(s_1) & \cdots & N_{m-1,p}(s_1) & N_{m,p}(s_1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ N_{0,p}(s_{m-3}) & N_{1,p}(s_{m-3}) & \cdots & N_{m-1,p}(s_{m-3}) & N_{m,p}(s_{m-3}) \\ N'_{0,p}(s_{m-2}) & N'_{1,p}(s_{m-2}) & \cdots & N'_{m-1,p}(s_{m-2}) & N'_{m,p}(s_{m-2}) \\ N_{0,p}(s_{m-2}) & N_{1,p}(s_{m-2}) & \cdots & N_{m-1,p}(s_{m-2}) & N_{m,p}(s_{m-2}) \end{bmatrix} \tag{2.14}$$

$[\boldsymbol{N}]$ is constructed analogously, but its construction is the transpose of $[\boldsymbol{M}]$. The control points, $\boldsymbol{P}_{i,j}$, $0 \le i \le m$, $0 \le j \le n$, are obtained by evaluating (2.11).

2.1.5 Matrix inversion

Matrix inversion is implicitly linked to solving systems of linear equations. Consider the following linear system, where $[\boldsymbol{A}]$ is a square matrix of order *n*, and $[\boldsymbol{b}]$ is an *n*-vector:

$$[\boldsymbol{A}][\boldsymbol{x}] = [\boldsymbol{b}] \tag{2.15}$$

The system has a unique solution, $[\boldsymbol{x}]$, if $[\boldsymbol{A}]$ is invertible. The solution is easily found if $[\boldsymbol{A}]$ is in upper-triangular form, with all diagonal entries non-zero, by means of back-substitution [Conte and de Boor, 1981]:

$$x_k = \frac{b_k - \sum_{j=k+1}^{n} a_{k,j} x_j}{a_{k,k}} \tag{2.16}$$

Where $[\boldsymbol{A}]$ is not in upper-triangular form, an equivalent system can be created by performing row and column operations, i.e. Gaussian elimination. It is of utmost importance to preserve numerical accuracy, and therefore the pivotal equation used for elimination should be chosen carefully. Conte and de Boor [1981] recommend a strategy called scaled partial pivoting which aims to select the equation with the largest coefficient of x_k relative to the size of the other coefficients in that row, thus avoiding division by near-zero coefficients.

An alternative approach to solving systems of linear equations is to use LU decomposition [Conte and de Boor, 1981], which factorises the coefficient matrix $[\boldsymbol{A}]$ into three:

$$[\boldsymbol{A}] = [\boldsymbol{P}][\boldsymbol{L}][\boldsymbol{U}]$$

where $[\boldsymbol{P}]$ is a permutation matrix that accounts for row interchanges, $[\boldsymbol{L}]$ a lower-triangular matrix containing multipliers, and $[\boldsymbol{U}]$ an upper-triangular matrix. The solution, $[\boldsymbol{x}]$, is found by forward- and back-substitution. The principle advantage to this approach is that it requires substantially fewer floating-point operations, thereby preserving numerical accuracy.

The inverse of $[\boldsymbol{A}]$, if it exists, can be found by considering that the product $[\boldsymbol{A}][\boldsymbol{A}]^{-1} = [\boldsymbol{I}]$. The *j*th column of $[\boldsymbol{A}]^{-1}$ is the solution of the linear equation:

$$[\boldsymbol{A}][\boldsymbol{x}] = [\boldsymbol{i}_j] \qquad (2.17)$$

where $[\boldsymbol{i}_j]$ is the *j*th column of the identity matrix. The system of linear equations need only be solved once if the row operations used during elimination are applied to each column of the identity matrix. This gives an efficient method for finding the inverse of a matrix.

2.2 Parameterisation methods

The interpolated curve or surface passes through the specified data points exactly – each data point therefore has a corresponding parameter value, s_k, $0 \le k \le m$, or two values in the case of a surface, s_k and t_l, $0 \le k \le m$, $0 \le l \le n$. Parameterisation is the process of determining these values, and can have a significant impact on the resulting entity. This section summarises several commonly used parameterisation methods. The performance of these methods is assessed numerically in Chapter 4 alongside the new methods, which are presented in the next chapter.

When a curve is constructed by an unconstrained interpolation, there are $(m+1)$ data points. Generally, parameter values are assigned in the range $[0,1]$, therefore $s_0 = 0$ and $s_m = 1$. All of the parameterisation algorithms presented in this section assume that a curve is being interpolated, and it is unconstrained. For surfaces, these algorithms can easily be applied to the rows and columns of data points. Where the interpolation is constrained, there are only $(m-1)$ data points, so *m* becomes $(m-2)$. For brevity, data points are denoted $\boldsymbol{Q}_k$, i.e. $\boldsymbol{Q}_k = \boldsymbol{C}(s_k)$, $0 \le k \le m$.

2.2.1 Equally spaced parameterisation

The simplest parameterisation method equally spaces the parameter values over $[0,1]$:

$$s_k = \frac{k}{m} \qquad\qquad 0 \le k \le m \qquad\qquad (2.18)$$

Equally spaced parameterisation is sometimes called uniform parameterisation. As the parametric interval between data points is constant, the resulting curve is often distorted when the points are unevenly distributed.

2.2.2 Chord length parameterisation

A number of parameterisations are distance-based; they aim to produce parameter values that vary linearly with arc length so that they reflect the relative spacing of data points. One such parameterisation is the cumulative chord length method [Ahlberg, et. al., 1967], and is probably the most commonly used parameterisation [Piegl and Tiller, 1997]. It determines the parameter values according to the proportional distances between data points:

$$s_0 = 0$$
$$s_k = s_{k-1} + \frac{\left\| \boldsymbol{Q}_k - \boldsymbol{Q}_{k-1} \right\|}{\sum_{c=1}^{m} \left\| \boldsymbol{Q}_c - \boldsymbol{Q}_{c-1} \right\|} \qquad 1 \le k \le m \tag{2.19}$$

The chord length is a reasonable approximation to the arc length if sufficient data points are used to control the curvature of the interpolant; as $(m+1) \rightarrow \infty$, the chord length becomes identical to the arc length. However, more points than necessary may be required to attain a specific curvature tolerance using the chord length parameterisation, which is undesirable as it can lead to data proliferation [Cripps and Lockyer, 2005].

2.2.3 Centripetal parameterisation

Centripetal parameterisation [Lee, 1989] is a variation on chord length, and is formulated by taking the square root of all distances in (2.19), i.e.:

$$s_0 = 0$$
$$s_k = s_{k-1} + \frac{\sqrt{\left\| \boldsymbol{Q}_k - \boldsymbol{Q}_{k-1} \right\|}}{\sum_{c=1}^{m} \sqrt{\left\| \boldsymbol{Q}_c - \boldsymbol{Q}_{c-1} \right\|}} \qquad 1 \le k \le m \tag{2.20}$$

Lee [1989] reports that this method almost invariably performs better than chord length parameterisation, especially when the interpolated data contains very sharp changes in direction.

2.2.4 Average parameterisations

Two further parameterisations are obtained by taking the geometric and harmonic averages of the three preceding parameterisations, i.e. the geometric parameterisation is given by:

$$s_k = \sqrt[3]{s_k^{eq} \cdot s_k^{ch} \cdot s_k^{ce}} \qquad 0 \le k \le m \qquad (2.21)$$

where s_k^{eq}, s_k^{ch} and s_k^{ce} represent the equally spaced, chord length and centripetal parameterisations respectively. The harmonic parameterisation is given by:

$$s_k = \frac{3}{\frac{1}{s_k^{eq}} + \frac{1}{s_k^{ch}} + \frac{1}{s_k^{ce}}} \qquad 0 \le k \le m \qquad (2.22)$$

The average parameterisations are adversely affected if any of the constituent parameterisations produce poor parameter values; each of the methods have strengths and weaknesses with different geometric configurations, implying that the resulting parameterisation will always be a compromise.

2.2.5 Circular arc parameterisation

The circular arc parameterisation is similar to chord length, but approximates the distance between points using circular arcs instead of straight lines. Consider the four points illustrated in Figure 2.1.

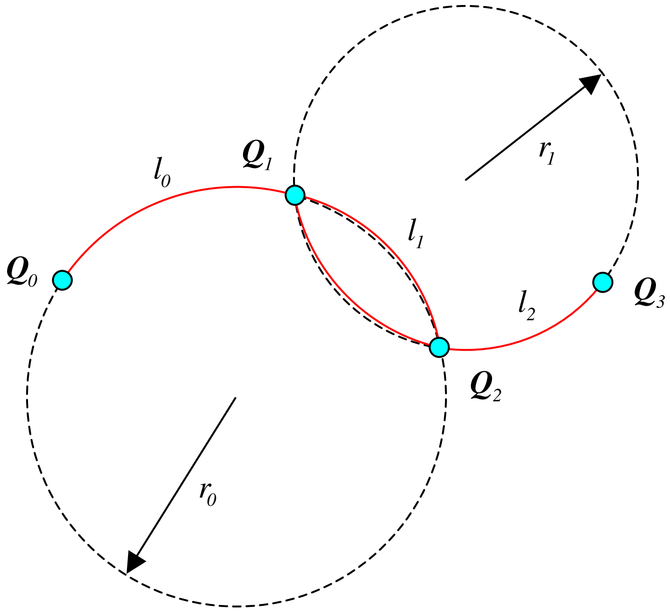


Figure 2.1 - Construction for circular arc parameterisation

A circular arc can be passed through the first three points, $\boldsymbol{Q}_0 \ldots \boldsymbol{Q}_2$, and has a radius r_0. The associated span length, l_0, can be obtained from the two points, $\boldsymbol{Q}_0$ and $\boldsymbol{Q}_1$, and the radius r_0. The last span length, l_2, can be obtained analogously. Intermediate spans have four surrounding points, so an average radius is used, e.g. l_1 is calculated using the points $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$ and the radius $\frac{r_0 + r_1}{2}$. Therefore, l_k is given by:

$$l_k = \begin{cases} 2r_0 \sin^{-1}\left(\dfrac{\left\|\boldsymbol{Q}_1 - \boldsymbol{Q}_0\right\|}{2r_0}\right) & \text{if } k = 0 \\ 2r_{m-2} \sin^{-1}\left(\dfrac{\left\|\boldsymbol{Q}_m - \boldsymbol{Q}_{m-1}\right\|}{2r_{m-2}}\right) & \text{if } k = m-1 \qquad 0 \le k \le m-1 \\ \left(r_{k-1} + r_k\right)\sin^{-1}\left(\dfrac{\left\|\boldsymbol{Q}_{k+1} - \boldsymbol{Q}_k\right\|}{r_{k-1} + r_k}\right) & \text{otherwise} \end{cases} \tag{2.23}$$

where the radii, r_k, $0 \le k \le m-2$, are given in terms of the data points [Faux and Pratt, 1979]:

$$r_k = \frac{\left\|\boldsymbol{Q}_{k+1} - \boldsymbol{Q}_k\right\| \left\|\boldsymbol{Q}_{k+2} - \boldsymbol{Q}_k\right\| \left\|\boldsymbol{Q}_{k+2} - \boldsymbol{Q}_{k+1}\right\|}{2\left\|\left(\boldsymbol{Q}_{k+1} - \boldsymbol{Q}_k\right) \times \left(\boldsymbol{Q}_{k+2} - \boldsymbol{Q}_k\right)\right\|} \qquad 0 \le k \le m-2 \qquad (2.24)$$

Despite the fact that there are an infinite number of arcs of fixed radius that can pass through two points (in 3D), the arc length is common to them all.

Circular arc parameterisation is calculated as follows:

$$\begin{aligned} s_0 &= 0 \\ s_k &= s_{k-1} + \frac{l_{k-1}}{\sum_{c=1}^{m-1} l_c} \qquad 1 \le k \le m \end{aligned} \qquad (2.25)$$

Circular arc parameterisation is unstable when three adjacent data points are collinear to within numerical tolerances. This causes the denominator of (2.24) to become zero, giving an infinite radius, r_k. However, this is simple to check for and avoid, replacing the value of l_k from (2.23) with $\left\|\boldsymbol{Q}_{k+1} - \boldsymbol{Q}_k\right\|$ when $\sin^{-1}\left(\frac{\left\|\left(\boldsymbol{Q}_{k+1} - \boldsymbol{Q}_k\right) \times \left(\boldsymbol{Q}_{k+2} - \boldsymbol{Q}_k\right)\right\|}{\left\|\left(\boldsymbol{Q}_{k+1} - \boldsymbol{Q}_k\right)\right\| \left\|\left(\boldsymbol{Q}_{k+2} - \boldsymbol{Q}_k\right)\right\|}\right) < \delta$, where δ is a small angle tolerance; a value of $\delta = 0.0002$ radians (approx. $0.01°$) is adopted in the following examples.

2.2.6 Farin's parameterisations

Farin [1988] proposes two parameterisation methods in the context of constructing a C^1 piecewise cubic curve. His development uses the Hermite equation form:

$$\begin{aligned} \boldsymbol{C}(u) = &\left(1 - 3t^2 + 2t^3\right)\boldsymbol{C}(s_k) + \left(3t^2 - 2t^3\right)\boldsymbol{C}(s_{k+1}) \\ &+ \left(t - 2t^2 + t^3\right)\Delta_k \boldsymbol{C}_u(s_k) + \left(-t^2 + t^3\right)\Delta_k \boldsymbol{C}_u(s_{k+1}) \end{aligned} \qquad 0 \le k < m \qquad (2.26)$$

where $\Delta_k = \left(s_{k+1} - s_k\right)$, and $t = \left(u - s_k\right)/\Delta_k$ is the local parameter of each segment. Both data points, $\boldsymbol{Q}_k$, and tangent directions, $\boldsymbol{t}_k \equiv \boldsymbol{C}_u(s_k) \big/ \left\|\boldsymbol{C}_u(s_k)\right\|$, $0 \le k \le m$, are required. The derivative magnitudes for each span must be determined, i.e.:

$$a_k \equiv \Delta_k \left\| \boldsymbol{C}_u\left(s_k\right) \right\| \qquad 0 \le k < m \qquad (2.27)$$

$$b_k \equiv \Delta_k \left\| \boldsymbol{C}_u\left(s_{k+1}\right) \right\| \qquad 0 \le k < m \qquad (2.28)$$

Farin's first method, a 'quick and easy' solution, obtains values for a_k and b_k in proportion to the chord length:

$$a_k = b_k = 1.2c_k \qquad 0 \le k < m \qquad (2.29)$$

where c_k is the chord length:

$$c_k = \left\| \boldsymbol{Q}_{k+1} - \boldsymbol{Q}_k \right\| \qquad 0 \le k < m \qquad (2.30)$$

The parameter values, s_k, $0 \le k \le m$, can then be solved uniquely using the first order parametric continuity condition:

$$\frac{a_k}{\Delta_k} = \frac{b_{k-1}}{\Delta_{k-1}} \qquad 1 \le k < m \qquad (2.31)$$

assuming $s_0 = 0$ and $s_m = 1$. Alternatively, the parameter values can be obtained more efficiently by assuming $s_0 = 0$ and $s_1 = 1$, then calculating s_{k+1}:

$$s_{k+1} = \frac{a_k\left(s_k - s_{k-1}\right)}{b_{k-1}} + s_k \qquad 1 \le k < m \qquad (2.32)$$

The parameter values will be in the range $0 \le s_k \le s_m$, and therefore need to be scaled down by a factor of s_m to be in the range [0,1].

Farin [1988] proposes a more 'sophisticated' method, which ensures that the parameterisation of each segment is linear in the direction of the chord, see Figure 2.2.

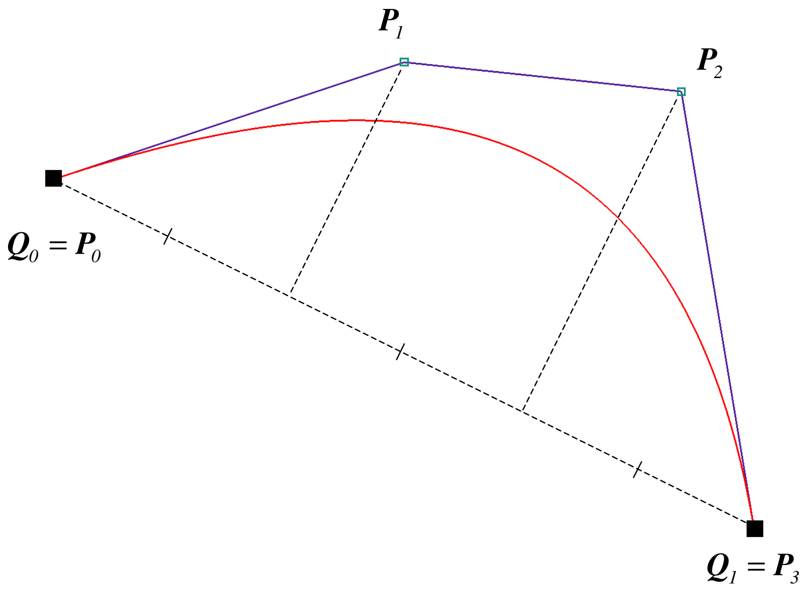


Figure 2.2 - A planar curve with uniform parameterisation along the chord, showing inner Bézier control points

As the angle between the chord and tangent increases, the magnitudes become unreasonably large, and at 90°, the numerical stability of the algorithm is compromised. Farin therefore limits the angle to a maximum of 60°, although little justification is given for this specific value other than convenience, as $\cos 60° = 1/2$. a_k and b_k can be found by trigonometric relationships:

$$a_k = \frac{c_k}{\max\left(\boldsymbol{s}_k \cdot \boldsymbol{t}_k, 0.5\right)} \qquad 0 \le k < m \qquad (2.33)$$

$$b_k = \frac{c_k}{\max\left(\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1}, 0.5\right)} \qquad 0 \le k < m \qquad (2.34)$$

where $\boldsymbol{s}_k$ is the unit direction of the chord:

$$\boldsymbol{s}_k = \left(\boldsymbol{Q}_{k+1} - \boldsymbol{Q}_k\right)/c_k \qquad (2.35)$$

The parameter values are then given by (2.31) or (2.32).

2.3 Derivative magnitude estimation methods

If the interpolation process is constrained, the end derivatives are required, i.e. $\boldsymbol{C}_u(0)$ and $\boldsymbol{C}_u(1)$ for a curve, or $\boldsymbol{S}_u(0,t_l)$, $\boldsymbol{S}_u(1,t_l)$, $\boldsymbol{S}_v(s_k,0)$, $\boldsymbol{S}_v(s_k,1)$, $0 \le k \le m-2$, $0 \le l \le n-2$, for a surface. For brevity, this section will consider only curves, as the algorithms are easily applied to the rows and columns of data points on a surface.

Usually, only the end tangents directions, $\boldsymbol{t}_0$ and $\boldsymbol{t}_{m-2}$, are available as the magnitudes are not a geometric feature, being a function of the parameterisation. The derivative magnitudes, τ_0 and τ_{m-2}, need to be estimated, where:

$$\tau_0 \boldsymbol{t}_0 = \boldsymbol{C}_u(s_0)$$
$$\tau_{m-2} \boldsymbol{t}_{m-2} = \boldsymbol{C}_u(s_{m-2})$$

This section summarises the existing methods for estimating derivative magnitudes. The performance of these methods is then assessed in Chapter 4.

It is very common to estimate the magnitudes with an approximation of the arc length of the curve. As many of the parameterisations given in Section 2.2 are distance based, they have corresponding methods for estimating the arc length, and therefore the derivative magnitude. Such an approach yields a single value for both the start and end magnitudes, i.e. $\tau_0 = \tau_{m-2}$. This is not true for non-distance based approaches.

2.3.1 Total chord length

Several of the parameterisation methods utilise the chord length between adjacent data points as an approximation to the arc length of the curve between those points. The simplest method for approximating the derivative magnitudes is therefore obtained by summing the chord lengths between all adjacent data points. Hence:

$$\tau_0 = \tau_{m-2} = \sum_{k=1}^{m-2} \left\| \boldsymbol{Q}_k - \boldsymbol{Q}_{k-1} \right\| \qquad (2.36)$$

2.3.2 Total arc length

A more realistic estimate of the arc length of a curve is obtained if a circular arc is fitted between each adjacent point, as the chord between two data points will underestimate the arc length of segments with high curvature (see Figure 2.1). These arc lengths are summed to give an estimation of the total arc length. The derivative magnitudes are therefore given by:

$$\tau_0 = \tau_{m-2} = \sum_{k=0}^{m-3} l_k \qquad (2.37)$$

where l_k is the length of the *k*th circular arc, given by (2.23).

2.3.3 Farin's derivative magnitudes

Farin's [1988] piecewise cubic C^1 curve construction methods provide values for the derivative magnitudes. For Farin's simple method, the derivative magnitudes are obtained by substituting (2.29) and (2.30) into (2.27) and (2.28):

$$\tau_0 = \frac{a_0}{\Delta_0} = \frac{1.2c_0}{\Delta_0} = \frac{1.2\left\| \boldsymbol{Q}_1 - \boldsymbol{Q}_0 \right\|}{\left(s_1 - s_0 \right)} \qquad (2.38)$$

$$\tau_{m-2} = \frac{b_{m-3}}{\Delta_{m-3}} = \frac{1.2c_{m-3}}{\Delta_{m-3}} = \frac{1.2\left\| \boldsymbol{Q}_{m-2} - \boldsymbol{Q}_{m-3} \right\|}{\left(s_{m-2} - s_{m-3} \right)} \qquad (2.39)$$

where the parameter values, s_k, $0 \le k \le m-2$, are given by (2.31).

It is noted that the derivative magnitudes, in general, will be different at the two ends, and that the magnitudes are scaled up according to the parametric interval of the span.

The derivative magnitudes for Farin's more 'sophisticated' method are obtained analogously by substituting (2.33) and (2.34) into (2.27) and (2.28):

$$\tau_0 = \frac{a_0}{\Delta_0} = \frac{c_0}{\Delta_0\, max(\boldsymbol{s}_0 \cdot \boldsymbol{t}_0, 0.5)} = \frac{\|\boldsymbol{Q}_1 - \boldsymbol{Q}_0\|}{(s_1 - s_0)\, max(\boldsymbol{s}_0 \cdot \boldsymbol{t}_0, 0.5)} \tag{2.40}$$

$$\tau_{m-2} = \frac{b_{m-3}}{\Delta_{m-3}} = \frac{c_{m-3}}{\Delta_{m-3}\, max(\boldsymbol{s}_{m-3} \cdot \boldsymbol{t}_{m-2}, 0.5)}$$

$$= \frac{\|\boldsymbol{Q}_{m-2} - \boldsymbol{Q}_{m-3}\|}{(s_{m-2} - s_{m-3})\, max(\boldsymbol{s}_{m-3} \cdot \boldsymbol{t}_{m-2}, 0.5)} \tag{2.41}$$

where $\boldsymbol{s}_k$ is given by (2.35).

2.4 Knot vector generation methods

The knot vector, ***U***, needs to be generated before a NUBS curve can be interpolated. NUBS surfaces require two knot vectors, ***U***, and ***V***, but as they are both constructed analogously this section will only discuss the generation of ***U*** for a curve. Generally, an open clamped knot vector is used, as this forces the ends of the control polygon to coincide with the ends of the curve (c.f. Figure 1.7). The knot vector contains $p+1$ repeated knots at either end, and is bounded on [0,1]:

$$\boldsymbol{U} = \{u_0 = 0, \ldots, u_p = 0, u_{p+1}, \ldots, u_m, u_{m+1} = 1, \ldots, u_{m+p+1} = 1\} \tag{2.42}$$

When a cubic curve contains a single segment, the knot vector becomes $\boldsymbol{U} = \{0,0,0,0,1,1,1,1\}$, i.e. the knot vector required to reproduce a Bézier curve.

When the curve contains multiple segments, i.e. $m+1-p>1$, the internal knots, u_i, $p+1 \le i \le m$, need to be distributed appropriately. There are several existing methods for this purpose, which are reviewed here.

2.4.1 Equally spaced knots

The simplest method for spacing the internal knots is to equally space them, i.e.:

$$u_0 = \ldots = u_p = 0\,,\; u_{m+1} = \ldots = u_{m+p+1} = 1$$

$$u_{p+i} = \frac{i}{m-p+1} \qquad i = 1,\ldots,m-p \tag{2.43}$$

Piegl and Tiller [1997] warn that distributing the knots in this way can lead to a singular system of equations. The problem arises because each internal knot defines the parametric boundary between two segments, assuming none of the knots are repeated. A single segment, of degree *p*, can only satisfy a maximum of $p+1$ data constraints. If the distribution of internal knots causes a single segment to have $> p+1$ data constraints, then intuitively the interpolation can not succeed. Mathematically, this is because $[\boldsymbol{M}]$ from (2.5) will not possess an inverse.

2.4.2 Averaged knot vector

Piegl and Tiller [1997] suggest that the internal knots should reflect the distribution of the parameter values, thereby ensuring that each segment does not contain $> p+1$ data constraints. They propose an averaging technique:

$$u_0 = \ldots = u_p = 0\,,\; u_{m+1} = \ldots = u_{m+p+1} = 1$$

$$u_{p+i} = \frac{1}{p}\sum_{k=i}^{i+p-1} s_k \qquad i = 1,\ldots,m-p \tag{2.44}$$

Whilst (2.44) ensures (2.5) will always have an inverse, the resulting curve segments do not necessarily start and end at a data point.

2.4.3 Natural knots

de Boor [1978] proposes a natural knot method that sets the internal knots to coincide with the parameter values, and therefore each segment is bounded by a data point. For a constrained cubic curve interpolation, the knots are given by:

$$u_0 = \ldots = u_3 = 0 \, , \; u_{m+1} = \ldots = u_{m+4} = 1$$
$$u_{3+i} = s_i \qquad i = 1, \ldots, m-3 \qquad (2.45)$$

For an unconstrained interpolation, the first and last segments must contain 3 data points:

$$u_{3+i} = s_{i+1} \qquad i = 1, \ldots, m-3 \qquad (2.46)$$

The advantage of this method is that each span clearly corresponds to two (or three) data points, and also guarantees that (2.5) has an inverse. Park [2001] recommends using the natural method to space internal knots for all odd degree B-Splines, although warns that even degree systems using this method can become ill-conditioned. He proposes the shifting method as an alternative for even degree systems, but details are omitted here because this thesis is primarily concerned with cubic spline interpolation.

2.5 Parameter reconciliation methods

Parameterisation methods operate in a single parametric direction, and act on a string of data points, i.e. data for a curve interpolation. For surfaces, the parameterisation needs to be calculated for each row and column of data points, resulting in multiple sets of parameter values. This section gives consideration to the problem of reconciling these to provide a single parameterisation in each direction.

An unconstrained surface is interpolated with a grid of data points, $\boldsymbol{Q}_{k,l}$, $0 \le k \le m$, $0 \le l \le n$. Each point has two corresponding parameter values, $s_{k,l}$ and $t_{k,l}$. The easiest way to reconcile all the sets of the parameter values for a single parametric direction is to perform an arithmetic average [Piegl and Tiller, 1997]:

$$s_k = \frac{1}{n+1}\sum_{l=0}^{n} s_{k,l} \qquad 0 \le k \le m \tag{2.47}$$

$$t_l = \frac{1}{m+1}\sum_{k=0}^{m} t_{k,l} \qquad 0 \le l \le n \tag{2.48}$$

These equations can be modified for constrained surfaces by replacing m with $(m-2)$, and n with $(n-2)$. Instead of averaging arithmetically, geometric and harmonic averages can be used. Also, the intervals may be averaged instead of the values themselves. However, any method used to reconcile the ideal parameter values for a surface into a single set will, in general, compromise the surface quality (cf. 1.3.10). The only way to preserve quality is to ensure that the reconciled values differ as little as possible from the ideal ones by using carefully spaced data points.

Numerical evidence suggests that there is very little performance difference between these methods. All of the surface interpolations performed in this thesis use the arithmetic averaging strategy proposed by Piegl and Tiller [1997], i.e. (2.47) and (2.48).

2.6 Twist vector estimation methods

When a constrained surface is interpolated, mixed partial derivatives, also known as twist vectors, are required for each corner of the surface (cf. 1.3.11). Producing a reasonable estimation for these vectors is problematic because they do not relate to the geometry of the surface. A number of methods have been published, which are reviewed here.

2.6.1 Zero twists

The first method, suggested by Ferguson [1964], is simply to set the four twist vectors to zero. A special category of surfaces, know as translational surfaces, do have zero twists at the corners, although for most other types of surface this method will give undesirable results. Characteristic effects include 'flat spots' [Barnhill *et. al.*, 1988], and uneven parameterisation in the locality of the twist.

2.6.2 Adini twists

The Adini twist estimation method [Barnhill, *et. al.*, 1978], uses the fact that the supplied points and derivatives are sufficient to completely define the boundaries of the corner sub-patches. In turn, this information can be used to construct a bilinearly-blended Coons patch in each corner of the surface. A Coons patch is defined as:

$$\boldsymbol{C}(u,v) = \boldsymbol{R}^{c}(u,v) + \boldsymbol{R}^{d}(u,v) - \boldsymbol{R}^{cd}(u,v) \tag{2.49}$$

where $\boldsymbol{R}^{c}(u,v)$ is a ruled surface defined by two boundary curves in the u-direction, $\boldsymbol{R}^{d}(u,v)$ is a ruled surface defined by two boundary curves in the v-direction, and $\boldsymbol{R}^{cd}(u,v)$ is a bilinear patch through the 4 corner points of the local patch.

Differentiation of (2.49) with respect to u and v yields a corner twist at $\boldsymbol{C}(0,0)$ of:

$$\begin{aligned} \boldsymbol{C}_{uv}(0,0) = {} & \boldsymbol{R}^{c}(0,1) - \boldsymbol{R}^{c}(0,0) + \boldsymbol{R}^{c}(1,0) - \boldsymbol{R}^{c}(1,1) \\ & + \boldsymbol{R}_{u}^{c}(0,1) - \boldsymbol{R}_{u}^{c}(0,0) + \boldsymbol{R}_{v}^{d}(1,0) - \boldsymbol{R}_{v}^{d}(0,0) \end{aligned} \tag{2.50}$$

The Adini twist is obtained by selecting the relevant Coons patch and corner, and then scaling the sampled twist by the parametric intervals in u and v, i.e. for the corner corresponding to the point $\boldsymbol{S}(0,0)$, the Adini twist $\boldsymbol{S}_{uv[A]}(0,0)$ is:

$$\boldsymbol{S}_{uv[A]}(0,0) = \frac{\boldsymbol{C}_{uv}^{0,0}(0,0)}{\left(u_{p+1}-u_p\right)\left(v_{q+1}-v_q\right)} \tag{2.51}$$

where the knot vectors of the NUBS surface are open clamped (2.42).

2.6.3 Bessel twists

Bessel twists [Barnhill et. al., 1988] are generated by constructing four bilinear patches in the vicinity of the required twist, and taking a bilinear interpolant of their twists at the common central vertex. However, it is not always possible to know the geometry of surrounding patches and the linearity of this method implies that the error in the estimation will grow rapidly as sparser data sets are employed.

2.6.4 Selesnick twists

The normal component of a twist vector is related to the Gaussian curvature at that point. The Selesnick [1981] method calculates this component given an estimation of Gaussian curvature. The tangent component can be calculated using another method, e.g. Adini. As twist vectors have no geometric interpretation, there is no assurance that the twist vector will be principally in the normal direction, and therefore the usefulness of this method is questionable.

2.6.5 Energy method

The Energy method [Hagen and Schulze, 1987] uses a standard fairness criterion [Nowacki and Reese, 1983]:

$$\int_s \left(\kappa_{min}^2 + \kappa_{max}^2\right) \mathrm{d}s ,$$

based on the maximum and minimum curvature values at the relevant corner to evaluate the normal component of the twist; Farin and Hagen [1992] have developed this concept further. Again, only the normal component is calculated, and the usefulness of these methods is limited to those situations where an accurate estimation of the curvature is available. The twist vector has a significant effect on the curvature of the surface in region where the twist acts – therefore any curvature estimation technique based on the interpolation data alone is likely to be inaccurate.

2.6.6 Brunet twists

Brunet's method [Barnhill et. al., 1988] is fundamentally different from the others; it starts from a given patch definition, and modifies it with the aim of becoming more like the Adini surface. In Bézier form, this is implemented by taking a weighted average of the interior control points from the old surface and those resulting from using zero twists.

2.6.7 Comparison of twist vector estimation methods

Barnhill et al. [1988] considered all but one of these methods, and concluded that Adini was the best method in general for estimating twist vectors following a series of numerical tests, which produced a variety of surface curvature plots that were compared visually. However, the Adini twist estimation method naïvely assumes that a bilinearly-blended Coons patch is a reasonable approximation to a general bi-parametric patch. Where this assumption is invalid, the Adini twist estimation method is inaccurate. Farin and Hagen [1992] did propose a further method subsequent to this review, although it relies on local Gaussian curvature estimates, which may be inaccurate.

2.7 Summary

This chapter has detailed the interpolation process for curves and surfaces. A number of factors affect the quality of the interpolant, and several existing methods for controlling these have been presented. Two key factors are the parameterisation and derivative magnitude estimation methods, which are most commonly resolved using a chord length approach. Chapter 3 introduces a new family of methods to obtain the parameter values and provide derivative magnitudes, and Chapter 4 compares the performance of the existing and new methods numerically.

This chapter also summarised various methods for generating the knot vector. It was concluded that the knot vector should be distributed in conjunction with the calculated parameter values, and assigning the internal knots to be equal to the parameter values had the advantage that each pair of adjacent points then precisely corresponds to a single segment.

Options were considered for reconciling the sets of parameter values for a surface to produce a single set in each parametric direction, although it was concluded that there was very little difference in performance between the various averaging methods. It was noted that any method that changed the parameter values from the ideal values would compromise the surface quality. To avoid this compromise, the points themselves need to be spaced appropriately, which is the subject of Chapter 5.

Several methods for estimating the twist vectors were outlined. It was reported that the Adini twist, which is based on a bilinearly-blended Coons patch, is the best method in general, although the formulation suggests that certain geometries may not yield acceptable results. Chapter 6 returns to the subject of twist vector estimation.

Chapter 3

Orthogonal curve construction

When constructing a piecewise curve, it is desirable that the parameter value varies linearly with arc length [Brodlie, 1980], [de Boor, 2001]. Many of the parameterisation and derivative magnitude estimation algorithms presented in Chapter 2 strive to obtain this property by approximating the distance along the curve between data points. Ball [2004] shows that an equivalent formulation of this property is that the first and second parametric derivatives must be orthogonal. Ball proves that, in general, this condition will not be satisfied at all points on the parametric segment simultaneously, and proposes a family of new C^1 piecewise curve construction methods that force the condition to be satisfied at various parametric locations. The piecewise construction allows the parameter values and the derivative magnitudes to be obtained, which can then be used to construct a C^2 NUBS spline by interpolation.

The orthogonal parameterisation and derivative magnitude estimation algorithms are investigated analytically to establish geometric conditions that can cause instability, degeneracy, or over-sensitivity to changes in input data. The performance of the algorithms is tested numerically in Chapter 4.

3.1 Orthogonal methods

The majority of curve construction methods strive to distribute the parameter values such that the interpolant's parameter will vary linearly with arc length. The general strategy used is to assign parameter values in proportion to the arc lengths of the curve segments, which are usually estimated using a distance-based scheme such as the chord length between data points. Ball [2004] shows that the parameter value of a cubic spline varies linearly with arc length when the first and second derivatives are orthogonal, i.e.:

$$\frac{d^2\boldsymbol{C}}{dt^2}\cdot\frac{d\boldsymbol{C}}{dt}=0 \tag{3.1}$$

where *t* is the local parameter of a curve segment.

The advantage of formulating the construction process in terms of the orthogonality conditions is that they are parametric expressions that can be satisfied exactly, whereas the distance-based parameterisations need to estimate the arc length using heuristic methods.

To create a piecewise C^1 construction, the data points, $\boldsymbol{Q}_k$, and tangent directions, $\boldsymbol{t}_k=\boldsymbol{C}_u(s_k)/\left\|\boldsymbol{C}_u(s_k)\right\|$, are required, $0\le k\le m-2$. The derivative magnitudes for each segment, $a_k=\Delta_k\left\|\boldsymbol{C}_u(s_k)\right\|$ and $b_k=\Delta_k\left\|\boldsymbol{C}_u(s_{k+1})\right\|$, $\Delta_k=(s_{k+1}-s_k)$, must be determined to satisfy the orthogonality conditions. The parameter values, s_k, $0\le k\le m-2$, can then be solved uniquely using the first order parametric continuity condition:

$$\frac{a_k}{\Delta_k}=\frac{b_{k-1}}{\Delta_{k-1}} \qquad 1\le k<m-2 \tag{3.2}$$

assuming $s_0=0$ and $s_{m-2}=1$. Rearranging (3.2) allows the parameter values to be obtained more efficiently by assuming $s_0=0$ and $s_1=1$, then calculating s_{k+1}:

$$s_{k+1} = \frac{a_k\left(s_k - s_{k-1}\right)}{b_{k-1}} + s_k \qquad 1 \le k < m-2 \tag{3.3}$$

The parameter values will be in the range $0 \le s_k \le s_{m-2}$, and therefore need to be scaled down by a factor of s_{m-2} to be in the range [0,1].

The derivative magnitudes at the start and end of the piecewise curve are obtained by

$$\tau_0 = \left\|\boldsymbol{C}_u\left(s_0\right)\right\| = \frac{a_0}{\Delta_0} \tag{3.4}$$

$$\tau_{m-2} = \left\|\boldsymbol{C}_u\left(s_{m-2}\right)\right\| = \frac{b_{m-3}}{\Delta_{m-3}} \tag{3.5}$$

A C^2 NUBS spline is constructed by interpolation using the same parameter values and end derivative magnitudes; however, the internal tangent directions and magnitudes will vary from those of the C^1 construction, and therefore the orthogonality conditions will be disturbed. It is possible at this stage to discard the derivative information and perform an unconstrained interpolation; in this case, $m-2$ must be replaced with m in this chapter, as it is assumed that the interpolation is constrained.

Each segment of the C^1 curve can be represented using the Hermite equation form:

$$\begin{aligned}\boldsymbol{C}(u) = &\left(1-3t^2+2t^3\right)\boldsymbol{C}\left(s_k\right) + \left(3t^2-2t^3\right)\boldsymbol{C}\left(s_{k+1}\right) \\ &+ \left(t-2t^2+t^3\right)\Delta_k\boldsymbol{C}_u\left(s_k\right) + \left(-t^2+t^3\right)\Delta_k\boldsymbol{C}_u\left(s_{k+1}\right)\end{aligned} \qquad 0 \le k < m-2 \tag{3.6}$$

where $t = \left(u - s_k\right)/\Delta_k$.

Substituting the first and second derivatives of (3.6) into (3.1) yields the orthogonality condition [Ball, 2004]:

$$\begin{aligned}&\left(36t-108t^2+72t^3\right)c_k^2+\left(-4+22t-36t^2+18t^3\right)a_k^2+\left(4t-18t^2+18t^3\right)b_k^2\\&\quad+\left(6-60t+126t^2-72t^3\right)c_k a_k \boldsymbol{s}_k\cdot\boldsymbol{t}_k+\left(-24t+90t^2-72t^3\right)c_k b_k \boldsymbol{s}_k\cdot\boldsymbol{t}_{k+1}\\&\quad+\left(-2+22t-54t^2+36t^3\right)a_k b_k \boldsymbol{t}_k\cdot\boldsymbol{t}_{k+1}=0\end{aligned}\tag{3.7}$$

where c_k is the chord length, and $\boldsymbol{s}_k$ the unit direction of the chord:

$$c_k=\left\|\boldsymbol{Q}_{k+1}-\boldsymbol{Q}_k\right\| \qquad 0\le k<m-2 \tag{3.8}$$

$$\boldsymbol{s}_k=\left(\boldsymbol{Q}_{k+1}-\boldsymbol{Q}_k\right)/c_k \qquad 0\le k<m-2 \tag{3.9}$$

and $\boldsymbol{t}_k$, $\boldsymbol{t}_{k+1}$ are the tangent directions. These are illustrated in Figure 3.1:

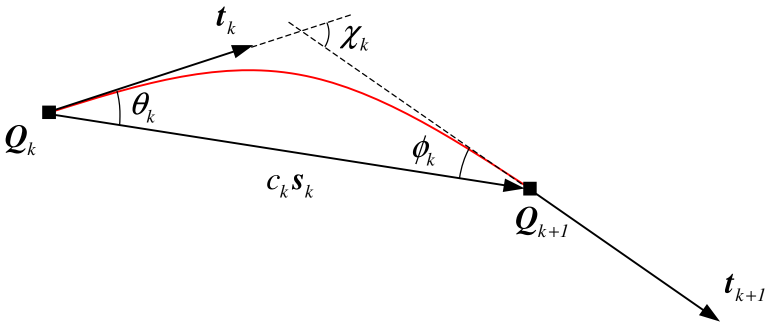


Figure 3.1 – Notation for C^1 cubic spline

The orthogonality condition (3.7) is cubic in *t*, and therefore there are, in general, at most three points on a non-rational parametric cubic segment where it can be satisfied. Ball's formulation controls the parametric location of two of these points, specifically at t and $(1-t)$ simultaneously. Three different algorithms are proposed that correspond to distinct locations of t; these are given by [Ball, 2004]:

$$W=9t(1-t) \qquad W=0,1,2 \tag{3.10}$$

Ball [2004] reformulates the orthogonality condition (3.7) to correspond with the parametric locations t and $(1-t)$, and substitutes (3.10) to eliminate t:

$$2Wc_k^2+(-2+W)a_k^2+(3-3W)c_ka_k\boldsymbol{s}_k\cdot\boldsymbol{t}_k+(-W)c_kb_k\boldsymbol{s}_k\cdot\boldsymbol{t}_{k+1}+(-1+W)a_kb_k\boldsymbol{t}_k\cdot\boldsymbol{t}_{k+1}=0 \tag{3.11}$$

$$2Wc_k^2+(-2+W)b_k^2+(-W)c_ka_k\boldsymbol{s}_k\cdot\boldsymbol{t}_k+(3-3W)c_kb_k\boldsymbol{s}_k\cdot\boldsymbol{t}_{k+1}+(-1+W)a_kb_k\boldsymbol{t}_k\cdot\boldsymbol{t}_{k+1}=0 \tag{3.12}$$

Given the geometric data c_k, $\boldsymbol{s}_k$, $\boldsymbol{t}_k$, $\boldsymbol{t}_{k+1}$, and the selected value of *W*, (3.11) and (3.12) can be solved for a_k and b_k, accepting only those solutions with positive values of a_k and b_k.

3.1.1 Orthogonal construction, W=0

When $W=0$, the orthogonality condition is satisfied at both ends of the segment. Substituting $W=0$ into (3.11) and (3.12), and solving the resulting linear equations for $a_k, b_k > 0$ yields [Ball, 2004]:

$$a_k=3c_k\left(\frac{2\boldsymbol{s}_k\cdot\boldsymbol{t}_k-\boldsymbol{s}_k\cdot\boldsymbol{t}_{k+1}(\boldsymbol{t}_k\cdot\boldsymbol{t}_{k+1})}{4-(\boldsymbol{t}_k\cdot\boldsymbol{t}_{k+1})^2}\right) \tag{3.13}$$

$$b_k=3c_k\left(\frac{2\boldsymbol{s}_k\cdot\boldsymbol{t}_{k+1}-\boldsymbol{s}_k\cdot\boldsymbol{t}_k(\boldsymbol{t}_k\cdot\boldsymbol{t}_{k+1})}{4-(\boldsymbol{t}_k\cdot\boldsymbol{t}_{k+1})^2}\right) \tag{3.14}$$

Certain geometric configurations of $\boldsymbol{s}_k\cdot\boldsymbol{t}_k$ and $\boldsymbol{s}_k\cdot\boldsymbol{t}_{k+1}$ lead to negative or improbably small values of a_k and b_k, so it is assumed $\boldsymbol{s}_k\cdot\boldsymbol{t}_k, \boldsymbol{s}_k\cdot\boldsymbol{t}_{k+1}\geq 0.5$, i.e. $\theta_k,\phi_k\leq 60°$ (see Figure 3.1.), in keeping with Farin [1988]. The parameter values are given by (3.2).

Ball [2004] proposes two variations to (3.13) and (3.14); the first is motivated by the convex planar configuration. The dot products $\boldsymbol{s}_k\cdot\boldsymbol{t}_k$ and $\boldsymbol{s}_k\cdot\boldsymbol{t}_{k+1}$ are replaced by the cosines $\cos(\theta_k)$ and $\cos(\phi_k)$ respectively, and $\boldsymbol{t}_k\cdot\boldsymbol{t}_{k+1}$ is replaced with $\cos(\theta_k+\phi_k)$, which is only equivalent to $\cos(\chi_k)$ when $\boldsymbol{t}_k$ and $\boldsymbol{t}_{k+1}$ are coplanar (see Figure 3.1.) This yields:

$$a_k = 3c_k \frac{2\cos\theta_k^* - \cos\phi_k^* \cos\left(\theta_k^* + \phi_k^*\right)}{4 - \cos^2\left(\theta_k^* + \phi_k^*\right)} \tag{3.15}$$

$$b_k = 3c_k \frac{2\cos\phi_k^* - \cos\theta_k^* \cos\left(\theta_k^* + \phi_k^*\right)}{4 - \cos^2\left(\theta_k^* + \phi_k^*\right)} \tag{3.16}$$

where $\theta_k^* = \min\left(\theta_k, 60°\right)$ and $\phi_k^* = \min\left(\phi_k, 60°\right)$.

The second variation is motivated by the circular configuration, where $\theta_k = \phi_k$; simplifying (3.15) and (3.16) with this assumption yields:

$$a_k = 3c_k \frac{\cos\theta_k^*}{2 + \cos 2\theta_k^*} \tag{3.17}$$

$$b_k = 3c_k \frac{\cos\phi_k^*}{2 + \cos 2\phi_k^*} \tag{3.18}$$

3.1.2 Circle orthogonal parameterisation

Cripps and Ball [2003] propose a parameterisation that is optimal for circular arcs, and is based on the orthogonal $W = 0$ construction; however, the method does not require tangent directions as they can be constructed from the data points. The parameterisation is able to construct a C^1 piecewise curve that has orthogonality between first and second derivatives at the knots.

Given $\boldsymbol{Q}_k$, $0 \le k \le m-2$, let $\boldsymbol{d}_k = \boldsymbol{Q}_k - \boldsymbol{Q}_{k-1}$, $1 \le k \le m-2$, and $\boldsymbol{e}_i = \boldsymbol{Q}_{k+1} - \boldsymbol{Q}_{k-1} = \boldsymbol{d}_k + \boldsymbol{d}_{k+1}$, $1 \le k \le m-3$, as illustrated in Figure 3.2, then the parameterisation is given by Cripps and Ball [2003]:

$$s_0 = 0, \; s_1 = 1$$

$$s_{k+1} = s_k + \left(s_k - s_{k-1}\right)\left[\frac{\left[\left(\boldsymbol{d}_{k+1} \cdot \boldsymbol{e}_k\right)^2 + \varphi_k\right]\left(\boldsymbol{d}_k \cdot \boldsymbol{e}_k\right)}{\left[\left(\boldsymbol{d}_k \cdot \boldsymbol{e}_k\right)^2 + \psi_k\right]\left(\boldsymbol{d}_{k+1} \cdot \boldsymbol{e}_k\right)}\right] \quad 1 \le k \le m-3 \tag{3.19}$$

where φ_k and ψ_k are scalars determined to ensure the condition (3.1) is satisfied at the knots. φ_k and ψ_k are given by Cripps and Lockyer [2005]:

$$\varphi_k = \frac{1}{2}\left\|\boldsymbol{d}_{k+1}\right\|^2 \left\|\boldsymbol{e}_k\right\|^2 \tag{3.20}$$

$$\psi_k = \frac{1}{2}\left\|\boldsymbol{d}_k\right\|^2 \left\|\boldsymbol{e}_k\right\|^2 \tag{3.21}$$

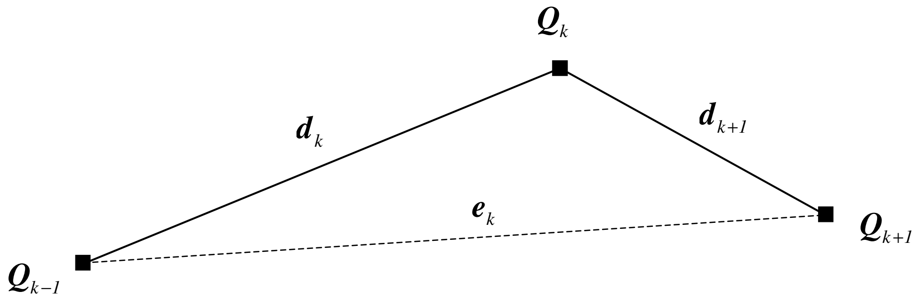


Figure 3.2 - Circle orthogonal construction

The parameter values given by (3.19) will be in the range $0 \le s_k \le s_{m-2}$, and therefore need to be scaled down by a factor of s_{m-2} to be in the range [0,1].

3.1.3 Orthogonal construction, W=1

When $W = 1$ is substituted into (3.11) and (3.12), the orthogonality condition is satisfied at $t = \left(3 \pm \sqrt{5}\right)/6$ and gives [Ball, 2004]:

$$2c_k^2 - a_k^2 - c_k b_k \boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1} = 0 \tag{3.22}$$

$$2c_k^2 - b_k^2 - c_k a_k \boldsymbol{s}_k \cdot \boldsymbol{t}_k = 0 \tag{3.23}$$

This value of W is special because it ensures the solution is independent of the angle between the two end tangents, χ_k. Eliminating b_k from (3.22) and (3.23) gives a quartic function:

$$f\left(a_k\right) = a_k^4 - 4c_k^2 a_k^2 + c_k^3 a_k \left(\boldsymbol{s}_k \cdot \boldsymbol{t}_k\right)\left(\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1}\right)^2 + 4c_k^4 - 2c_k^4\left(\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1}\right)^2 = 0 \tag{3.24}$$

Ball [2004] shows that, if $\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1} > 0$, then $0 < a_k < \sqrt{2}c_k$. The required root of $f(a_k) = 0$ can therefore be found numerically; the corresponding value of $b_k > 0$ can be obtained from (3.24). The parameter values are given by (3.2).

3.1.4 Orthogonal construction, W=2

Substituting $W = 2$ into (3.11) and (3.12) causes the orthogonality condition to be satisfied at $t = 1/3$ and $t = 2/3$, and gives [Ball, 2004]:

$$4c_k^2 - 3c_k a_k \boldsymbol{s}_k \cdot \boldsymbol{t}_k - 2c_k b_k \boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1} + a_k b_k \boldsymbol{t}_k \cdot \boldsymbol{t}_{k+1} = 0 \tag{3.25}$$

$$4c_k^2 - 2c_k a_k \boldsymbol{s}_k \cdot \boldsymbol{t}_k - 3c_k b_k \boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1} + a_k b_k \boldsymbol{t}_k \cdot \boldsymbol{t}_{k+1} = 0 \tag{3.26}$$

which have solutions:

$$a_k = \frac{8c_k(\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1})}{5(\boldsymbol{s}_k \cdot \boldsymbol{t}_k)(\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1}) + \sqrt{25(\boldsymbol{s}_k \cdot \boldsymbol{t}_k)^2(\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1})^2 - 16(\boldsymbol{s}_k \cdot \boldsymbol{t}_k)(\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1})(\boldsymbol{t}_k \cdot \boldsymbol{t}_{k+1})}} \tag{3.27}$$

$$b_k = \frac{8c_k(\boldsymbol{s}_k \cdot \boldsymbol{t}_k)}{5(\boldsymbol{s}_k \cdot \boldsymbol{t}_k)(\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1}) + \sqrt{25(\boldsymbol{s}_k \cdot \boldsymbol{t}_k)^2(\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1})^2 - 16(\boldsymbol{s}_k \cdot \boldsymbol{t}_k)(\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1})(\boldsymbol{t}_k \cdot \boldsymbol{t}_{k+1})}} \tag{3.28}$$

If $\boldsymbol{t}_k = \boldsymbol{t}_{k+1}$, then there is no real solution if $0 < \boldsymbol{s}_k \cdot \boldsymbol{t}_k = \boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1} < 0.8$. Ball [2004] suggests three possible solutions:

a) Restrict applications such that $\boldsymbol{s}_k \cdot \boldsymbol{t}_k, \boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1} \geq 0.8$, i.e. $\theta, \phi < 36.9°$ (3 s.f.), see Figure 3.1.

b) Replace $\boldsymbol{t}_k \cdot \boldsymbol{t}_{k+1}$ by its average value $(\boldsymbol{s}_k \cdot \boldsymbol{t}_k)(\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1})$. This then gives the same formulae as (2.33) and (2.34), and extends the applicability to $\boldsymbol{s}_k \cdot \boldsymbol{t}_k, \boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1} \geq 0.5$.

c) Replace $\boldsymbol{t}_k \cdot \boldsymbol{t}_{k+1}$ by $\cos(\theta_k + \phi_k)$, motivated by the convex planar configuration. This gives:

$$a_k = \frac{8c_k}{\cos\theta_k^* \left(5 + \sqrt{9 + 16\tan\theta_k^* \tan\phi_k^*}\right)} \tag{3.29}$$

$$b_k = \frac{8c_k}{\cos\phi_k^* \left(5 + \sqrt{9 + 16\tan\theta_k^* \tan\phi_k^*}\right)} \tag{3.30}$$

where $\theta_k^* = \min(\theta_k, 60°)$ and $\phi_k^* = \min(\phi_k, 60°)$.

3.1.5 Derivative magnitude estimation

Each of the orthogonal construction methods, $W = 0,1,2$, can be used to generate the parameter values and derivative magnitudes such that a piecewise cubic C^1 curve could be constructed, having orthogonal derivatives at the parameter values specified by (3.10). However, when the points, end tangents directions, end tangent magnitudes and the parameter values are used to interpolate a cubic NUBS curve (i.e. C^2 continuity across segments), the orthogonality conditions are, in general, no longer satisfied. Ball [2005] shows that it is possible to change the end tangent magnitudes, τ_0 and τ_{m-2}, to force orthogonality at either end of the interpolated spline.

Given $(m-1)$ data points, $\boldsymbol{Q}_0 \ldots \boldsymbol{Q}_{m-2}$, end tangent directions, $\boldsymbol{t}_0$ and $\boldsymbol{t}_{m-2}$, and the knot vector, the cubic NUBS interpolant is uniquely defined by the two end tangent magnitudes, $\tau_0 = \varphi c$ and $\tau_{m-2} = \psi c$, where c is the accumulated chord length given by (2.36). The cubic spline, $\boldsymbol{C}(\varphi,\psi)$, $0 \le \varphi, \psi \le 1$, can be expressed as the sum of three other cubic splines [Ball, 2005]:

$$\boldsymbol{C}(\varphi,\psi) = (\varphi + \psi - 1)\boldsymbol{C}(1,1) + (1-\psi)\boldsymbol{C}(1,0) + (1-\varphi)\boldsymbol{C}(0,1) \tag{3.31}$$

Let $\boldsymbol{C}_0''(\varphi,\psi)$ and $\boldsymbol{C}_{m-2}''(\varphi,\psi)$ denote the respective end second derivatives, and

$$\boldsymbol{D}_0''(\varphi,\psi)=\boldsymbol{C}_0''(\varphi,\psi)\cdot\boldsymbol{t}_0 \tag{3.32}$$

$$\boldsymbol{D}_{m-2}''(\varphi,\psi)=\boldsymbol{C}_{m-2}''(\varphi,\psi)\cdot\boldsymbol{t}_{m-2} \tag{3.33}$$

To force orthogonality at the ends of the spline, values of φ,ψ need to be found such that:

$$\boldsymbol{D}_0''(\varphi,\psi)=0 \tag{3.34}$$

and $$\boldsymbol{D}_{m-2}''(\varphi,\psi)=0 \tag{3.35}$$

Ball [2005] formulates a pair of linear equations in $(\varphi-1),(\psi-1)$:

$$\boldsymbol{D}_0''(1,1)+(\varphi-1)\left[\boldsymbol{D}_0''(1,1)-\boldsymbol{D}_0''(0,1)\right]+(\psi-1)\left[\boldsymbol{D}_0''(1,1)-\boldsymbol{D}_0''(1,0)\right]=0 \tag{3.36}$$

$$\boldsymbol{D}_{m-2}''(1,1)+(\varphi-1)\left[\boldsymbol{D}_{m-2}''(1,1)-\boldsymbol{D}_{m-2}''(0,1)\right]+(\psi-1)\left[\boldsymbol{D}_{m-2}''(1,1)-\boldsymbol{D}_{m-2}''(1,0)\right]=0 \tag{3.37}$$

Hence the two end tangent magnitudes, $\tau_0=\varphi c$ and $\tau_{m-2}=\psi c$, can be found.

3.2 Analytical evaluation of algorithms

The previous section introduced the orthogonal construction methods corresponding to $W=0,1,2$, which strive to establish orthogonality between the first and second derivatives at different parametric locations; however, no indication was given to suggest which of the methods presented are of greatest practical interest. This section aims to provide insight into the behaviour of these methods by critically evaluating them in terms of their stability, degeneracy, and sensitivity. The numerical performance of the algorithms is tested in Chapter 4 using a range of data configurations; recommendations on the most useful methods for general interpolation are then made, drawing on the information provided in both chapters.

In this section, it is assumed that the input data is not too extreme, i.e. that the tangents form an angle no greater than $90°$ with the chord, i.e. $0° \leq \theta_k, \phi_k \leq 90°$, and that no two consecutive data points are the same, i.e. $\boldsymbol{Q}_k \neq \boldsymbol{Q}_{k+1}$, $0 \leq k < m-2$.

3.2.1 Stability of algorithms

Instability may arise in an algorithm under certain geometric conditions, causing the result to be undefined. Typically, this occurs when the algorithm requires division by zero. The stability of an algorithm does not give any indication of how well an algorithm will perform given a specific data set, rather how robust it is numerically. This is evident when considering the equally spaced parameterisation method – the output is completely stable as it is not a function of the input data, but in general produces a poor result. The output of an algorithm is likely to be unrealistic, albeit valid, when the geometry is close to that which causes instability.

Table 3.1 summarises all of the conditions that cause instability in the orthogonal methods, and the equation numbers from which they are derived. The table also states the geometric configurations that result in the instability condition being satisfied. The orthogonal methods operate by calculating the derivative magnitudes for each segment; instability generally arises when a denominator evaluates to zero when calculating these magnitudes, or when the numerator evaluates to zero causing the denominator of another equation to become zero. Following the table are a few illustrations of the geometric configurations that cause failure.

Method	**Equations**	**Condition for instability**	**Geometric configuration**
$W = 0$	(3.14), (3.3)	$2\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1} - (\boldsymbol{s}_k \cdot \boldsymbol{t}_k)(\boldsymbol{t}_k \cdot \boldsymbol{t}_{k+1}) = 0$	i) $\phi_k = 90°$ and $\theta_k = 0, 90°$ (Figure 3.3) ii) Inflecting condition $\phi_k = \tan^{-1}\left(\dfrac{2-\cos^2\theta_k}{\sin\theta_k \cos\theta_k}\right)$ (Figure 3.4)
$W = 0\ RP$	(3.16), (3.3)	If θ_k, ϕ_k not restricted to $60°$: $2\cos\phi_k - \cos\theta_k \cos(\theta_k + \phi_k) = 0$	i) $\phi_k = 90°$ and $\theta_k = 0, 90°$ (Figure 3.3) ii) Inflecting condition $\phi_k = \tan^{-1}\left(\dfrac{2-\cos^2\theta_k}{\sin\theta_k \cos\theta_k}\right)$ (Figure 3.4)
$W = 0\ RC$	(3.18), (3.3)	If θ_k, ϕ_k not restricted to $60°$: $\cos\phi_k = 0$	$\phi_k = 90°$
CO	(3.19)	$(\boldsymbol{d}_{k+1} \cdot \boldsymbol{e}_k) = 0$	$\angle(\boldsymbol{d}_{k+1}, \boldsymbol{e}_k) = 90°$ (refer to Figure 3.2)
$W = 1$	n/a	Never	n/a
$W = 2a$	(3.27), (3.28)	No instability, however (3.27) and (3.28) may be undefined (complex solution) when $0 < \boldsymbol{s}_k \cdot \boldsymbol{t}_k, \boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1} < 0.8$	Possibly undefined when $\theta_k, \phi_k > 36.9°$
$W = 2b$	(2.33)	If θ_k, ϕ_k not restricted to $60°$: $\boldsymbol{s}_k \cdot \boldsymbol{t}_k, \boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1} = 0$	$\theta_k, \phi_k = 90°$
$W = 2c$	(3.29), (3.30)	If θ_k, ϕ_k not restricted to $60°$: $\cos\theta_k, \cos\phi_k = 0$	$\theta_k, \phi_k = 90°$

Table 3.1 – Conditions that cause instability for the orthogonal parameterisations

Figures 3.3 and 3.4 demonstrate failure conditions for $W = 0$ and $W = 0\ RP$, and were interpolated using the $W = 1$ method for illustrative purposes. These configurations cause the numerators of (3.14) and (3.16), for the $W = 0$ and $W = 0\ RP$ methods respectively, to become zero. This results in division by zero when the parameter values are calculated using (3.3).

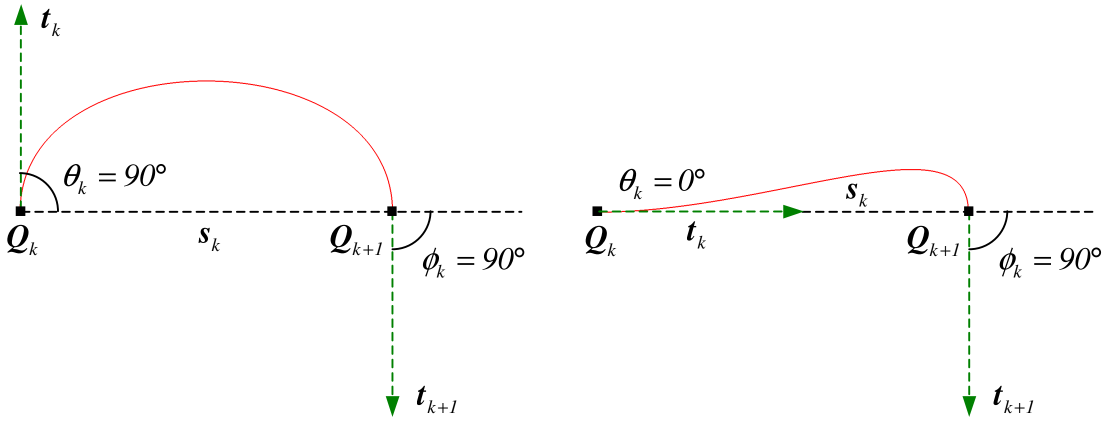


Figure 3.3 – Conditions for instability with $W = 0$ orthogonal parameterisation

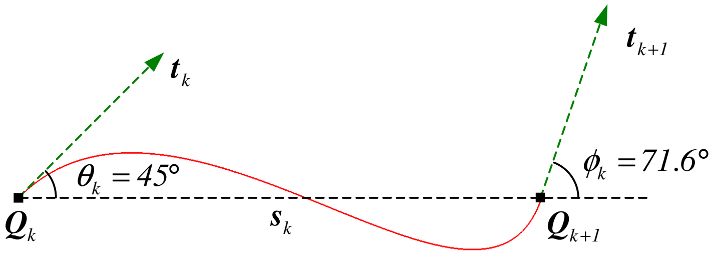


Figure 3.4 – Inflecting condition for instability with $W = 0$ orthogonal parameterisation

Figure 3.4 illustrates just one of an infinite number of inflecting configurations that cause instability for the $W = 0$ and $W = 0\ RP$ methods. When $\boldsymbol{t}_k$ and $\boldsymbol{t}_{k+1}$ are coplanar, the angle between the tangents is $\chi_k = \theta_k - \phi_k$. For every value of $0 \le \theta_k \le 90°$, there is a corresponding value of ϕ_k that will cause the $W = 0$ and $W = 0\ RP$ parameterisations to fail; the exact value is given by:

$$2\cos\phi_k = \cos\theta_k \cos\gamma_k = \cos\theta_k \cos\left(\theta_k - \phi_k\right) \qquad \text{(From the numerator of (3.16))}$$

$$\Rightarrow \qquad 2\cos\phi_k = \cos\theta_k \left(\cos\theta_k \cos\phi_k + \sin\theta_k \sin\phi_k\right)$$

$$\Rightarrow \qquad 2 = \cos^2\theta_k + \sin\theta_k \cos\theta_k \tan\phi_k$$

$$\Rightarrow \qquad \phi_k = \tan^{-1}\left(\frac{2-\cos^2\theta_k}{\sin\theta_k \cos\theta_k}\right)$$

It is evident that all of the methods, with the exception of $W = 1$, become unstable when interpolating certain geometric configurations. Generally, the angles between tangent and chord must be very large (e.g. $> 60°$) before any problems are encountered. Many of the methods have numerical artifices to prevent angles in excess of $60°$ being used, thereby averting difficulties, but this approach disturbs the orthogonality conditions for the C^1 piecewise construction.

3.2.2 Degenerate performance

Degenerate performance occurs when the output of an algorithm is invalid, for example when a parameterisation algorithm produces repeated parameter values for non-coincident points, or a derivative magnitude estimation algorithm yields a zero magnitude. The output of an algorithm is likely to be unrealistic, albeit valid, when the geometric conditions are close to that which causes degenerate behaviour.

Table 3.2 summarises all of the conditions that cause degeneracy in the orthogonal methods, and the equation numbers from which they are derived. The table also states the geometric configurations that result in the degeneracy condition being satisfied. The orthogonal methods operate by calculating the derivative magnitudes for each segment; degeneracy generally arises when a numerator evaluates to zero when calculating these magnitudes, which causes the parameter values for consecutive data points to become identical. Following the table is an illustration of a geometric configuration that causes failure.

Method	**Equations**	**Condition for degeneracy**	**Geometric configuration**
$W=0$	(3.13), (3.3)	$2\boldsymbol{s}_k \cdot \boldsymbol{t}_k - (\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1})(\boldsymbol{t}_k \cdot \boldsymbol{t}_{k+1}) = 0$	i) $\theta_k = 90°$ and $\phi_k = 0, 90°$ ii) Inflecting condition $\theta_k = \tan^{-1}\left(\frac{2-\cos^2\phi_k}{\sin\phi_k \cos\phi_k}\right)$ (Figure 3.5)
$W=0\ RP$	(3.15), (3.3)	If θ_k, ϕ_k not restricted to $60°$: $2\cos\theta_k - \cos\phi_k \cos(\theta_k + \phi_k) = 0$	i) $\theta_k = 90°$ and $\phi_k = 0, 90°$ ii) Inflecting condition $\theta_k = \tan^{-1}\left(\frac{2-\cos^2\phi_k}{\sin\phi_k \cos\phi_k}\right)$ (Figure 3.5)
$W=0\ RC$	(3.17), (3.3)	If θ_k, ϕ_k not restricted to $60°$: $\cos\theta_k = 0$	$\theta_k = 90°$
CO	(3.19)	$(\boldsymbol{d}_k \cdot \boldsymbol{e}_k) = 0$	$\angle(\boldsymbol{d}_k, \boldsymbol{e}_k) = 90°$ (refer to Figure 3.2)
$W=1$	n/a	Never	n/a
$W=2a$	(3.27), (3.28)	No instability, however (3.27) and (3.28) may be undefined (complex solution) when $0 < \boldsymbol{s}_k \cdot \boldsymbol{t}_k, \boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1} < 0.8$	Possibly undefined when $\theta_k, \phi_k > 36.9°$
$W=2b$	n/a	Never	n/a
$W=2c$	n/a	Never	n/a

Table 3.2 – Conditions that cause degeneracy for the orthogonal parameterisations

Figure 3.5 demonstrates the failure condition for $W=0$ and $W=0\ RP$ when interpolating an inflecting configuration; the red curve illustrates the degenerate curve interpolated using $W=0$, and the blue curve illustrates a non-degenerate curve interpolated using $W=1$. The geometric configuration causes the numerators of (3.13) and (3.15), for the $W=0$ and $W=0\ RP$ methods respectively, to become zero. When the results are used to calculate the parameter values using (3.3), consecutive values are identical, i.e. the parametric interval for the segment will be zero. Also, the starting derivative magnitude for the segment will be zero, allowing the tangent at the start of the interpolated curve to differ from the specified tangent.

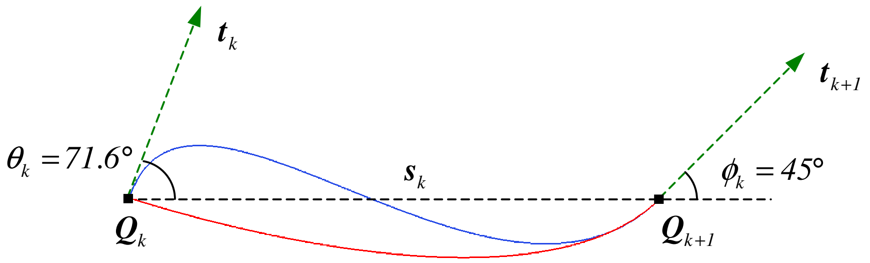


Figure 3.5 – Degenerate performance with $W=0$ (red), compared with $W=1$ (blue)

Again, degeneracy only occurs with large angles between the tangent and chord (e.g. $>60°$), and a number of the methods have numerical artifices to limit the angles.

3.2.3 Sensitivity to change

The sensitivity of an algorithm is an indication of how large the change in output is following a small change in input. The majority of the orthogonal parameterisations require tangent directions at each of the data points, and operate by varying the magnitudes for each span, i.e. a_k and b_k. As these magnitudes are calculated analogously, only a_k is considered. This subsection investigates the sensitivity of a_k / c_k following a change in the configuration of the tangents (i.e. θ_k and ϕ_k) for each of the algorithms, $W=0,1,2$.

W=0 orthogonal parameterisation

(3.13) can be reformulated in terms of θ_k, ϕ_k and χ_k (Figure 3.1):

$$a_k = 3c_k\left(\frac{2\cos\theta_k - \cos\phi_k\cos\chi_k}{4-\cos^2\chi_k}\right) \tag{3.38}$$

When $\boldsymbol{t}_k$ and $\boldsymbol{t}_{k+1}$ are coplanar, and configured such that the curve is inflecting (e.g. Figure 3.4), then the angle $\chi_k = \theta_k - \phi_k$. For $0° \le \theta_k, \phi_k \le 90°$, the relationship with a_k/c_k is illustrated in Figure 3.6:

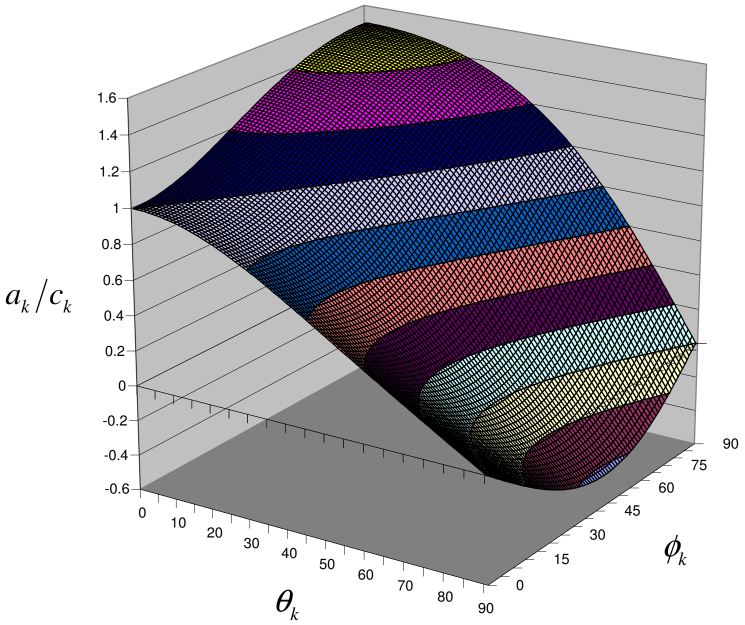


Figure 3.6 – Variation of a_k/c_k with θ_k and ϕ_k for $W = 0$ (inflecting)

It is evident that, for small θ_k, ϕ_k, slight changes in angle do not produce a large change in a_k; when $\theta_k = \phi_k = 0°$, a change in angle $\Delta\theta_k = 1°$ produces a 0.0254% change in a_k/c_k. However, the gradient of a_k/c_k increases with larger θ_k, ϕ_k, indicating greater sensitivity: when $\theta_k = 70°, \phi_k = 45°$, a change in angle $\Delta\theta_k = 1°$ produces a 178% change in a_k/c_k. The percentage change is particularly large with this configuration as the absolute value of a_k/c_k is very close to zero, and therefore the parameterisation is on the verge of degenerating.

For non-inflecting planar configurations, the angle between tangents is given by $\chi_k = \theta_k + \phi_k$. This is equivalent to the formula when it has been relaxed using the convex planar configuration (3.15) without capping the angles to 60°. The relationship is illustrated in Figure 3.7:

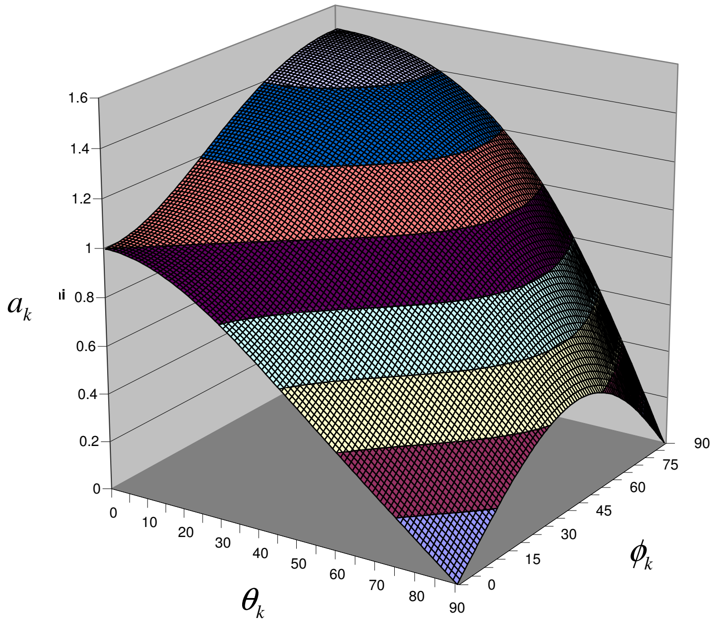
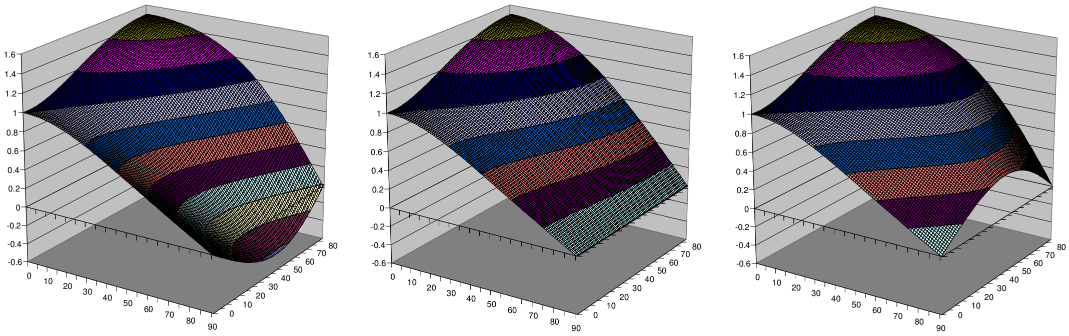


Figure 3.7 – Variation of a_k/c_k with θ_k and ϕ_k for $W = 0$ (non-inflecting)

In this configuration, the gradient of a_k again increases rapidly with larger values of θ_k, ϕ_k reaffirming that the parameterisation is fairly sensitive to changes in input. When $\theta_k = \phi_k = 0°$, a change in angle $\Delta\theta_k = 1°$ produces a 0.0254% change in a_k/c_k, whereas at $\theta_k = 90°, \phi_k = 45°$ it produces a 4.70% change.

All non-planar configurations can be represented by rotating $\boldsymbol{t}_{k+1}$ about an axis through $\boldsymbol{Q}_k$, $\boldsymbol{Q}_{k+1}$, by an angle of $180°$. As the angle varies between 0 and $180°$, the resulting relationship is seen to vary between the two planar configurations, Figure 3.8.



Rotated *0*° (Inflecting planar configuration) → Rotated *90*° → Rotated *180*° (Non-inflecting planar configuration)

Figure 3.8 – The effect on a_k/c_k when rotating $\boldsymbol{t}_{k+1}$ about the chord between up to *180*°

W=0 orthogonal parameterisation – relaxed circular configuration

The relationship between a_k/c_k and θ_k, ϕ_k for the $W=0$ relaxed circular configuration (3.17) is illustrated in Figure 3.9; it is noted that this relationship is identical for both the inflecting and non-inflecting cases. The relationship is independent of the angle between tangents, therefore Figure 3.9 is the same for non-planar configurations.

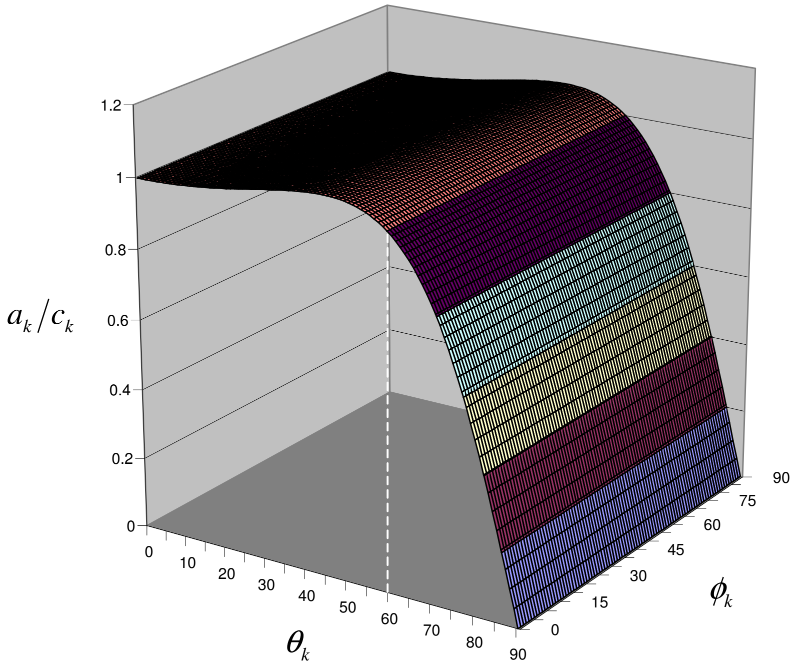


Figure 3.9 – Variation of a_k/c_k with θ_k and ϕ_k for $W=0$ (relaxed circular)

The parameterisation is relatively insensitive to changes in tangent angle for $\theta_k < 60°$, but with larger angles, a_k/c_k becomes highly sensitive. When $\theta_k = 0°$, a change in angle $\Delta\theta_k = 1°$ produces a 0.00508% change in a_k/c_k, whereas at $\theta_k = 89°$ it produces a 100% change. a_k/c_k is invariant with ϕ_k.

W=1 orthogonal parameterisation

The relationship between a_k/c_k and θ_k, ϕ_k for the $W = 1$ parameterisation is illustrated in Figure 3.10.

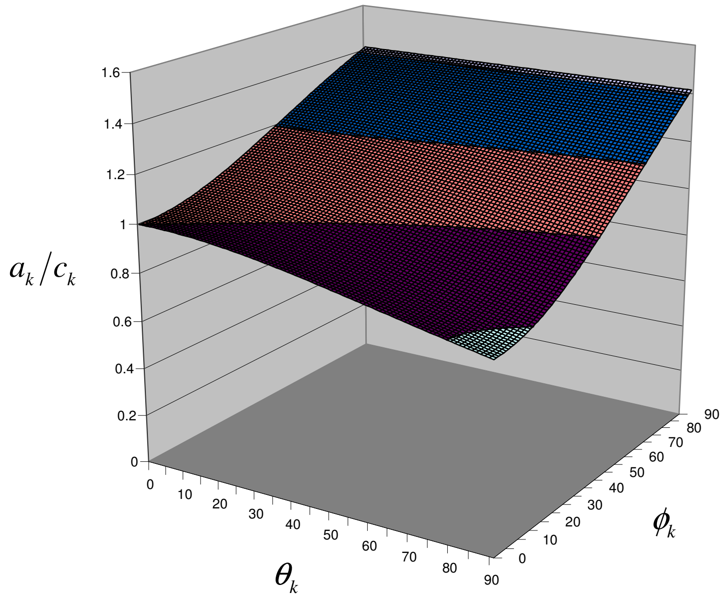
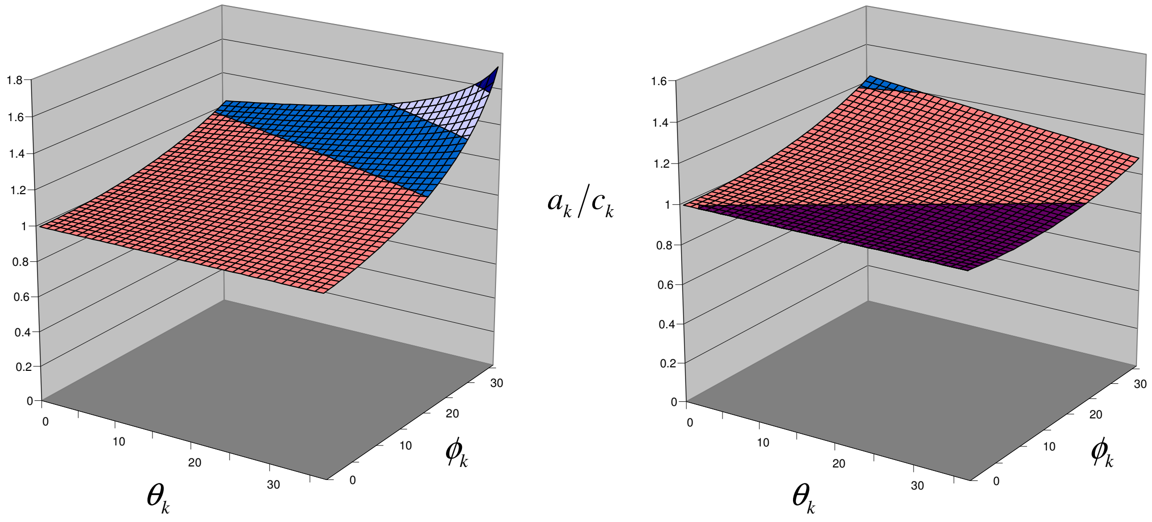


Figure 3.10 – Variation of a_k/c_k with θ_k and ϕ_k for $W = 1$

It is evident that a_k has a relatively shallow gradient for $0° \le \theta_k, \phi_k \le 90°$ compared with the $W = 0$ variations, indicating a more predictable behaviour. When $\theta_k = 0°$, a change in angle $\Delta\theta_k = 1°$ produces a 0.00510% change in a_k/c_k, whereas at $\theta_k = 89°$ it produces a 0.403% change. The $W = 0$ method is independent of the angle between tangents, and therefore the relationship displayed in Figure 3.10 is equally valid for all planar and non-planar configurations.

W=2 orthogonal parameterisation

The relationship between a_k and θ_k, ϕ_k for $W = 2a$ is illustrated in Figure 3.11 for the inflecting and non-inflecting planar configurations. If $\boldsymbol{t}_{k+1}$ is rotated about an axis through $\boldsymbol{Q}_k$, $\boldsymbol{Q}_{k+1}$, between 0° and 180°, the relationship varies between these planar configurations. It is noted that a_k is only valid for $\theta_k, \phi_k < 36.9^\circ$, and this is reflected in the scale.



Rotated 0°
(Inflecting configuration)

Rotated 180°
(Non-Inflecting configuration)

Figure 3.11 – Variation of a_k/c_k with θ_k and ϕ_k for $W = 2a$

Assuming $\theta_k, \phi_k < 36.9^\circ$, a_k/c_k does not become overly-sensitive to changes in θ_k, ϕ_k. The maximum change in a_k/c_k occurs when $\chi = 0^\circ$, in the region of $\theta_k = 39^\circ, \phi_k = 39^\circ$, where a change in angle of $\Delta\theta_k = 1^\circ$ affects the output by 0.0160%. If the applicability of the algorithm is extended by replacing $\boldsymbol{t}_k \cdot \boldsymbol{t}_{k+1}$ with its average value $(\boldsymbol{s}_k \cdot \boldsymbol{t}_k)(\boldsymbol{s}_k \cdot \boldsymbol{t}_{k+1})$, Farin's sophisticated parameterisation is reproduced (2.16)-(2.17), and the algorithm is no longer affected by the angle between the tangents. These formulations give unrealistically large values of a_k when $\theta_k, \phi_k > 60^\circ$, and become unstable at $\theta_k, \phi_k = 90°$; Farin therefore limits θ_k, ϕ_k to a maximum of $60°$. The relationship between a_k/c_k and θ_k, ϕ_k is given in Figure 3.12. Note that the angles on the graph axes have been switched for clarity.

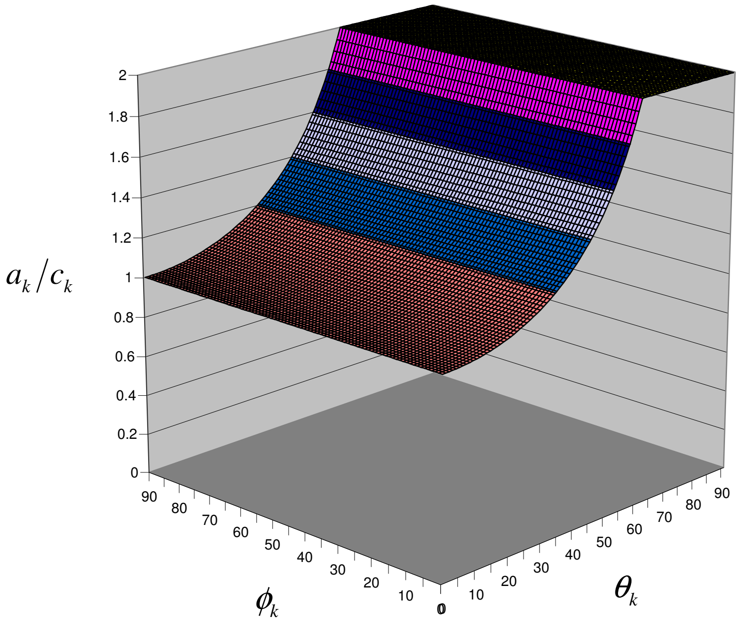


Figure 3.12 – Variation of a_k/c_k with θ_k and ϕ_k for $W = 2b$

The gradient of a_k/c_k begins to rise quickly as θ_k increases, but is capped when $\theta_k > 60°$; just before this point, a change in angle of $\Delta\theta_k = 1°$ would cause a 3.01% change in a_k/c_k. It is noted that a_k/c_k is invariant with ϕ_k.

The alternative proposed by Ball is to replace $\boldsymbol{t}_k \cdot \boldsymbol{t}_{k+1}$ with $\cos(\theta_k + \phi_k)$, limiting θ_k, ϕ_k to a maximum of $60°$. The relationship between a_k/c_k and θ_k, ϕ_k is given in Figure 3.13. Again, the axes have been switched for clarity.

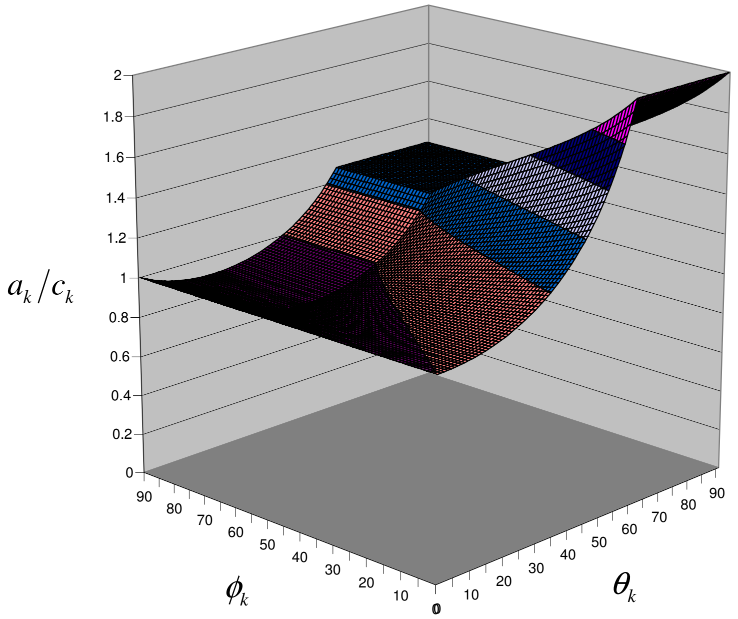


Figure 3.13 – Variation of a_k/c_k with θ_k and ϕ_k for $W = 2c$

The numerical restrictions on θ_k, ϕ_k prevent a_k from becoming overly-sensitive to changes in input data. The largest change in output resulting from a change in angle of $\Delta\theta_k = 1°$ occurs when $\theta_k = 60°, \phi_k = 0$, with a value 2.92%.

3.3 Summary of algorithms

This chapter has introduced the concept of orthogonality as a means of controlling the parameter of a curve such that it varies linearly with arc length. Three types of algorithm were presented to attain the desired orthogonality conditions at distinct locations corresponding to $W = 0,1,2$.

The $W = 0$ construction satisfies the condition at the beginning and end of each C^1 segment, i.e. $t = 0,1$. It is stable and does not produce degenerate results for 'reasonable' data configurations where $\theta_k, \phi_k < 60°$; where the angles exceed $60°$, they need to be capped otherwise the derivative magnitudes can become improbably small, zero, or even negative, and the algorithm can become overly sensitive to changes in input data.

The $W = 1$ construction satisfies the orthogonality condition at $t = \left(3 \pm \sqrt{5}\right)/6$ within each C^1 segment. The algorithm is stable for all data configurations of practical interest, and does not need to restrict applications or limit angles to ensure numerical robustness or reasonable performance. The algorithm cannot be solved analytically, requiring the roots to be found numerically; however, the lower and upper bounds for the roots is known, i.e. $0 < a_k < \sqrt{2}c_k$, and a solution is therefore guaranteed.

The $W = 2$ construction satisfies the orthogonality condition at $t = 1/3$ and $t = 2/3$ within each C^1 segment. The algorithm is restricted to applications where $\theta_k, \phi_k < 36.9°$, which could be a limiting factor when interpolating sparse data. Variations of this algorithm increase its range to $\theta_k, \phi_k < 60°$, above which the angles need to be limited.

For small θ_k, ϕ_k, all of the algorithms appear to yield sensible results, are stable, and not overly-sensitive. For more exotic data configurations, only $W = 1$ can guarantee a reasonable solution, as there are no configurations that cause it to fail or become overly-sensitive. It has the added advantage that it does not require a numerical artifice to limit $\theta_k, \phi_k < 60°$, and is independent of the angle between the tangents, χ_k. As a consequence, it does not compromise the orthogonality of non-planar segments. All of the new parameterisations are tested numerically in the next chapter, and the performance compared with existing methods.

Chapter 4

Numerical testing of parameterisation algorithms

The previous two chapters have presented a number of parameterisation algorithms, along with several corresponding derivative magnitude estimation methods. Whilst these chapters have described how the methods work, and any theoretical limitations they may have, no conclusions have been drawn with regard to their overall performance. This chapter seeks to compare the performance of the algorithms numerically and, in conjunction with the analytical evaluations from the previous chapter, conclude which are most suited to the interpolation of general data sets. A number of case studies are therefore considered, each with different geometric characteristics. Firstly, the algorithms are used to interpolate a series of evenly spaced points sampled from a circular arc, to test the performance of the algorithms with 'ideally' spaced data. Next, points are sampled unevenly from the circular arc, with the distance between points increasing exponentially. The algorithms are then used to interpolate points sampled from GCS profiles, both non-inflecting and inflecting.

The performance of the algorithms is assessed using two methods: absolute positional error, and relative curvature error. The merits of the two approaches are discussed, and consideration is given to the problem of calculating a sensible relative curvature error when the curve inflects.

4.1 Test methodology

4.1.1 Algorithms

The purpose of the tests is to assess the performance of the parameterisation algorithms presented in Chapters 2 and 3, which are listed in Tables 4.1a – 4.1b, with the aim of identifying those algorithms that are most suited to interpolating general data sets. Abbreviations of the method names are used in this chapter, and are given in brackets. Each parameterisation will be tested in conjunction with its corresponding derivative magnitude method. In the case of the orthogonal methods, the data will be interpolated twice: firstly with its corresponding derivative method, and secondly with the method that forces the orthogonality condition to be satisfied at the ends of the curve.

Parameterisation Method	**Derivative Magnitude Method**
Evenly spaced (*ES*)	Total chord length (*TCL*)
Chord length (*CL*)	(*TCL*)
Centripetal (*Cent*)	(*TCL*)
Geometric (*Geom*)	(*TCL*)
Harmonic (*Harm*)	(*TCL*)
Circular arc length (*CA*)	Total arc length (*TAL*)
Farin simple (*FSimp*)	(*FSimp*)

Table 4.1a – Existing parameterisation and derivative magnitude methods

Parameterisation Method	**Derivative Magnitude Methods**	
$W = 0$	$W = 0$	Force orthog. at ends (FO)
$W = 0$ Relaxed Planar (RP)	$W = 0\ RP$	(FO)
$W = 0$ Relaxed Circular (RC)	$W = 0\ RC$	(FO)
Circle orthogonal (CO)	(CO)	
$W = 1$	$W = 1$	(FO)
$W = 2a$	$W = 2a$	(FO)
$W = 2b$ (Farin's sophisticated)	$W = 2b$	(FO)
$W = 2c$	$W = 2c$	(FO)

Table 4.1b – Orthogonal parameterisation and derivative magnitude methods

4.1.2 Data configurations

The first case study interpolates 6 points sampled evenly from a unit semi-circle. The curvature of a circle is constant, and therefore evenly spaced points should be ideal [Cripps and Ball, 2003]. The second case study also uses points sampled from a semi-circle, but the points are spaced with exponentially increasing intervals; the purpose of unevenly spacing the points is to test the parameterisations' performance with data that is not ideally spaced.

The third case study interpolates 7 points sampled from a non-inflecting GCS curve. The curvature profile of a GCS section is rational linear and varies monotonically, therefore the sampled points are guaranteed to represent a quality curve definition. The final case study interpolates 10 points sampled from an inflecting GCS curve.

4.1.3 Performance assessment methods

The performance of the algorithms needs to be assessed using a quantitative measure. Two measures are proposed: absolute positional error, and relative curvature error. Both methods have advantages and disadvantages.

Absolute positional error is easily understood, and it is simple to relate to physical tolerances. Meaningful comparisons can be made between the resulting interpolants for a specific case study, although comparison with other curves is more difficult because the error is not relative to the size of the entity. Calculating the absolute positional error is not straightforward, however, as the interpolants are parametric curves whilst the 'true' curves, from which the points are sampled, are defined analytically.

Two methods that can overcome these issues are discussed, and provide an absolute positional error between two curves. Firstly, the absolute positional error can be taken as the distance between the curves in the direction of the normal from the host, or 'true', curve, as illustrated in Figure 4.1(a). However, this method can give misleading results in extreme situations, Figure 4.1(b).

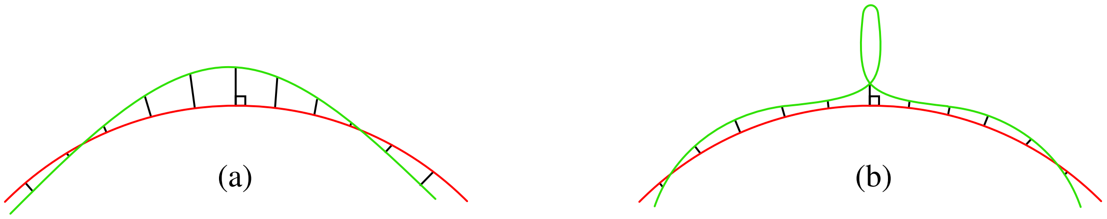


Figure 4.1 – Absolute error calculated using distance in normal direction

An alternative approach is adopted: both the parametric and analytical curves can be geometrically parameterised, that is to construct a parameter that varies linearly with a specific geometric feature of the curve. For this application, it is convenient to geometrically parameterise the curves in terms of their arc length, scaling the parameter values to be on [0,1]. The absolute error is then simply given as the distance between points with the same geometric parameter value, see Figure 4.2. As an interpolated curve must pass through the data points, the error must be zero at those locations; to ensure this criterion is satisfied, each segment must be individually geometrically parameterised.

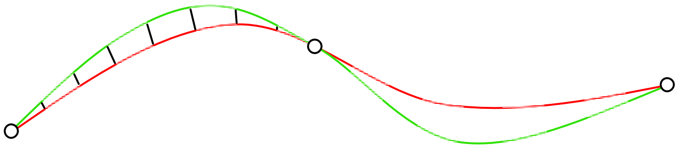


Figure 4.2 – Geometric parameterisation of curve segments

Creating a geometric parameterisation for a GCS curve is simple as it is already arc length parameterised. A circle can be represented as a GCS section, and therefore poses no problems. For a parametrically defined curve, $\boldsymbol{C}(u)$, the relationship between the host parameter, t, and the local geometric parameter, μ, must be established, where t is the local parameter for the kth curve segment:

$$t = \frac{u - s_k}{s_{k+1} - s_k}, \tag{4.1}$$

and u is the global parameter of the curve; s_k, $0 \le k \le m-2$, are the parametric locations of the data points. Generally, this relationship is complex, and best calculated numerically. A lookup table can be employed to record corresponding t and μ values, and linear (or quadratic) interpolation used to generate intermediate values. The lookup table should contain T values, and it is suggested that $T = \{2^c + 1\}$ where c is a positive integer; a value of $c = 1$ implies that the segment is sub-divided at the geometric mid-point, and incrementing c has the effect of placing additional points halfway between each of the previous points. A value of $T = 129$ per curve segment was found to be sufficient, as the difference in results obtained with $T = 257$ was negligible for the four case studies considered.

The geometric parameters need to be evenly distributed on [0,1] for each segment:

$$\mu_i = \frac{i}{T-1} \qquad i = 0,\ldots,T-1 \tag{4.2}$$

Corresponding t values are initially set to the same values as the geometric parameters:

$$t_i^0 = \mu_i \qquad i = 0,\ldots,T-1 \qquad (4.3)$$

where the superscript denotes the level of recursion. The final corresponding t values are obtained by iteration:

$$t_i^{h+1} = t_i^h + \left(\mu_i - \left(\sum_{f=0}^{i-1} l_f^h \right) \right) \qquad i = 0,\ldots,T-2 \qquad (4.4)$$

where h is the level of iteration, and:

$$l_i^h = \frac{\left\| \boldsymbol{C}\left(t_{i+1}^h \left(s_{k+1} - s_k\right) + s_k\right) - \boldsymbol{C}\left(t_i^h \left(s_{k+1} - s_k\right) + s_k\right) \right\|}{\sum_{f=0}^{T-2} \left\| \boldsymbol{C}\left(t_{f+1}^h \left(s_{k+1} - s_k\right) + s_k\right) - \boldsymbol{C}\left(t_f^h \left(s_{k+1} - s_k\right) + s_k\right) \right\|} \qquad i = 0,\ldots,T-2$$

$$(4.5)$$

To ensure the parameter values are valid following floating-point computation, any values of $t_i^{h+1} > 1.0$ should be set to 1.0. Similarly, any values of $t_i^{h+1} < 0.0$ should be set to 0.0. The stop criterion for the iterative process is:

$$t_i^h - t_i^{h+1} < \varsigma \qquad i = 0,\ldots,T-1 \qquad (4.6)$$

where ς is the specified tolerance on the parameter. For most applications, $\varsigma = 0.0001$ was found to be acceptable; decreasing ς had little impact on the results, but significantly increased computational time.

The maximum positional error between the GCS, $\boldsymbol{G}(\xi)$, and the parametric curve segment, $\boldsymbol{C}\left(s_k \le u \le s_{k+1}\right)$ is given by:

$$\max\left[\boldsymbol{G}\left(\mu_i \left(\xi_{k+1} - \xi_k\right) + \xi_k\right) - \boldsymbol{C}\left(t_i^h \left(s_{k+1} - s_k\right) + s_k\right) \right] \qquad i = 0,\ldots,T-1 \qquad (4.7)$$

where ξ_k are the arc lengths on the GCS corresponding to the sampled points, i.e. $\boldsymbol{G}(\xi_k)=\boldsymbol{C}(s_k)$, $0 \le k \le m-2$.

The second measure of performance is relative curvature error. Curvature can highlight problems that are not apparent when considering the positional error, as it is highly sensitive to changes in shape. For example, curvature values can indicate if a derivative magnitude is too small or too large. Relative curvature error is used because the error term is scaled according to the size of the entity, and therefore allows comparisons to be made between curves of different proportions.

The curvature of a parametrically defined curve is given by:

$$\kappa(u)=\frac{\left\|\boldsymbol{C}_u(u)\times\boldsymbol{C}_{uu}(u)\right\|}{\left\|\boldsymbol{C}_u(u)\right\|^3} \tag{4.8}$$

and the relative curvature error (as a percentage) is given by:

$$r=100\cdot\frac{\left|\kappa_{actual}(u)-\kappa_{true}(u)\right|}{\left|\kappa_{true}(u)\right|} \tag{4.9}$$

If the curve inflects, curvature values tend to zero at the point of inflection, causing (4.9) to become infinite unless $\kappa_{actual}(u)=\kappa_{true}(u)$. The relative curvature error in this region is therefore less meaningful. O'Neill [1993] and Ma [2006] propose a method for handling relative curvature values at an inflection point. When $\left|\kappa(u)\right|<\rho$, where ρ is a small curvature threshold, $\left|\kappa(u)\right|$ is transformed to $f\left(\left|\kappa(u)\right|\right)=\left(\rho+\left|\kappa(u)\right|\right)/2$, as shown in Figure 4.3.

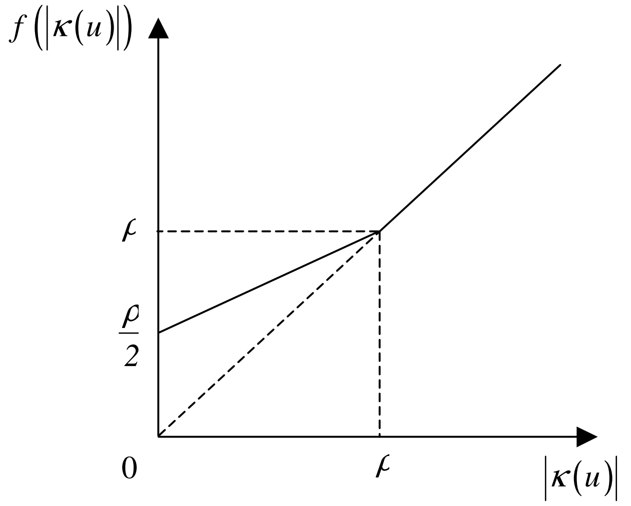


Figure 4.3 – Small curvature transformation

The choice of ρ will affect the resulting relative curvature error. O'Neill [1993] suggests that $\rho = 1.0 \times 10^{-4}$ is a reasonable value for practical purposes; Ma [2006] argues that ρ can be chosen according to the maximum curvature of the application, and proposes $\rho = 0.05 \times |\kappa_{\max}(u)|$. The latter approach is adopted in this thesis, as it ensures that the method is geometrically invariant under translation, rotation and scaling, since $|\Delta\kappa(u)|$ is invariant under translation and rotation, and relative error is scaling invariant.

Despite handling small curvature errors as a special case, the usefulness of the maximum error as a performance assessment method is questionable. The maximum relative error is very likely to occur at, or near, the inflection point, even when the curvature elsewhere is poor. Where the curve contains an inflection, the absolute positional error is to be considered the more consistent and reliable assessment method. If the curve does not inflect, both measures are useful for comparing the performance of different methods.

4.2 Case study 1 - circular arc, points spaced evenly

The first case study interpolates 6 points sampled equally from a semi-circle, i.e.:

$$\boldsymbol{Q}_k = \left(\cos\left[\frac{\pi k}{5}\right], \sin\left[\frac{\pi k}{5}\right], 0 \right) \qquad k = 0, \ldots, 5 \qquad (4.10)$$

The curve interpolation is constrained; 6 data points are used following the heuristic rule given by Cripps and Lockyer [2005] for approximating a unit circle to within a 5% relative curvature error using circle orthogonal parameterisation:

$$N = 1 + \mathrm{int}\left(\frac{6\theta}{\pi}\right) \qquad (4.11)$$

where N here is the number of data points to use in a constrained interpolation, and θ is the span of the circular arc in radians. The data points and tangents are illustrated in Figure 4.4, and are interpolated using the chord length parameterisation and the total chord length derivative magnitudes.

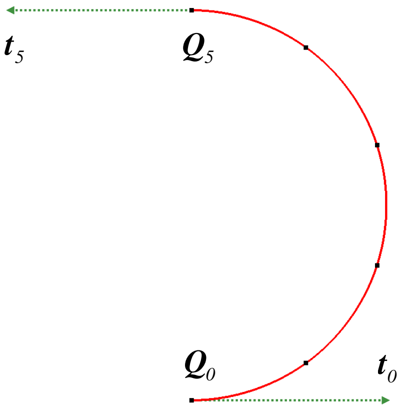


Figure 4.4 – Case study 1 – evenly spaced points from a semi-circle

The maximum absolute positional errors and the maximum relative curvature errors for case study 1 are displayed in Table 4.2 for all of the algorithms specified in Table 4.1. It is broken into 3 sub-sections, corresponding to existing parameterisations, orthogonal parameterisations with their corresponding derivatives, and orthogonal parameterisations with the derivatives that force orthogonality at the ends.

Parameterisation	**Derivative Magnitude**	**Max Absolute Positional Error**	**Relative Curvature Error %**
ES	*TCL*	0.000929	6.798927
CL	*TCL*	0.000929	6.798927
Cent	*TCL*	0.000929	6.798927
Geom	*TCL*	0.000929	6.798927
Harm	*TCL*	0.000929	6.798927
CA	*TAL*	0.000468	3.538828
FSimp	*FSimp*	0.005421	25.834078
$W = 0$	$W = 0$	0.000447	3.519094
$W = 0$ *RP*	$W = 0$ *RP*	0.000447	3.519094
$W = 0$ *RC*	$W = 0$ *RC*	0.000447	3.519094
CO	*CO*	0.000447	3.519094
$W = 1$	$W = 1$	0.000465	3.535666
$W = 2a$	$W = 2a$	0.000481	3.550879
$W = 2b$	$W = 2b$	0.001265	4.279773
$W = 2c$	$W = 2c$	0.000481	3.550879
$W = 0$	*FO*	0.000447	3.519094
$W = 0$ *RP*	*FO*	0.000447	3.519094
$W = 0$ *RC*	*FO*	0.000447	3.519094
$W = 1$	*FO*	0.000447	3.519094
$W = 2a$	*FO*	0.000447	3.519094
$W = 2b$	*FO*	0.000447	3.519094
$W = 2c$	*FO*	0.000447	3.519094

Table 4.2 – Case study 1 results

The positional error profiles are given for each of the three sub-sections – see Figures 4.5 to 4.7. The chord length method is shown on all for comparison.

It is noted that many of the algorithms can produce identical output; where this occurs, a single colour is used to represent all the identical error profiles. If two distinct curves are very close, one may become obscured; consideration of the data table should clarify its performance. Curvature error profiles generally replicate the information in the positional error plots, and are therefore omitted.

4.2.1 Positional error profiles

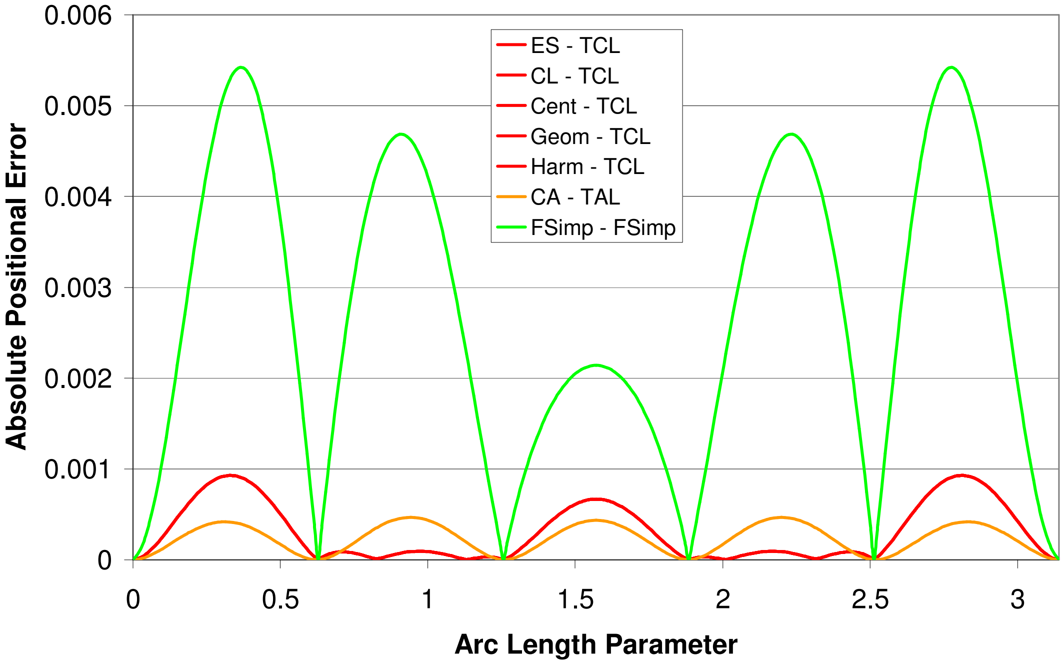


Figure 4.5 – Case study 1 positional error profiles – existing parameterisations

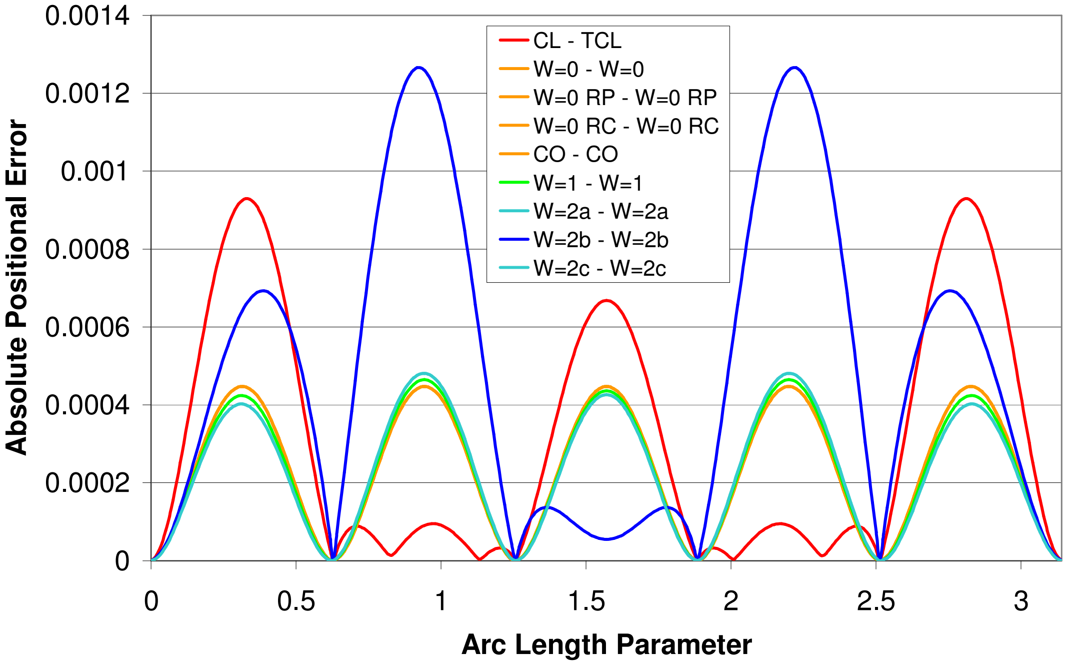


Figure 4.6 – Case study 1 positional error profiles – orthogonal parameterisations with corresponding derivative magnitudes

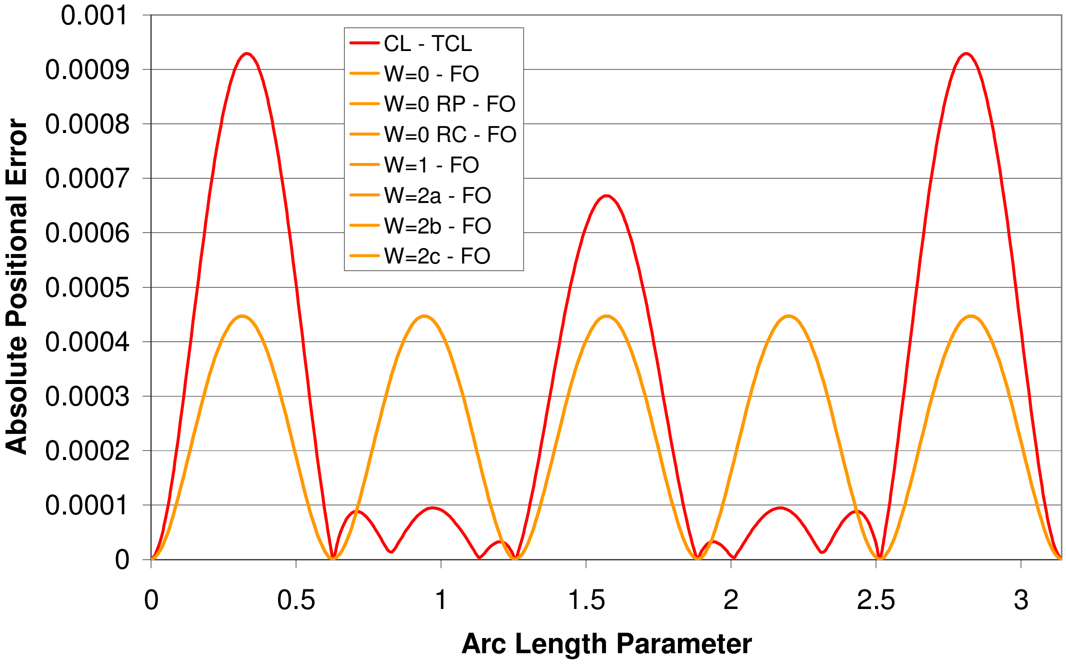


Figure 4.7 – Case study 1 positional error profiles – orthogonal parameterisations with derivative magnitudes that force orthogonality at the ends

4.2.2 Analysis of results

Considering Figure 4.5, it is evident that many of the existing parameterisations produce identical results for this particular case study. The points are equally spaced, and therefore the equally spaced, chord length and centripetal methods yield the same parameter values. The geometric and harmonic averages take an average of these three methods, and therefore also produce identical results. The circular arc method is seen to have marginally better performance, as the maximum error is smaller, and consistent for each segment; this is because the derivative magnitude estimate is superior than the chord length approximation. Farin's simple construction method is seen to be poor, as it assumes that the derivative magnitudes for each segment will be $1.2c$, where c is the chord length between points. For this specific case, $c = 0.618$ and the arc length between the points is 0.628, giving a ratio of 1.02; hence, the derivative magnitudes are too large.

The orthogonal parameterisations all have comparable maximum positional errors in Figure 4.6, with the exception being the $W = 2b$ (Farin's sophisticated) method, which is significantly worse than the others. Again, this is caused by the derivative magnitudes being too large. It is noted that for planar, circular data, all of the $W = 0$ methods and the circle orthogonal method yield identical results; they easily outperform the chord length method, and marginally outperform the circular arc method.

All of the interpolants in Figure 4.7 have the same derivative magnitude method, and therefore yield identical results because the points are equally spaced. It is evident that equally spaced points are indeed 'ideal' for a circle, because the error profile is identical for each segment. This also implies that the derivative magnitudes produced by the method that forces orthogonality at the ends are consistent with the derivatives at the internal knots on the C^2 interpolated curve. This is also true of the $W = 0$ derivatives, and all its variations.

4.3 Case study 2 - circular arc, points spaced exponentially

The second case study again interpolates 6 points sampled from a semi-circle, but the points are spaced unequally to assess the performance of the algorithms with data that is not 'ideal'. There are an infinite number of ways to unequally space data points on a semi-circle, but it is desirable that the point spacing varies from dense to sparse, as the affect of point spacing on the resulting error profiles is then clear. One approach that achieves this is to space the points exponentially:

$$\boldsymbol{Q}_k = \left(\cos\left[\pi \frac{\left(e^{k/5} - 1\right)}{\left(e - 1\right)} \right], \sin\left[\pi \frac{\left(e^{k/5} - 1\right)}{\left(e - 1\right)} \right], 0 \right) \qquad k = 0, \ldots, 5 \tag{4.12}$$

The points, tangents, and the chord length interpolant are shown in Figure 4.8:

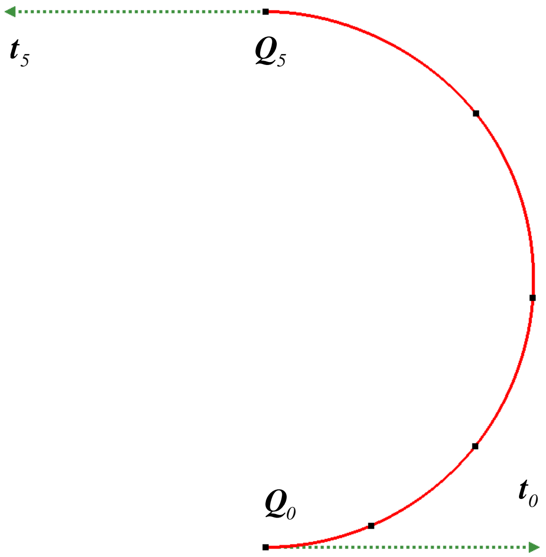


Figure 4.8 – Case study 2 – points spaced exponentially from a semi-circle

The maximum absolute positional errors and the maximum relative curvature errors for case study 2 are displayed in Table 4.3 for all of the algorithms specified in Table 4.1.

Parameterisation	Derivative Magnitude	Max Absolute Positional Error	Relative Curvature Error %
ES	*TCL*	0.028327	165.616606
CL	*TCL*	0.004534	15.870222
Cent	*TCL*	0.016747	73.262239
Geom	*TCL*	0.016220	70.248957
Harm	*TCL*	0.016226	69.693339
CA	*TAL*	0.002037	7.099909
FSimp	*FSimp*	0.007686	28.927641
$W = 0$	$W = 0$	0.002382	8.217297
$W = 0$ *RP*	$W = 0$ *RP*	0.002382	8.217297
$W = 0$ *RC*	$W = 0$ *RC*	0.002382	8.217297
CO	*CO*	0.002382	8.217297
$W = 1$	$W = 1$	0.002090	7.269744
$W = 2a$	$W = 2a$	0.001848	6.488423
$W = 2b$	$W = 2b$	0.003542	9.268460
$W = 2c$	$W = 2c$	0.001848	6.488423
$W = 0$	*FO*	0.002451	8.462202
$W = 0$ *RP*	*FO*	0.002451	8.462202
$W = 0$ *RC*	*FO*	0.002451	8.462202
$W = 1$	*FO*	0.002380	8.288994
$W = 2a$	*FO*	0.002323	8.151178
$W = 2b$	*FO*	0.001570	6.233795
$W = 2c$	*FO*	0.002323	8.151178

Table 4.3 – Case study 2 results

Again, the positional error profiles are given for each of the three sub-sections – see Figures 4.9 to 4.11. The chord length method is shown on all profiles for comparison.

4.3.1 Positional error profiles

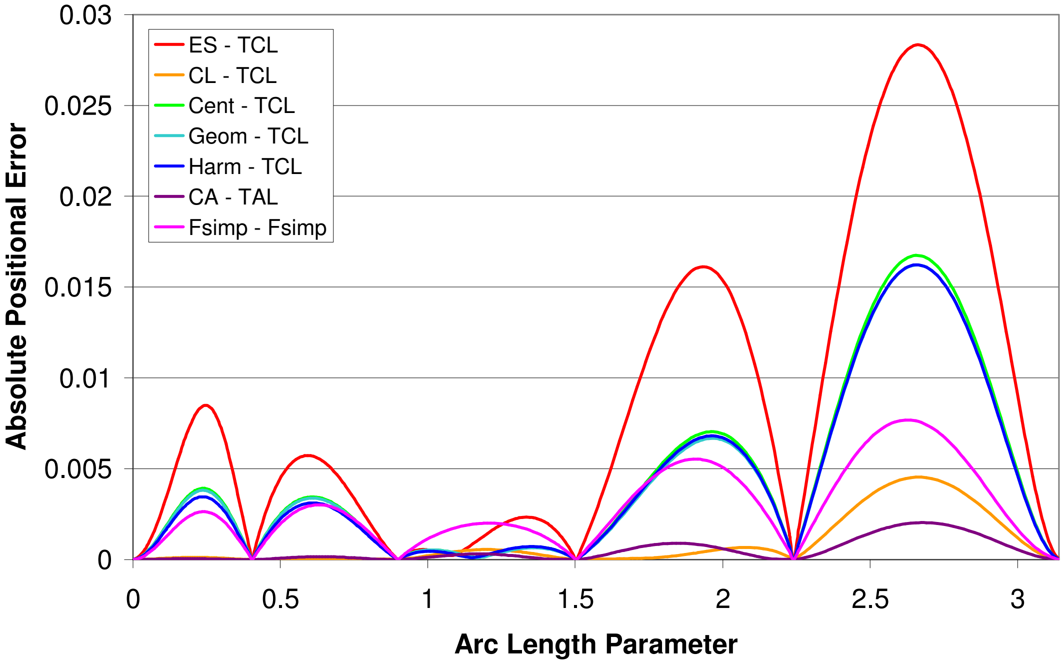


Figure 4.9 – Case study 2 positional error profiles – existing parameterisations

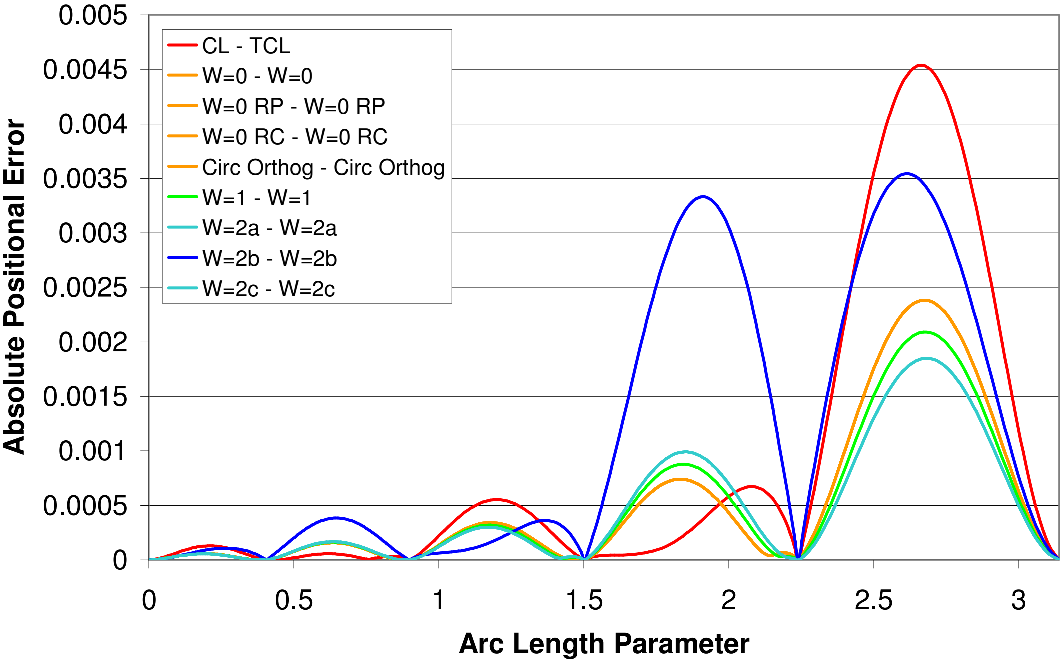


Figure 4.10 – Case study 2 positional error profiles – orthogonal parameterisations with corresponding derivative magnitudes

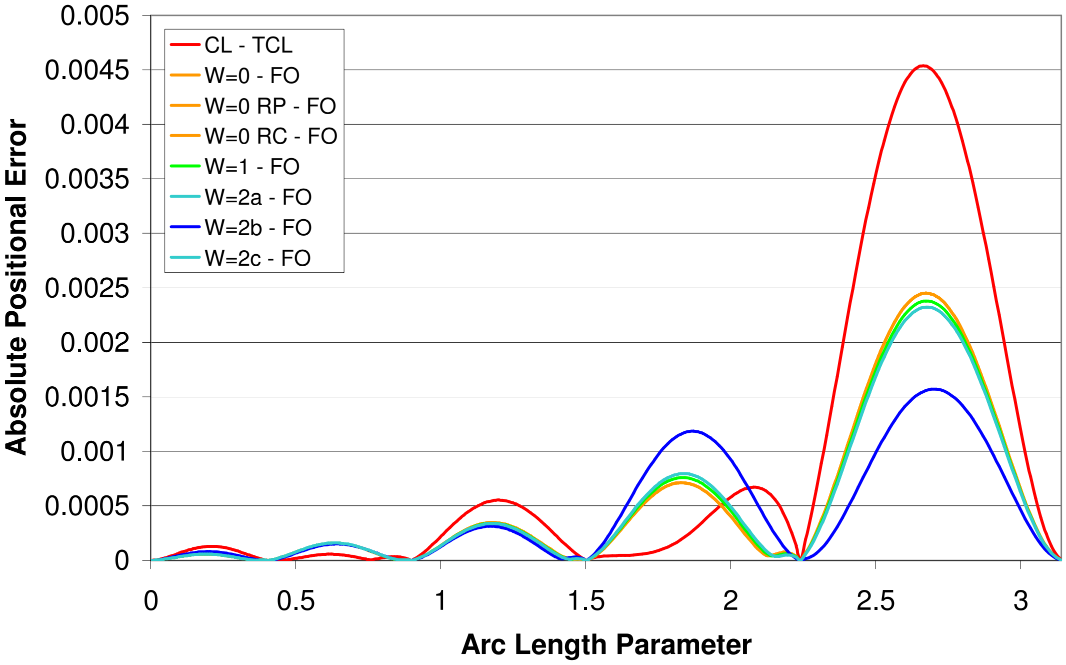


Figure 4.11 – Case study 2 positional error profiles – orthogonal parameterisations with derivative magnitudes that force orthogonality at the ends

4.3.2 Analysis of results

The positional error in Figure 4.9 reveals that the equally spaced parameterisation method performs very badly on unequally spaced data. Consequently, the geometric and harmonic parameterisations are also poor. Interestingly, the centripetal parameterisation performs badly too; Lee [1989] claims that it performs well when the data contains sharp corners, but clearly it cannot handle simple configurations when they are not 'ideally' spaced.

Again, the orthogonal parameterisations all perform well, with the $W = 2b$ parameterisation being the weakest (Figure 4.10) because the derivative magnitudes are too large. However, when the $W = 2b$ parameterisation is used in conjunction with the derivatives that force orthogonality at the ends, the performance is much improved (Figure 4.11). The remaining orthogonal parameterisations are approximately comparable when using the derivatives that force orthogonality.

It is evident from all of the error distributions that as the point spacing becomes sparser, the errors become significantly larger. This is true for all the parameterisations and derivative magnitude methods, and therefore the spacing of data points is a significant factor in the quality of the resulting interpolant.

4.4 Case study 3 - GCS, non inflecting

The third case study is a non-inflecting planar GCS curve. GCS curves, first proposed by Ali [1994], are defined by their curvature profile, which is rational linear and varies monotonically. The curvature at any point, $\kappa(\xi)$, is given by:

$$\kappa(\xi) = \frac{(\kappa_1 - \kappa_0 + r\kappa_1)\xi + \kappa_0 S}{r\xi + S}, \qquad 0 \le \xi \le S,\ r > -1 \tag{4.11}$$

where ξ is the arc length parameter, κ_0 the starting curvature, κ_1 the finishing curvature, r the shape factor, and S the total arc length.

The winding angle of a GCS, $\theta(s) = \int_0^{\xi} \kappa(\xi)\, d\xi$, is given by Ali as:

$$\theta(\xi) = \begin{cases} \dfrac{S}{r^2}\left[(1+r)(\kappa_0 - \kappa_1)\log\left(1 + r\dfrac{\xi}{S}\right) + r\left((1+r)\kappa_1 - \kappa_0\right)\dfrac{\xi}{S}\right] & \text{if } r \neq 0 \\ \dfrac{(\kappa_1 - \kappa_0)\dfrac{\xi^2}{2} + \kappa_0 S\xi}{S} & \text{otherwise} \end{cases} \tag{4.12}$$

A point on a GCS, $(x(\xi), y(\xi), 0)$, must be obtained by numerical integration as they involve Fresnal integrals:

$$x(\xi) = \int_0^{\xi} \cos(\theta(\xi))\, d\xi \tag{4.13}$$

$$y(\xi)=\int_{0}^{\xi}\sin\left(\theta(\xi)\right)\,d\xi \qquad (4.14)$$

The previous section highlighted the fact that the location of the data points has a significant effect on the resulting interpolant. This case study evaluates 7 points from a GCS curve, which are given and assumed to adequately characterise the shape – Figure 4.12. The number and spacing of data points was determined using methods that are discussed in Chapter 5.

GCS Data:
Length, $S = 0.90$
Shape Factor, $r = -0.95$
Start Curv., $\kappa_0 = 1.0$
End Curv., $\kappa_1 = 20.0$

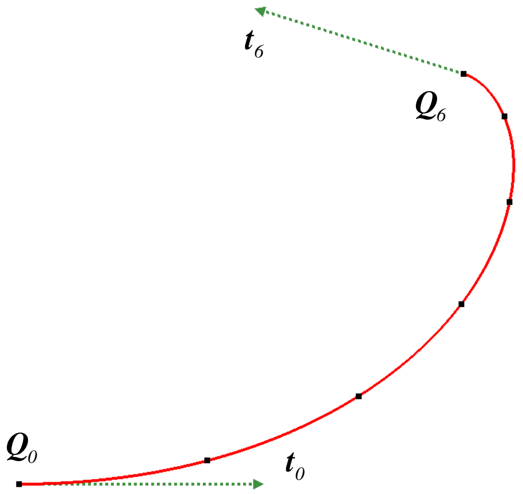


Figure 4.12 – Case study 3 – points from a non-inflecting GCS curve

The maximum absolute positional errors and the maximum relative curvature errors for case study 3 are displayed in Table 4.4 for all of the algorithms specified in Table 4.1.

Parameterisation	**Derivative Magnitude**	**Max Absolute Positional Error**	**Relative Curvature Error %**
ES	*TCL*	0.007605	416.347974
CL	*TCL*	0.000478	8.933751
Cent	*TCL*	0.003184	110.138838
Geom	*TCL*	0.003637	131.678403
Harm	*TCL*	0.003682	134.145154
CA	*TAL*	0.000430	8.409158
FSimp	*FSimp*	0.001086	34.714553
$W = 0$	$W = 0$	0.000151	11.328833
$W = 0$ *RP*	$W = 0$ *RP*	0.000151	11.328833
$W = 0$ *RC*	$W = 0$ *RC*	0.000263	12.830279
CO	*CO*	0.000362	12.002592
$W = 1$	$W = 1$	0.000167	9.900945
$W = 2a$	$W = 2a$	0.000306	17.506601
$W = 2b$	$W = 2b$	0.000786	28.783219
$W = 2c$	$W = 2c$	0.000306	17.506601
$W = 0$	*FO*	0.000167	11.635568
$W = 0$ *RP*	*FO*	0.000167	11.635568
$W = 0$ *RC*	*FO*	0.000628	10.341897
$W = 1$	*FO*	0.000377	10.262211
$W = 2a$	*FO*	0.000766	13.020017
$W = 2b$	*FO*	0.000560	10.249609
$W = 2c$	*FO*	0.000766	13.020017

Table 4.4 – Case study 3 results

The positional error profiles are given for each of the three sub-sections – see Figures 4.13 to 4.15. The chord length method is shown on all profiles for comparison.

4.4.1 Positional error profiles

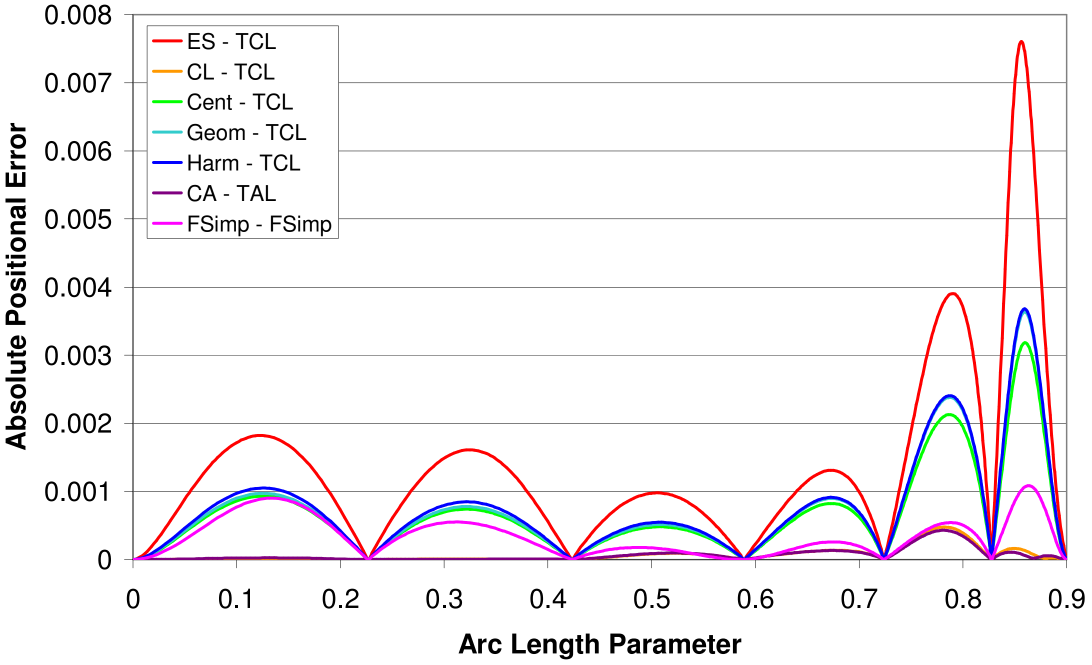


Figure 4.13 – Case study 3 positional error profiles – existing parameterisations

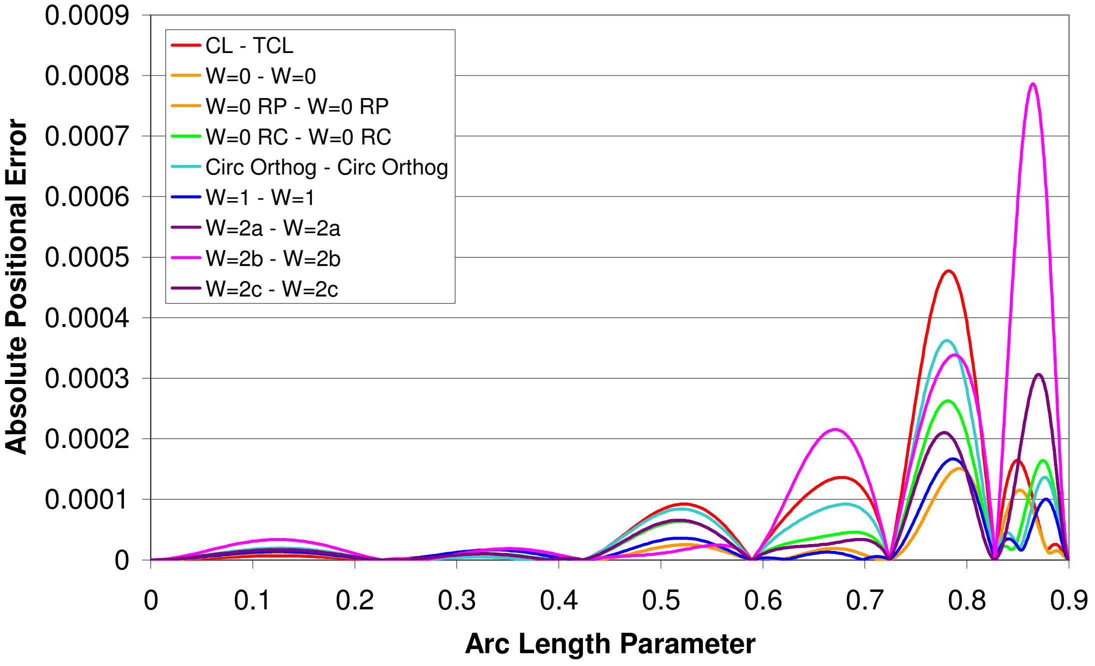


Figure 4.14 – Case study 3 positional error profiles – orthogonal parameterisations with corresponding derivative magnitudes

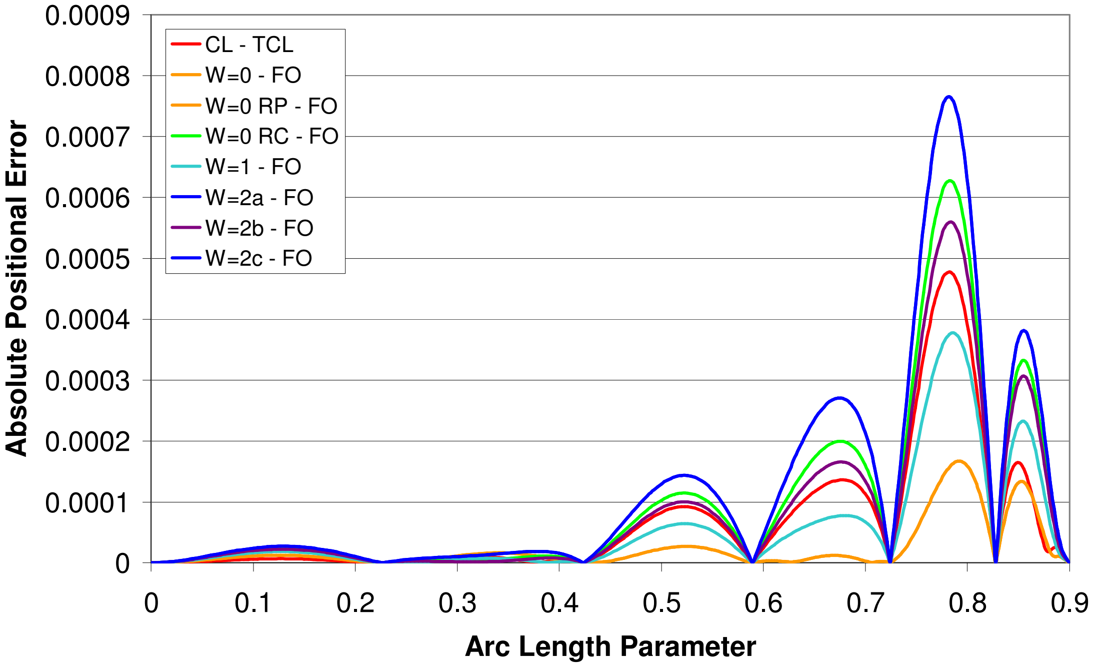


Figure 4.15 – Case study 3 positional error profiles – orthogonal parameterisations with derivative magnitudes that force orthogonality at the ends

4.4.2 Analysis of results

Once again, Figure 4.13 illustrates the fact that the equally spaced and average parameterisations give poor results when the data is unequally spaced, principally because the equally spaced method is not distributing the parameter values to correspond with the spacing of data points. It is seen again that the centripetal method, despite Lee's [1989] claim, performs significantly worse than the chord length method. Lee argued that the centripetal method was superior to chord length, especially when the data contains sharp changes in direction; however, where the data represents a smooth curve with unequally spaced data points, the chord length parameterisation is evidently better. The centripetal method does not attempt to distribute parameters such that they vary linearly with arc length; rather it reduces the parametric interval for larger spans so that it can increase the parametric interval for tight turns where the data is very closely spaced. This allows it to perform well with very specific data sets that contain sparse data points and sharp turns controlled by two very close data points; however, it is fundamentally inappropriate for general interpolation where the data points are distributed smoothly such that they characterise the shape. The circular arc method consistently produces a good approximation of the true arc length, and therefore performs well.

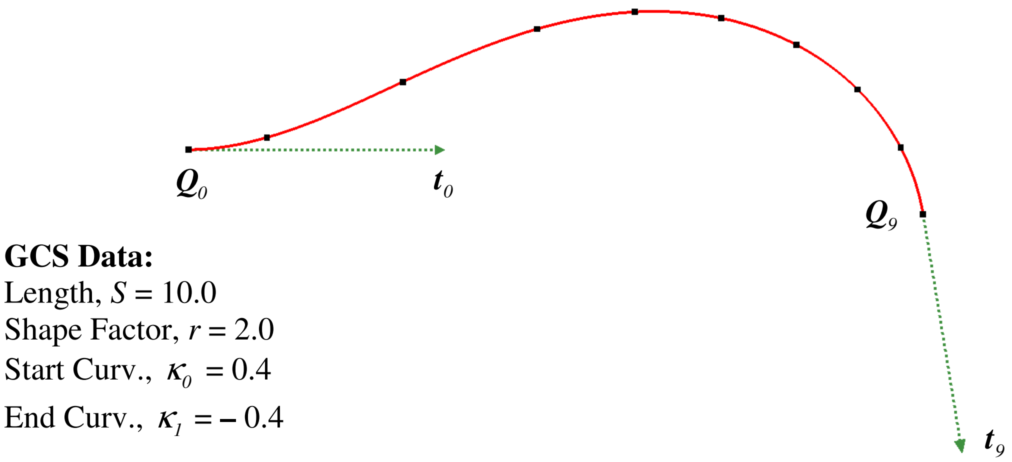
The majority of the orthogonal parameterisations have similar relative curvature errors to the chord length method (Table 4.4), but significantly smaller maximum positional errors. This is because the maximum positional errors occur in the region of high curvature, and therefore the chord becomes a less accurate representation of the arc length in that vicinity; the relative curvature error does not highlight the problem as the absolute curvature value is large. The $W = 0$ and $W = 1$ methods perform particularly well, although the $W = 2b$ method is poor, owing principally to derivative magnitudes that are too large.

When the derivatives that force orthogonality at the ends are used, the $W = 0$ and $W = 1$ methods again perform very well, but the positional error for the $W = 0\ RC$ and $W = 2$ methods is larger than the chord length interpolant. The $W = 0\ RC$ parameterisation assumes that the tangent between the chord is the same at the start and end of each segment, which is only true for a circle, and therefore would not be expected to perform well for a spiral segment; clearly forcing the derivatives to be orthogonal at the ends of the interpolant does not improve the performance for this parameterisation. The $W = 2b$ method is much improved by the derivatives that force orthogonality, but it is still slightly worse than the chord length method.

It is apparent that, despite the points being much closer, the positional error is significantly larger in the regions of high curvature; the relative curvature error does not increase in the same fashion because the absolute curvature is also increasing.

4.5 Case study 4 - GCS, inflecting

The final case study is a planar GCS curve that inflects. There are 10 points, which are given and assumed to characterise the shape (Figure 4.16); the methods that are used to obtain the number and spacing of points are discussed in Chapter 5.



GCS Data:
Length, $S = 10.0$
Shape Factor, $r = 2.0$
Start Curv., $\kappa_0 = 0.4$
End Curv., $\kappa_1 = -0.4$

Figure 4.16 – Case study 4 – points from an inflecting GCS curve

The maximum absolute positional errors and the maximum relative curvature errors for case study 4 are displayed in Table 4.5 for all of the algorithms specified in Table 4.1.

Parameterisation	**Derivative Magnitude**	**Max Absolute Positional Error**	**Relative Curvature Error %**
ES	*TCL*	0.021108	124.654416
CL	*TCL*	0.001595	28.461913
Cent	*TCL*	0.012522	66.406228
Geom	*TCL*	0.011998	66.992622
Harm	*TCL*	0.012005	72.992814
CA	*TAL*	0.001498	29.342923
FSimp	*FSimp*	0.005868	30.259348
$W = 0$	$W = 0$	0.001006	28.790816
$W = 0$ *RP*	$W = 0$ *RP*	0.001006	28.790816
$W = 0$ *RC*	$W = 0$ *RC*	0.001421	27.390656
CO	*CO*	0.001567	27.476649
$W = 1$	$W = 1$	0.001259	27.596282
$W = 2a$	$W = 2a$	0.001518	27.184208
$W = 2b$	$W = 2b$	0.001065	25.199613
$W = 2c$	$W = 2c$	0.001518	27.184208
$W = 0$	*FO*	0.001013	28.851117
$W = 0$ *RP*	*FO*	0.001013	28.851117
$W = 0$ *RC*	*FO*	0.001586	28.712727
$W = 1$	*FO*	0.001360	28.401553
$W = 2a$	*FO*	0.001721	28.823576
$W = 2b$	*FO*	0.001535	28.969306
$W = 2c$	*FO*	0.001721	28.823576

Table 4.5 – Case study 4 results

The positional error profiles are given for each of the three sub-sections – see Figures 4.17 to 4.19. The chord length method is shown on all profiles for comparison.

4.5.1 Positional error profiles

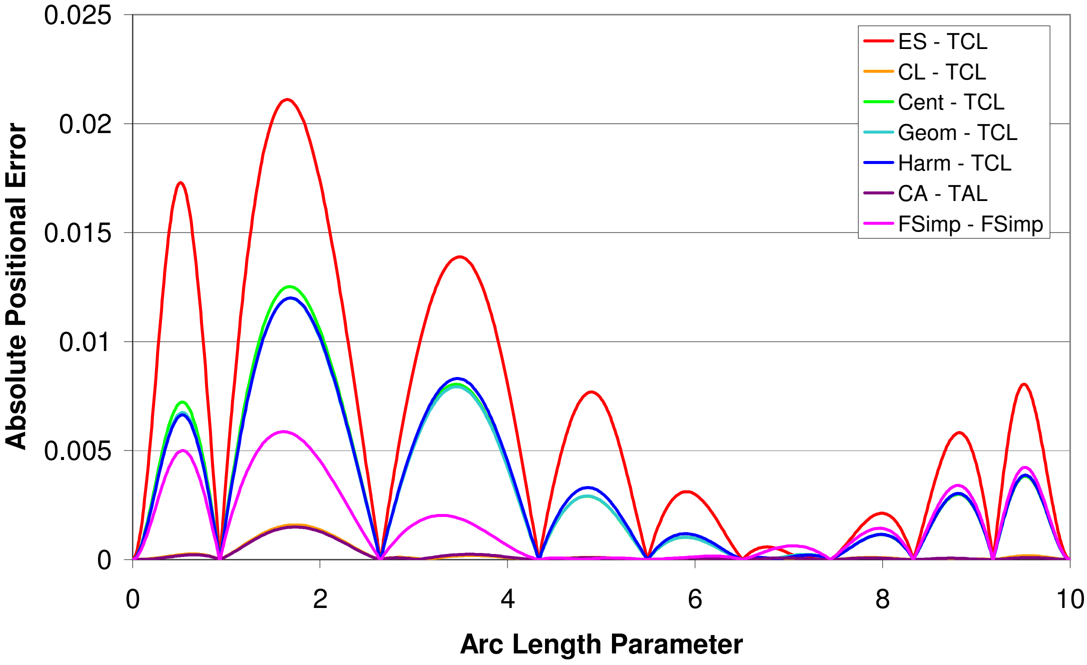


Figure 4.17 – Case study 4 positional error profiles – existing parameterisations

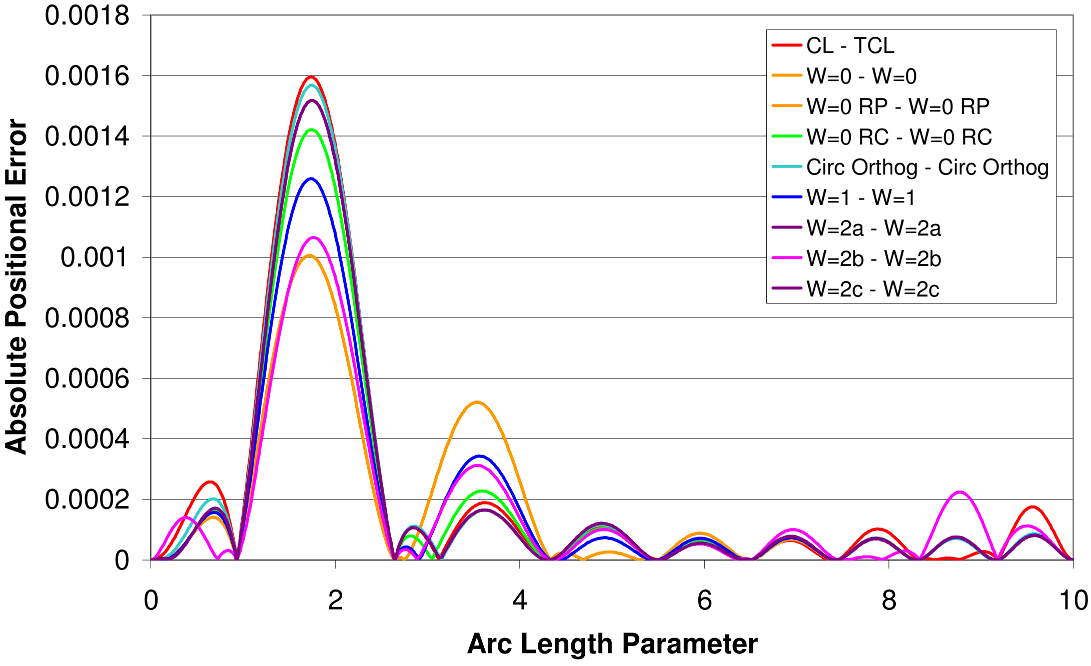


Figure 4.18 – Case study 4 positional error profiles – orthogonal parameterisations with corresponding derivative magnitudes

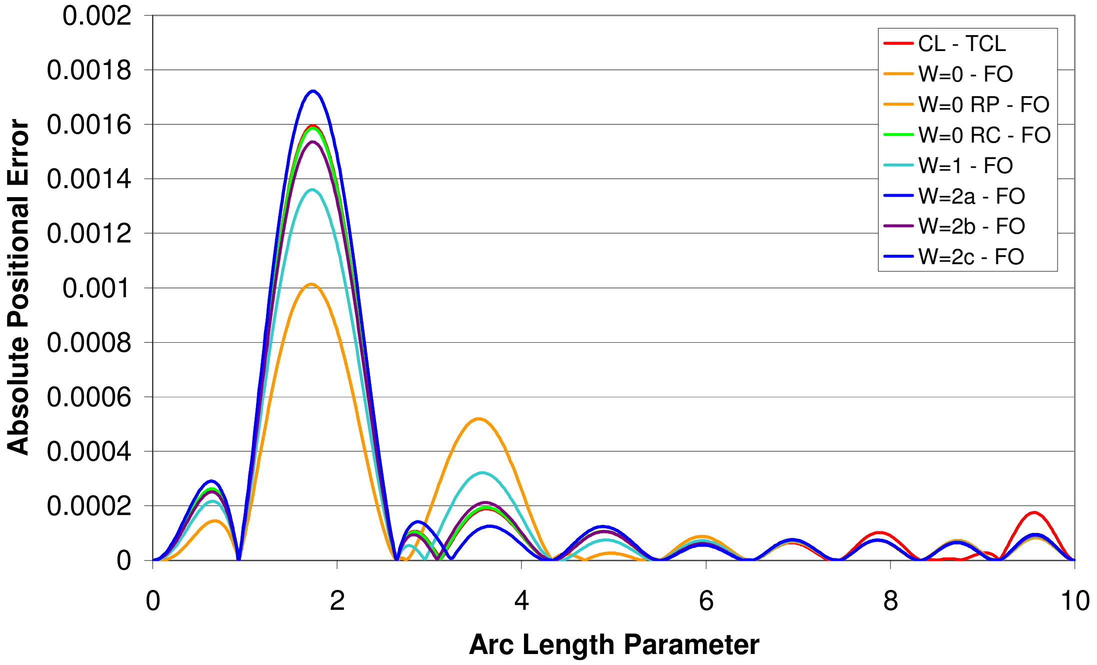


Figure 4.19 – Case study 4 positional error profiles – orthogonal parameterisations with derivative magnitudes that force orthogonality at the ends

4.5.2 Analysis of results

Figure 4.17 again emphasises the poor performance of the existing algorithms; the equally spaced and average parameterisations due to the data points not being spaced with equidistant intervals, the centripetal parameterisation because it is not striving for an arc length parameterisation on smooth data, and Farin's simple method because it assumes a fixed arc length to chord ratio. The chord length and circular arc parameterisations have very comparable performance, but are generally inferior to the orthogonal methods.

It is noted that the maximum relative curvature errors displayed in Table 4.5 are comparable for all of the orthogonal methods, although the usefulness of this error term is questionable as the curves inflect. The $W = 0\ RC$ method is only marginally better than the chord length parameterisation; again, it was not anticipated that it would perform well on data sets that are not symmetrical (i.e. angle between chord and tangent equal at either end of each segment). The $W = 0$, $W = 1$ and $W = 2b$ parameterisations all have a comparatively small maximum positional error when used in conjunction with their corresponding derivative magnitude methods, indicating that the C^1 curve construction process is not adversely affected by inflecting configurations. The $W = 0$ and $W = 1$ parameterisations also perform well when coupled with the derivative method that forces orthogonality at the ends, as the new magnitudes differ only slightly from the original ones; however, the $W = 2b$ case no longer performs well, as the magnitudes are smaller.

The positional error profiles all indicate that the point density in the region of the inflection is not quite sufficient, as this is the region that has the greatest error. This does not, however, invalidate the test results, as parameterisations must be able to operate with less than 'ideal' data.

4.6 Summary

The numerical case studies considered in this chapter allow certain conclusions to be drawn with regard to the performance of the parameterisation and derivative magnitude estimation algorithms presented in chapters 2 and 3. Three broad categories are used to summarise the performance: 'poor', 'reasonable' and 'good' – see Table 4.6.

Poor	**Reasonable**	**Good**
Equally spaced	Chord Length	$W = 0$
Centripetal	Circular Arc	$W = 0$ Relaxed Planar
Geometric	$W = 0$ Relaxed Circular	$W = 1$
Harmonic	Circular Orthogonal	
Farin Simple	$W = 2a$	
	$W = 2b$ (Farin Sophisticated)	
	$W = 2c$	

Table 4.6 – Performance of algorithms

‘Poor’ parameterisations are defined as those which are insensitive to either the spacing of data points, or the configuration of tangents directions. The worst method was the equally spaced parameterisation, as it simply assigns a constant parametric interval to each segment, irrespective of the geometric distribution of points. The geometric and harmonic average parameterisations are therefore adversely affected. The centripetal parameterisation does not attempt to distribute parameter values to reflect the arc length of the curve, rather concentrating the parametric intervals for the segments that have shorter chords; this strategy is only appropriate with very specific data sets that contain sharp corners and sparse flat regions. Finally, Farin’s simple method only succeeds if the ratio of the arc length to the chord is close to the ‘magic’ number of 1.2.

The next category is the ‘reasonable’ parameterisations, which are defined as those that produce parameter values and derivative magnitudes that closely resemble the true arc length-equivalents, and perform well for the majority of data configurations. Of the existing methods, this category includes the chord length and circular arc parameterisations. The circular arc parameterisation has a more accurate method of estimating the arc length of a segment, and therefore usually outperforms the chord length parameterisation by a small margin. The chord length approach, however, is very stable, simple and intuitive. The $W = 0\ RC$ parameterisation assumes that the angle between the tangent and the chord is constant at the beginning and end of each segment, which for general data proves to compromise the results, although it usually outperforms the chord length method. The CO method does not use the intermediate tangent information, assuming that the data points lie on a circle. Whilst this is an advantage when the intermediate tangents are not available, the results are not as good for non-circular data compared with other orthogonal methods. The $W = 2$ parameterisations did not perform consistently well in the numerical tests; interestingly, the $W = 2b$ occasionally performed extremely well, but with other geometries was very poor.

The last category, or the 'good' parameterisations, is defined as those that consistently performed well for all the geometries considered, and includes the $W = 0$ and $W = 1$ methods. Of these parameterisations, the case studies highlight that neither method outperforms the other in all circumstances, and that the difference in performance between these methods is generally negligible. However, the analytical investigation from the previous chapter provides additional guidance when making a decision.

The $W = 0$ method performed consistently well with both its corresponding derivative magnitude method, and the derivatives that force orthogonality at the ends. The algorithm requires that the angles between the tangents and chord be limited to $60°$ to ensure numerical stability, although such geometries will rarely be encountered when interpolating data that characterises the desired shape. The $W = 0$ method is a function of the angle between tangents for each segment; the $W = 0\ RP$ version relaxes the construction, removing the dependence on the angle between tangents, but this compromises the orthogonality conditions for non-planar and inflecting configurations.

The $W = 1$ method also performed consistently well with both derivative methods. The algorithm is stable for all reasonable data configurations, and has the advantage that it is independent of the angle between tangents. This means that the orthogonality condition is not compromised by relaxing the construction to a planar configuration.

It is evident that formulating the parameterisations in terms of the orthogonality between first and second derivatives is a much superior approach of distributing parameter values compared with the conventional methodologies. In all the numerical cases considered, the $W = 0$ and $W = 1$ methods outperformed the chord length parameterisations in terms of maximum absolute positional error. The $W = 1$ parameterisation method is recommended over the $W = 0$ method, as it is stable for all reasonable data configurations, and is independent of the angle between tangents.

Chapter 5

Spacing of data points

The focus of the previous three chapters has been on optimising the interpolation process, i.e. given data points and derivatives, selecting the most appropriate methods for obtaining the parameter values, derivative magnitudes, etc.. This chapter considers how the data points are actually obtained, as the location of the data points can have a significant impact on the quality of the resulting interpolant, and yet the subject is often overlooked. For example, points that are equally spaced along the circumference of a circle are ideal, but equally spaced points used to interpolate a GCS leads to a poor representation – Figure 5.1(a). An alternative spacing method that provides a greater density of points for regions of higher curvature is much more successful, Figure 5.1(b).

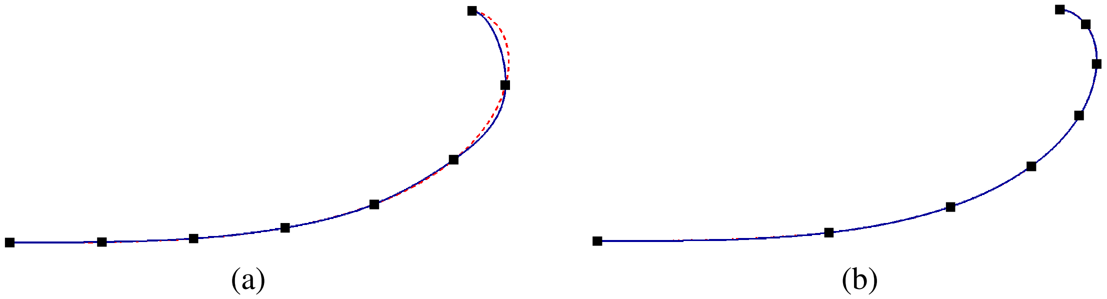


Figure 5.1 – Points sampled from a GCS (a) Equally spaced; (b) Improved spacing

This chapter assumes that points can be sampled at any location on the original geometry, which might be a defined analytically, e.g. a circle or GCS, or parametrically, e.g. a NUBS curve. Alternatively, the curve could be originally represented by a dense string of points, from which a significantly smaller subset could be retained, i.e. data reduction.

Some consideration is also given to the number of points used to represent a given curve. Increasing the number of data points, if spaced optimally, will decrease the maximum positional error between the original and the interpolant. However, over definition is undesirable, as it leads to data proliferation and can compromise follow on activities [Cripps and Lockyer, 2005].

The discussion in this chapter is restricted to the positioning of points on a planar curve, although many of the methods presented can easily be modified for general freeform curves. Points on a surface would need to be calculated as rows and columns of curves, and then average values positioned using a geometric arc length parameterisation method such as that presented by Czerkawski [1996] and improved by Chong [2006].

5.1 Background

The optimal spacing of data points has been considered in contexts other than interpolation; Pitteway [1967] describes an algorithm for plotting ellipses such that the pen, when drawing in straight line segments, deviates from the desired curve as little as possible. A similar algorithm is given by Partridge [1968] for plotting hyperbolas. Smith [1971], motivated by the limitations of computer display hardware, proposed methods for drawing ellipses, hyperbolas and parabolas using a fixed number of points. This last article is particularly relevant to the current topic.

Smith argues that, given a fixed number of points, there is an obvious distribution of those points to graphically represent a circle. Equal angle increments with a point on the perimeter for each angle leads to a satisfactory representation, assuming the number of points, *N*, is great enough. However, using the same method to produce an ellipse can fail to represent the small ends if the eccentricity is large, as shown in Figure 5.2:

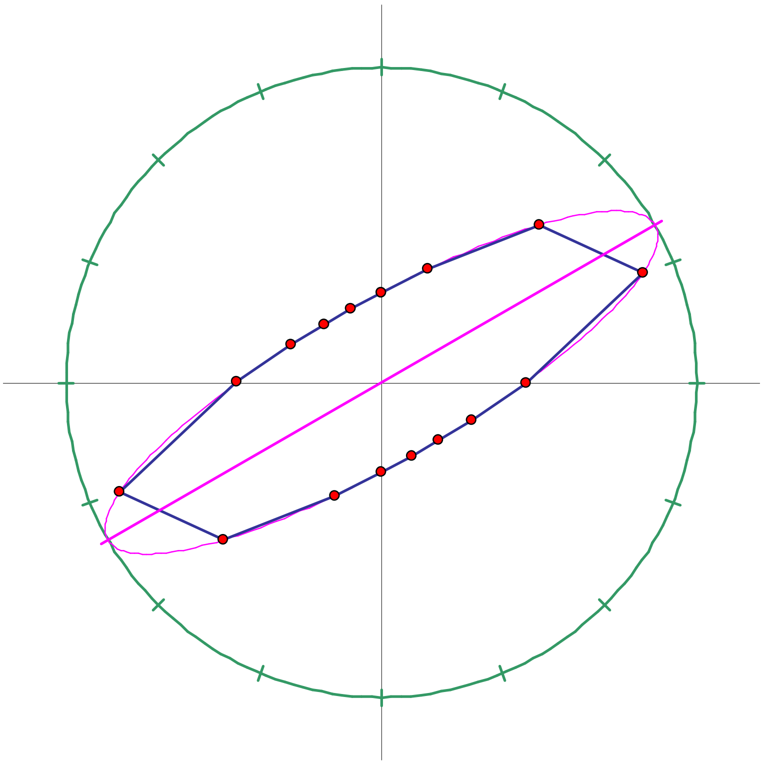


Figure 5.2 – Equal angle representation of an ellipse

Smith [1971] proposes that the ellipse be represented parametrically. An ellipse centred at the origin with axes *2a* and *2b* is given by:

$$x = a\cos\phi \tag{5.1}$$

$$y = a\sin\phi \tag{5.2}$$

where ϕ is a parameter (not an angle) that is incremented in $2\pi/(N-1)$ steps over the interval $[0,2\pi]$. An example of the resulting point spacing is given in Figure 5.3. The density of the points is greater in the regions of higher curvature. Smith proves that this point distribution method actually yields the maximum possible inscribed area for the polygon using any given *N*.

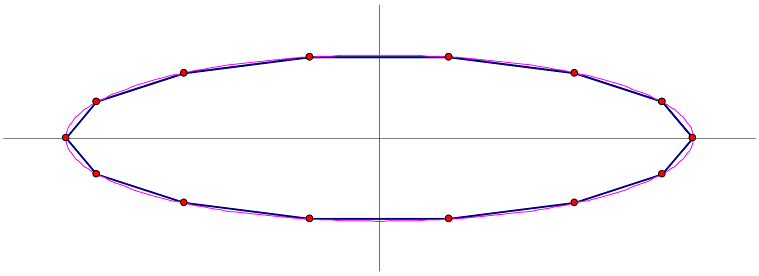


Figure 5.3 – Points spaced on the ellipse such that the inscribed area is maximum

In a similar fashion, Smith [1971] provides parametric representations for the hyperbola and parabola, again both of which produce a polygon with the maximum inscribed area. Whilst these methods have no direct application to the spacing of data points on a freeform curve, the concept of spacing points such that the inscribed area is maximum can be utilised; a method developed using this concept is presented in Section 5.3.2.

5.2 Existing methods for point spacing with freeform curves

Each of the point spacing methods presented in this chapter position data points, $\boldsymbol{Q}_k^*$, $0 \le k \le N-1$, where the superscript denotes the method used. The location of $\boldsymbol{Q}_k^*$ can also be represented by geometric parameters: the arc length, ξ_k^*, is the distance along the curve from $\boldsymbol{Q}_0^*$ to $\boldsymbol{Q}_k^*$, and the winding angle, ω_k^*, is the total winding angle through which the curve turns between $\boldsymbol{Q}_0^*$ and $\boldsymbol{Q}_k^*$. As the arc length and winding angle are geometric parameters, they are independent of the method used to define the original curve. If the point is sampled from a parametric curve, then it has an associated parameter value, i.e. $\boldsymbol{Q}_k^* = \boldsymbol{C}\left(s_k^*\right)$. There is a unique correlation between each of these representations; corresponding values can be calculated using a numerical search.

A commonly used method for spacing points along a freeform planar curve is to evenly space the data points by arc length. The arc length from $\boldsymbol{Q}_0$ to $\boldsymbol{Q}_k^{distance}$ is given by:

$$\xi_k^{distance} = \frac{k\xi_{N-1}^{distance}}{N-1}, \qquad 0 \le k \le N-2 \qquad (5.3)$$

where $\xi_{N-1}^{distance}$ is the total arc length, and can be found numerically by approximating the curve with a large number of linear segments. Rogers and Adams [1990] warn that points spaced equally by distance can lead to a poor representation when the radius of curvature decreases monotonically, as illustrated in Figure 5.1(a).

When points are evenly spaced on a circle, the winding angle between each point is constant. This suggests that points on any curve can be distributed by constant winding angle, thus providing a greater density of points in regions of higher curvature. The total winding angle from the start of the curve to $\boldsymbol{Q}_k^{angle}$ is given by:

$$\omega_k^{angle} = \frac{k\omega_{N-1}^{angle}}{N-1}, \qquad 0 \le k \le N-2 \qquad (5.4)$$

where ω_{N-1}^{angle} is the total winding angle, which can be found numerically by approximating the curve with a large number of linear segments, and summing the angle between consecutive segments. Figure 5.4 illustrates a GCS interpolating points sampled by constant winding angle.

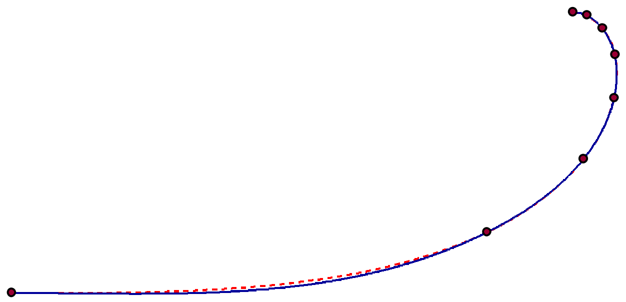


Figure 5.4 – Points sampled from a GCS by constant winding angle

It is evident, however, that the greater density of points at one end of the GCS seriously compromises the shape at the other end where there are now insufficient points.

Both Smith [1971] and Rogers and Adams [1990] advocate that the point distribution should 'reflect' the curvature, although neither suggest any practical way of implementing this. Where the curvature of the original curve monotonically increases, e.g. like a GCS, then implementing a linear relationship between the density of the points and the value of curvature is possible – see Figure 5.5. It is clear that such a direct relationship produces very poor results when the curvature variation is large, as the curve is seriously under-represented in the region of low curvature. Even despite the poor results, implementing this as a general point spacing method is troublesome: many curve definitions will have fluctuating and complex curvature profiles, or are defined by a dense string of points and, subsequently, have no explicit curvature definition. As a consequence, no further consideration is given to points spaced linearly with curvature in this chapter. It is therefore desirable that any new methods are sensitive to variations in curvature, but be formulated in terms of position and tangent information.

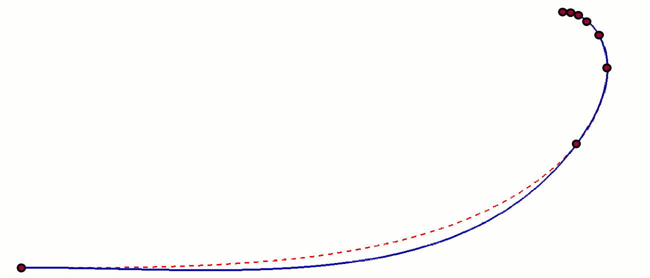


Figure 5.5 – Points sampled from a GCS by a constant curvature variation

5.3 New methods for point spacing

5.3.1 Weighted average of distance and angle

The first new method to be proposed is a 'quick and easy' method, and is produced by taking a weighted average of the methods that space points equally by distance and angle. The average is arithmetic, and applies to the geometric arc length from the start of the curve to the *k*th points, produced by the two methods:

$$\xi_k^{weighted} = \frac{1}{3}\xi_k^{distance} + \frac{2}{3}\xi_k^{angle} \qquad 0 \le k \le N-1 \qquad (5.5)$$

where ξ_k^* is the arc length distance from $\boldsymbol{Q}_0^*$ to $\boldsymbol{Q}_k^*$, and the superscript denotes the point spacing method. The weightings in (5.5) were obtained heuristically. Figure 5.6 illustrates the point spacing method on a GCS. The density of the points increase with curvature, as when spaced by angle, but the 'flatter' region is more controlled, as when spaced by distance.

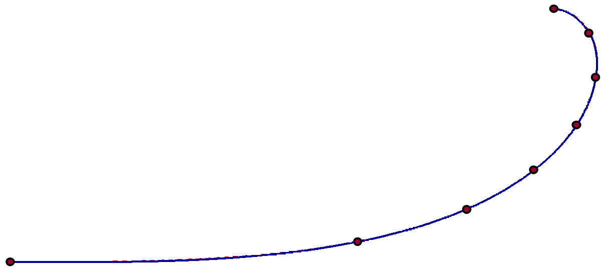


Figure 5.6 – Points sampled from a GCS by a weighted average of distance and angle

5.3.2 Maximum inscribed area

Another method is proposed that is motivated by the work of Smith [1971], who distributed points on ellipses, hyperbolas and parabolas such that the inscribed area was a maximum. This concept can be applied to the distribution of points on a freeform curve. Smith proves that a polygon inscribed in a convex curve has maximum area when, for every three consecutive points, the tangent at the middle point parallels the chord between the other two points. This is illustrated in Figure 5.7; it is intuitive because the maximum area of the triangle occurs when the height, perpendicular to the chord, is greatest. Because the curve is convex, the maximum height must occur when the tangent is parallel to the chord.

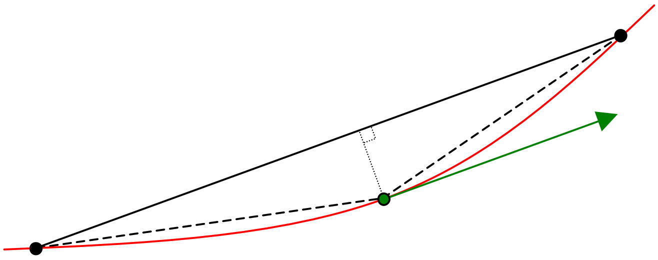


Figure 5.7 – Condition for maximum inscribed area of convex polygon

The location of the points on the curve can be repositioned iteratively such that:

$$\left\| \left(\boldsymbol{C}\left(s_{k+1}\right) - \boldsymbol{C}\left(s_{k-1}\right) \right) \times \boldsymbol{C}_u\left(s_k\right) \right\| = 0 \qquad 1 \le k \le N-2 \qquad (5.6)$$

It is noted that this method can only be used where the curve is strictly convex, i.e. contains no inflections, although the curve could be split into sections with monotonic curvature, and the method applied to each section individually. However, the location of the points between sections may not be ideal, and could result in some sections being under-specified; it is therefore not recommended. The method is illustrated in Figure 5.8 for convex data.

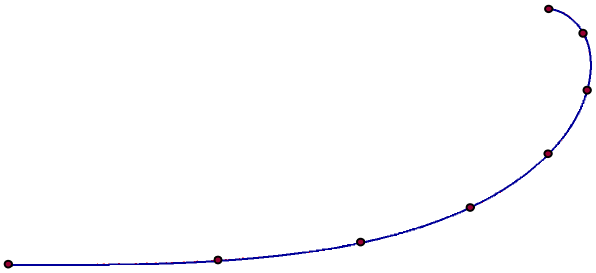
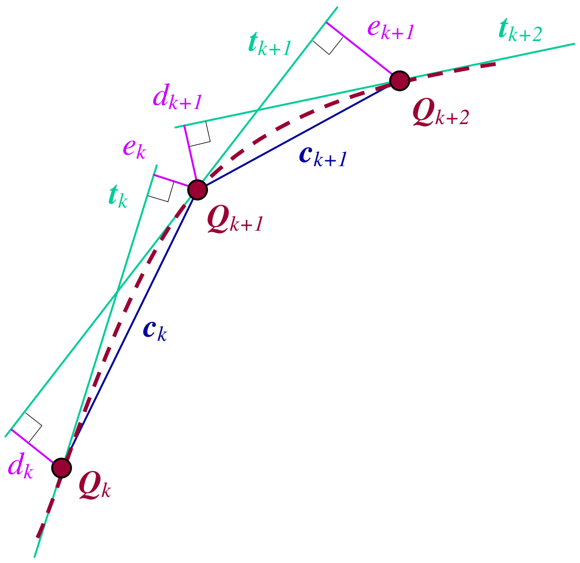


Figure 5.8 – Points sampled from a GCS by the maximum inscribed area method

5.3.3 Constant projected distance

A third new method is proposed, that aims to maintain as constant the sum of the projected distances for each segment, see Figure 5.9:



$\boldsymbol{Q}_k$ Data points, $0 \le k \le N-1$

$\boldsymbol{c}_k$ Chord, $(\boldsymbol{Q}_{k+1} - \boldsymbol{Q}_k)$

$\boldsymbol{t}_k$ Unit tangent to curve at $\boldsymbol{Q}_k$

d_k Projected distance between chord $\boldsymbol{c}_k$, and tangent $\boldsymbol{t}_{k+1}$ at $\boldsymbol{Q}_k$

e_k Projected distance between chord $\boldsymbol{c}_k$, and tangent $\boldsymbol{t}_k$ at $\boldsymbol{Q}_{k+1}$

Figure 5.9 – Construction of the projected distances

The method requires the sum of the projected distances for each segment to be constant, i.e.:

$$\left(d_k + e_k\right) = \left(d_{k+1} + e_{k+1}\right) \qquad 0 \le k \le N-2 \tag{5.7}$$

where:

$$d_k = \left\|\boldsymbol{c}_k \times \boldsymbol{t}_{k+1}\right\| \tag{5.8}$$

$$e_k = \left\|\boldsymbol{c}_k \times \boldsymbol{t}_k\right\| \tag{5.9}$$

Conceptually, requiring $\left(d_k + e_k\right)$ to be a constant ensures that regions of higher curvature receive a greater density of points because the angle between the chord and tangent will be large. However, regions of low curvature are less likely to be under-represented because $\left(d_k + e_k\right)$ increases with the length of the chord. The method therefore creates a balance between the point density and curvature. The spacing of data points on a GCS using the constant projected distance (CPD) method is illustrated in Figure 5.10.

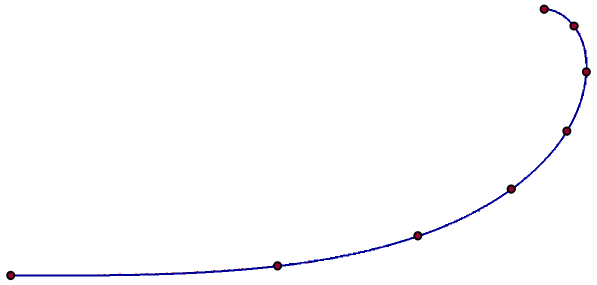


Figure 5.10 – Points sampled from a GCS by the constant projected distance method

The points need to be iteratively repositioned to satisfy condition (5.7). The process is well behaved when the curve is convex: moving point $\boldsymbol{Q}_{k+1}$ further along the curve has the effect of increasing both d_k and e_k, and reducing both d_{k+1} and e_{k+1}. This is not true in the region of an inflection, but numerical evidence suggests that $\left(d_k + e_k\right)$ will still increase, and $\left(d_{k+1} + e_{k+1}\right)$ decrease, so a numerical solution is always possible.

The CPD method tends to under-represent regions that contain an inflection. Consider the curve in Figure 5.11:

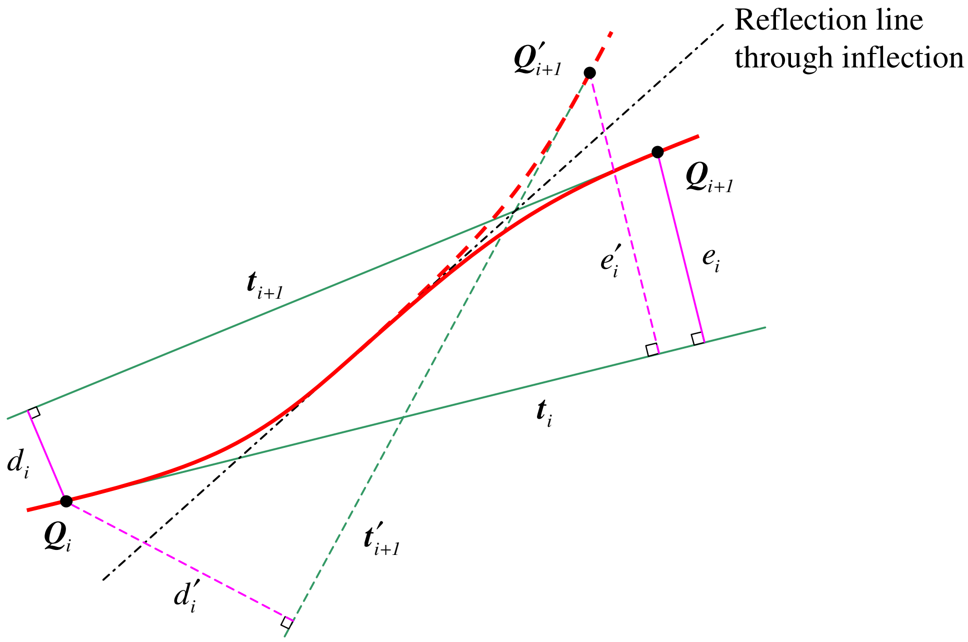


Figure 5.11 – Construction of CPD method when the curve inflects

The curve inflects somewhere between $\boldsymbol{Q}_i$ and $\boldsymbol{Q}_{i+1}$; it is possible to reflect the curve beyond the inflection point such that the resulting curve has the same absolute curvature, but remains convex. One might expect that the distribution of points on these two curves should be mirrored in the reflection line, but Figure 5.11 illustrates that the sum of the projected distances is very different for the corresponding $\boldsymbol{Q}_{i+1}$ and $\boldsymbol{Q}'_{i+1}$ points. Without reflecting the curve, the inflection causes the projected distances d_i and e_i to under-represent the extent of the local change in shape.

An inflecting version of the CPD method, ICPD, therefore checks to see if two consecutive points bound an inflection, and reflects the curve in the tangent at the inflection point if they do. The parametric location of the reflected point, $\boldsymbol{Q}'_{i+1}$, that satisfies (5.7) is used to obtain the corresponding point on the true curve, $\boldsymbol{Q}_{i+1}$. The difference between the CPD and ICPD methods is given in Figure 5.12. It is anticipated that this method will perform well when the rate of change in curvature is large.

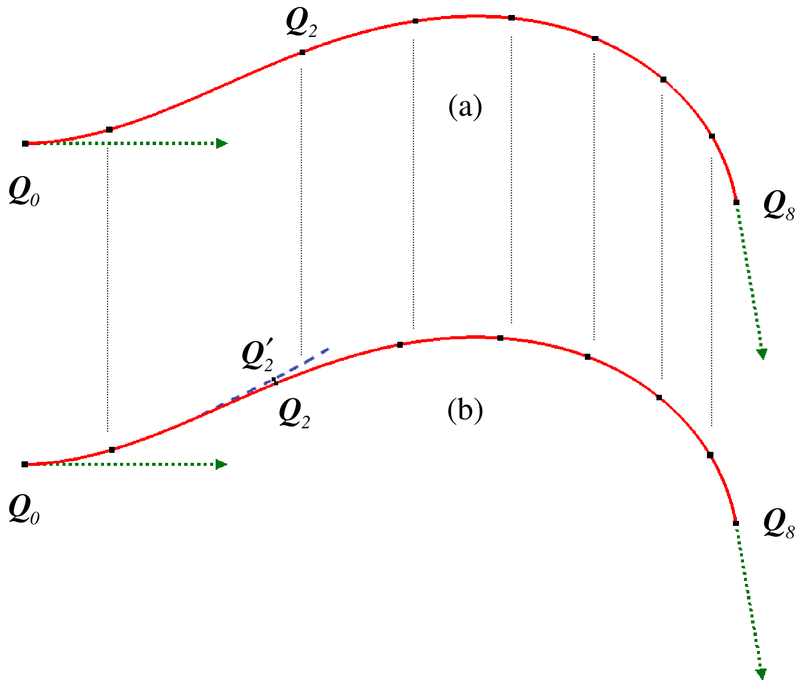


Figure 5.12 – Inflecting GCS curve spaced using (a) CPD, (b) Inflecting CPD methods

5.4 Numerical testing of spacing methods

The performance of the spacing algorithms is assessed numerically by sampling and interpolating points from three different curves: a non-inflecting GCS, an inflecting GCS, and a damped oscillatory function. There would be no benefit in sampling points from a circle, as each of the methods reviewed would equally space the points. In all cases, the points will be interpolated using the $W = 1$ orthogonal parameterisation method, following the conclusions of Chapter 4, with the derivative magnitudes that force orthogonality at the ends. The performance of the algorithms will be assessed in terms of the maximum absolute positional error between the interpolated curve and the original definition, calculated using a geometric parameterisation approach (see Chapter 4). Relative curvature error is not used as a measure of performance, because two of the case studies contain inflections, causing the relative curvature measure to become less meaningful and possibly misleading.

5.4.1 Non-inflecting GCS

The first case study requires points to be sampled from a non-inflecting GCS profile, which is defined as follows:

Total arc length	S	1.393465
Shape factor	r	-0.952602
Start curvature	κ_0	0.000002
End curvature	κ_1	20.587628

Table 5.1 – Case study 1 GCS definition

The minimum number of points that should be used to interpolate any given curve is discussed in Section 5.5; following the recommendations in that section, 8 points are used to interpolate the GCS curve. The point distributions and maximum positional errors for each point spacing method are illustrated in Figure 5.13.

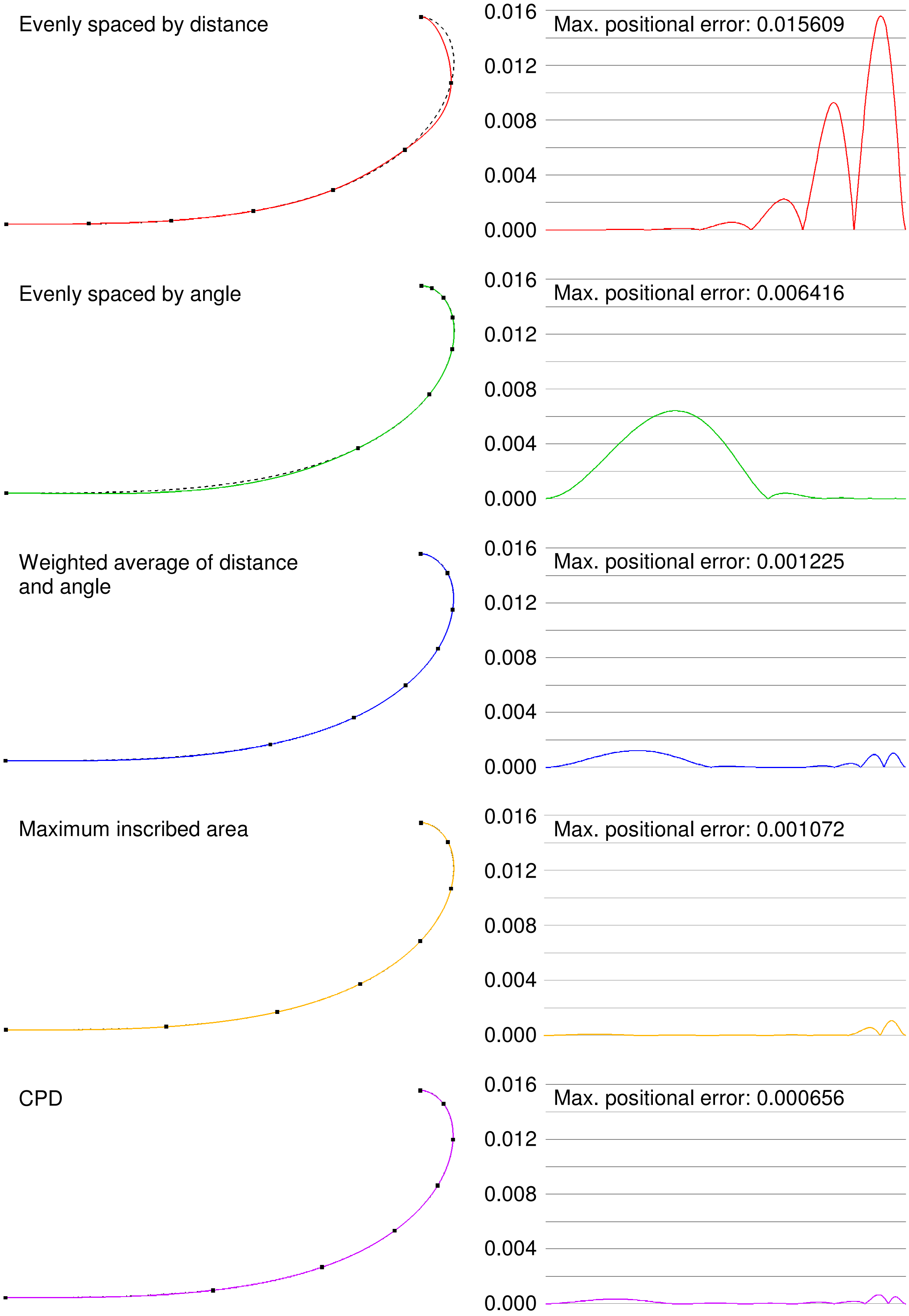


Figure 5.13 – Case study 1 – point distributions on a non-inflecting GCS

It is evident from Figure 5.13 that points spaced evenly by distance or angle do not perform well; distance based points do not account for any change in curvature, whereas angle based points overcompensate in the region of increased curvature. As the weighted average method balances these two distributions, it does perform well; the heuristically defined weighting factor is seen to be appropriate as the positional error is approximately equal at either end of the GCS.

The maximum inscribed area method performs very well, although the CPD method is better still, having a slightly reduced maximum error in the region of highest curvature. The maximum inscribed area method has a smaller error at the beginning of the curve, but reducing the error in one region of the curve often means the error in another region will be larger. It is noted that the inflecting CPD method produces an identical result to the CPD method as the curve does not contain an inflection.

5.4.2 Inflecting GCS

The second case study requires points to be sampled from an inflecting GCS profile, which is defined in Table 5.2; 10 points are used following the guidelines in Section 5.5.

Total arc length	S	24.0
Shape factor	r	-0.58
Start curvature	κ_0	0.16
End curvature	κ_1	-0.65

Table 5.2 – Case study 2 GCS definition

It is noted that the maximum inscribed area method is not used because it only operates with convex data – see Section 5.3.2. The point distributions and maximum positional errors for each point spacing method are illustrated in Figure 5.14.

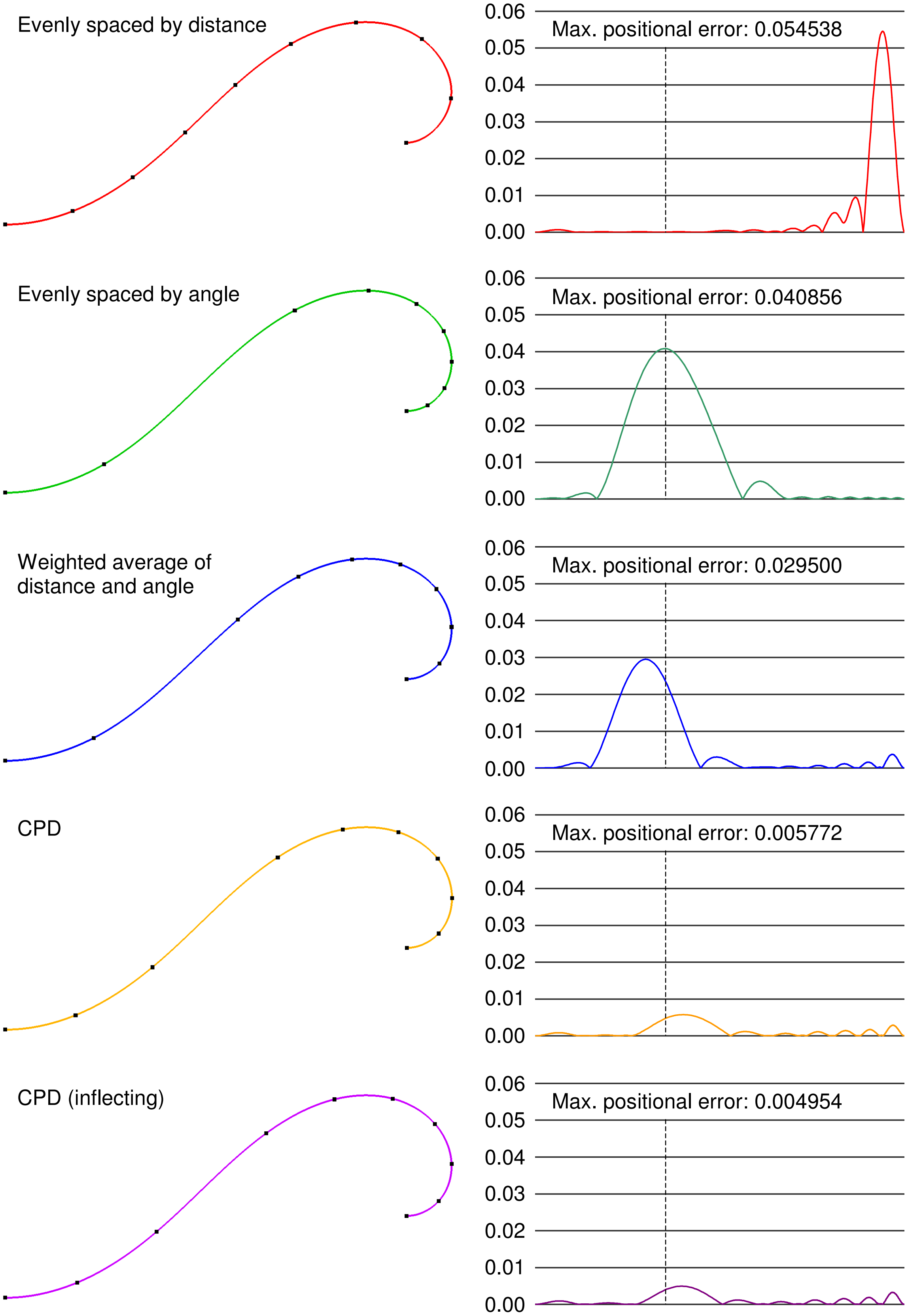


Figure 5.14 – Case study 2 – point distributions on an inflecting GCS

Again, points equally spaced by distance are completely insensitive in regions with larger curvature, and therefore perform badly. All of the other methods have fewer points in the region of the inflection, allowing a greater density of points where the curvature is greater, hence the error distributions are similar. It is apparent that the points equally spaced by angle are a very poor representation of the curve in the inflecting region, as the angular variation is minimal. Whilst the weighted average method has a smaller maximum positional error compared with the previous two methods, it still does not perform very well around the inflection. The weighting factor is seen to bias the resulting point distribution too much towards the angle spaced points, indicating that there is no 'ideal' weighting factor.

The CPD method performs very well over the entire curve, having a significantly reduced maximum positional error. The ICPD method slightly outperforms the standard method because the curve inflects, causing the density of points to be increased in the region of the inflection. The difference is only small because the angular variation in the region of the inflection is small, as expected.

5.4.3 Damped oscillatory function (data reduction)

The final case study has two aims; firstly, it tests the point spacing algorithms on more 'exotic' data, which is irregular, contains large variations in curvature, and has multiple inflections. The second aim is to demonstrate how the algorithms can be used for data reduction.

The data is generated using the damped oscillatory function:

$$y = 1 - e^{-\frac{1}{2}x}\cos\left(\frac{1}{2}\sqrt{3}x\right) + \frac{19}{\sqrt{3}}e^{-\frac{1}{2}x}\sin\left(\frac{1}{2}\sqrt{3}x\right), \quad 0 \le x \le 12 \qquad (5.10)$$

where points are sampled $x = 0.0, 0.1, \ldots, 11.9, 12.0$. This creates a set of 121 points, Figure 5.15, which is significantly greater than the number of points required to characterise the geometric shape. From this data set, the algorithms are used to select a subset of 15 points, following the general guidance given in Section 5.5, which are then used to interpolate the curve.

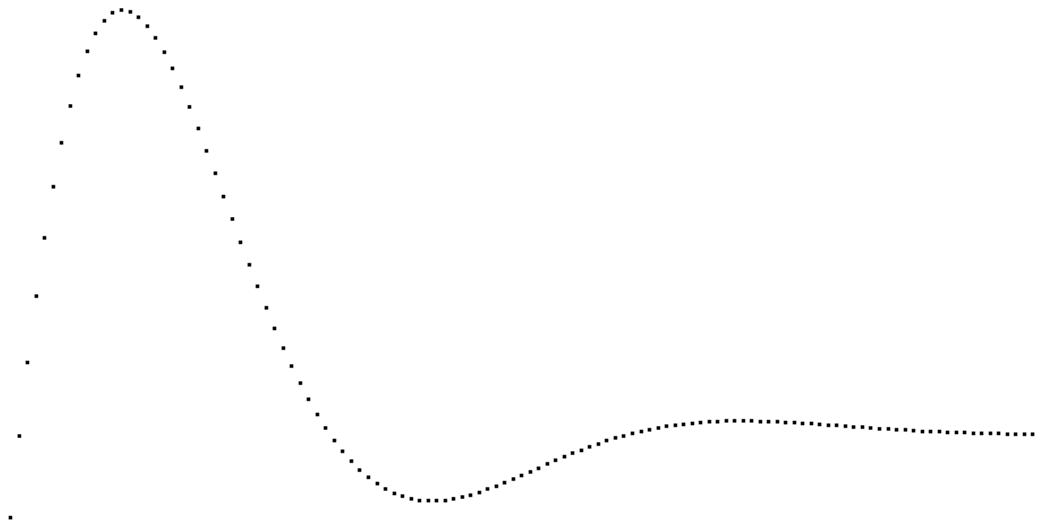


Figure 5.15 – Case study 3 - points used for data reduction

The point distributions and maximum positional errors for each point spacing method are illustrated in Figure 5.16. Again, the maximum inscribed area method is not used, as the data contains three inflections – see Section 5.3.2.

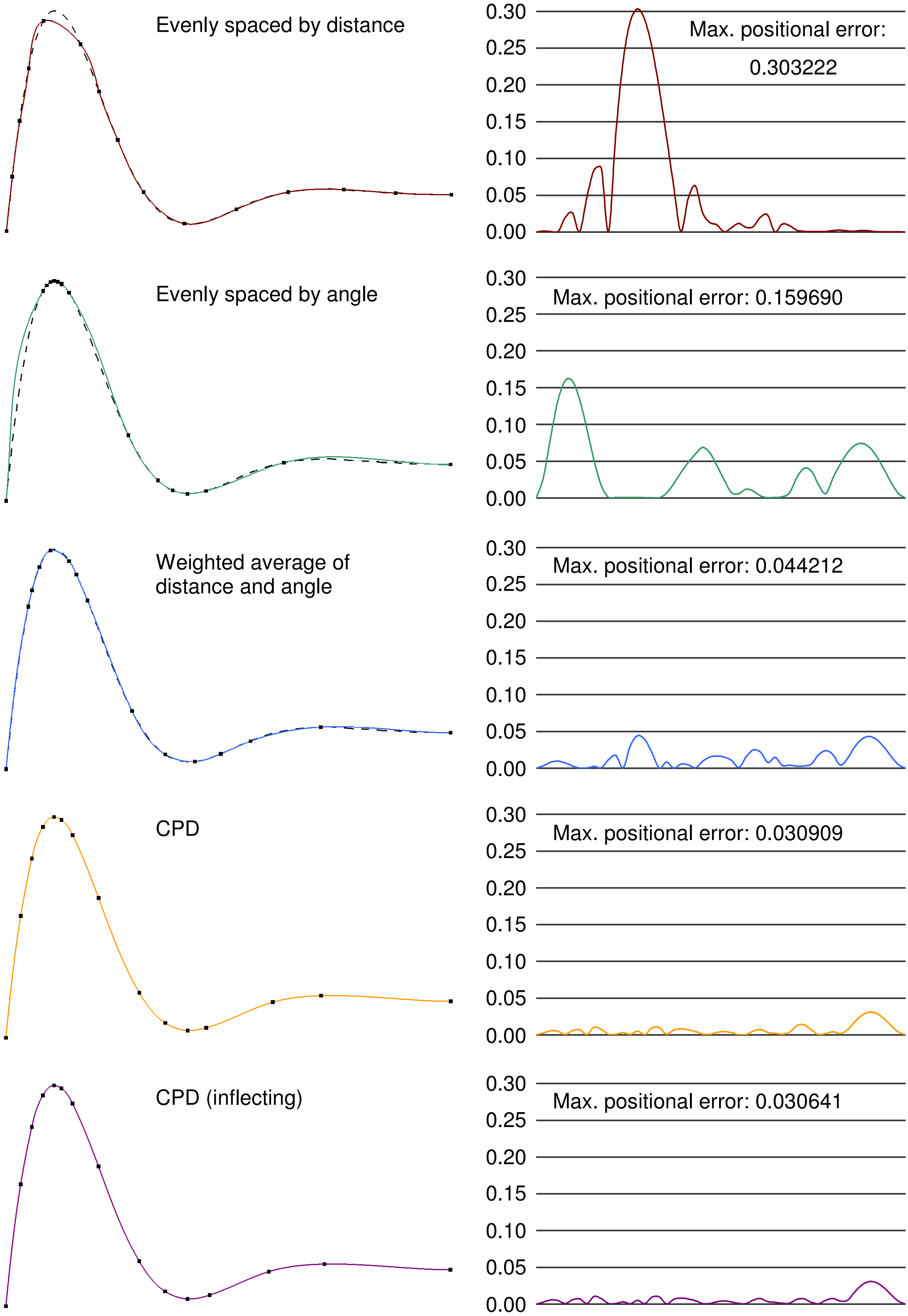


Figure 5.16 – Case study 3 – data reduction

The interpolants passing through the points spaced by constant distance or winding angle are once again poor; the constant distance method does not provide a sufficient density of points in the regions of high curvature, and the constant angle method under-specifies regions with low curvature. The weighted average method provides a much improved representation of the data, although the CPD methods produce the best results. The inflecting CPD method has only a marginal performance advantage over the standard method. This is because the CPD method placed data points very near to the first two inflection points of the curve, which effectively makes the algorithms perform identically in those regions. The ICPD method reduces the error near to the third inflection point, but the angular variation in this region is small, causing the difference to be minimal.

5.4.4 Selection of spacing methods

The case studies highlight that the existing methods of spacing points by constant distance or constant angle will often result in a poor representation of the original curve. A non-iterative method takes the weighted average of these methods, and usually yields improved results. The maximum inscribed area method was seen to outperform the weighted average method, although is only applicable to data that is strictly convex.

The constant projected distance methods were seen to outperform the other methods in all of the experiments conducted; they place points in such a way that regions of high curvature have a higher density of points, as the angle between chord and tangent is large, and regions of low curvature are not under-represented, as the projected distances are proportional to the chord between data points. The ICPD method was seen to have a marginal performance advantage over the CPD method because it treats any segments with inflections as if the curve were convex.

5.5 Number of data points

It can be expected that in general, increasing the number of points will reduce the error between the original and interpolated curves, and likewise decreasing the number of data points will increase the error. Generally, one should select the number of points that is necessary to achieve the required accuracy for the given application. For example, automotive industries require very high tolerances for A-class surfaces such as exterior car panels, whereas toolmakers for the casting industry will probably use more relaxed tolerances [Cripps, 2003]. Using significantly more points than required will usually guarantee tolerances are met, although Cripps and Lockyer [2005] warn that this can lead to data proliferation, which has adverse implications for data processing, management and manipulation. In this section, a 'guideline' for the minimum number of points that will yield an acceptable result is presented.

Cripps [2003] reports that a relative curvature tolerance of 5% is sufficient to satisfy the exacting tolerances required by the aeronautical and automotive industries. Motivated by this, Cripps and Lockyer [2005] propose a heuristic rule for the number of data points required to interpolate a circular arc using the circle orthogonal parameterisation:

$$N = 1 + \mathrm{int}\left(\frac{6\omega}{\pi}\right) \tag{5.11}$$

where ω is the swept angle of the circular arc.

The rule is unaffected if the angle, ω, is considered as the total winding angle of the curve, allowing it to be applied to general freeform curves. An investigation consisting of seven GCS curves was used to test the applicability of this rule to data with more complex curvature profiles than a circle. The GCS definitions are given in Table 5.3:

Total arc length	S	3.141
Shape factor	r	1.0
Start curvature	κ_0	1.0
End curvature	κ_1	0.1, 0.3, 0.5, 0.7, 1.0, 1.4, 2.0

Table 5.3 – GCS definitions for ascertaining the optimum number of points

The seven GCS curves and their corresponding curvature profiles are illustrated in Figure 5.17:

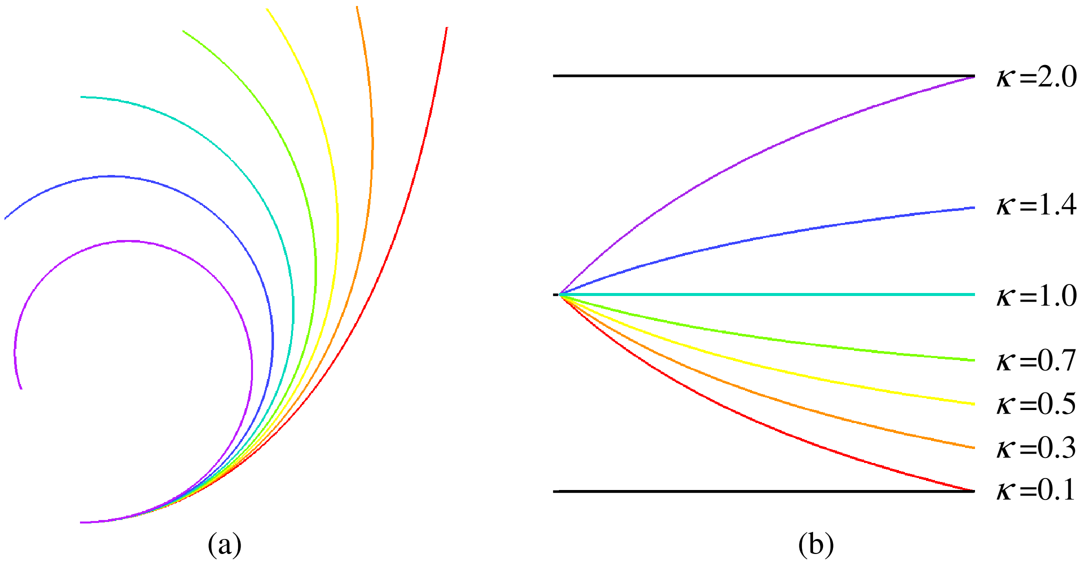


Figure 5.17 – (a) GCS curves and (b) curvature profiles, used in determining the optimum number of points

The investigation established that (5.11) was generally applicable to all of the GCS curves, although it tended to over-specify the number of points required. A more appropriate rule is:

$$N = 1 + \text{int}\left(\frac{5\omega}{\pi}\right) \tag{5.12}$$

The majority of circular arcs, $\pi/6 \leq \omega \leq 2\pi$, interpolated using (5.12) and the circle orthogonal parameterisation still yield a relative curvature error $< 5\%$. When (5.12) is applied to the seven GCS curves specified in Table 5.3, and the points spaced using the constant projected distance method, the curvature errors are displayed in Table 5.4. Curvature errors are used following Cripps and Lockyer [2005].

End Curvature, κ_1	**Winding angle,** $\omega°$	**Number of points,** N	**Relative curvature error with** N**,** %	**Relative curvature error with** N-1**,** %
0.1	80.6	3	11.670424	149.887082
0.3	102.7	4	4.707136	10.627244
0.5	124.7	5	3.147426	5.869782
0.7	146.8	5	3.942306	7.487661
1.0	180	6	3.517725	5.716739
1.4	224.1	7	4.296107	6.421410
2.0	290.4	9	4.471289	6.006655

Table 5.4 – Results from interpolating seven GCS curves with (5.12)

The table lists the number of points specified by (5.12) for each of the GCS profiles, and the corresponding relative curvature errors; the table also gives the relative curvature errors when *N* is reduced by one. It is evident that, for all cases, when *N* is less than the value given by (5.12), the relative curvature error is greater than 5%. Conversely, when *N* is given by (5.12), the relative curvature error is less than 5%, with the exception of the $\kappa_1 = 0.1$ case. This indicates that (5.12) is an appropriate guideline for the minimum number of points to use for a given interpolation.

The $\kappa_1 = 0.1$ case has a higher curvature error than anticipated; this is caused by the low value of the absolute curvature, as it is approaching an inflection, and causes the relative curvature error to become large. If each of the GCS profiles were interpolated using exactly six points, one would intuitively expect the relative curvature error to decrease as the winding angle decreased. This is illustrated in Figure 5.18. However, the relative curvature error is seen to rise for GCS with the smallest end curvature, because κ_1 is small. In contrast, the absolute positional error is seen to continue diminishing with decreasing winding angle.

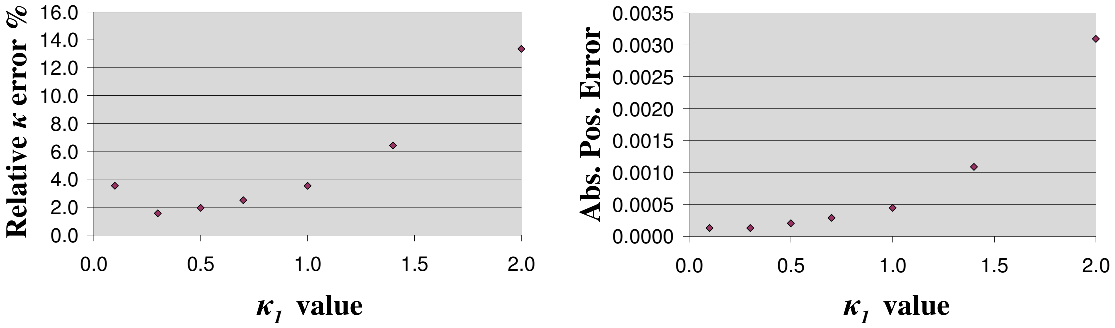


Figure 5.18 – Comparison of relative curvature error with absolute positional error

(5.12) applies to strictly convex curves. For inflecting curves, it should be applied to each convex section; the total number of points required for the interpolation is then the sum of those required for each section, minus one for each boundary between segments. Where the curve contains large variations in curvature, the number of points should be increased by one per convex section.

5.6 Summary

This chapter has considered the problem of how points should be distributed to minimise the resulting error following interpolation, and the related problem of specifying the minimum number of points required to satisfy the tolerances generally required within industry. It was assumed that the points can be obtained from the original geometry, which can be defined analytically, parametrically, or as a dense string of points.

Two conventional methods were considered that spaced points either by constant arc length, or by constant winding angle. It was discovered that the former was insensitive to changes in curvature, under-representing areas of high curvature, whereas the latter over-compensated. A new method was proposed that takes a weighted average of these two approaches, and was found to yield improved results.

Motivated by the work of Smith [1971], a second method was developed for convex curves, which distributes data points such that the inscribed area is maximal. This method also gave good results, but requires the solution to be found by iteration, and also can only be applied to strictly non-inflecting curves. The original geometry can be split into sections having monotonic curvature variation, applying the algorithm to each section individually; however, this approach forces data points to be placed at inflection points, which can result in some sections being under-specified.

A third method was proposed that aims to keep the sum of the projected distances constant throughout the curve. The method also requires iteration to converge upon a solution, but the results were found to be excellent, outperforming all other methods in all the tests conducted. A version of the method was presented that is slightly more sensitive to curves that inflect, although the original approach is not restricted to non-inflecting curves.

The second aspect of the problem was determining the number of data points to use for any given interpolation. It was concluded that the number of points required is application dependent, although a heuristic rule was suggested that relates the number of data points required for interpolation to the winding angle of the curve. This rule was seen to be appropriate for a number of different GCS curves.

It is evident that the choice of a point spacing method, and the number points used, has a significant effect on the quality of the resulting interpolant. The constant projected distance method is recommended for distributing data points; the number of points used should reflect the application, but a reasonable guideline is that an additional point should be used for every 36° change in winding angle.

Chapter 6

Twist vector estimation

Chapter 1 highlighted the need for four mixed partial derivatives, or twist vectors, when interpolating a constrained NUBS surface. Generally, these twist vectors need to be estimated, although being second derivatives they are not related to the geometry of the surface, rather its parameterisation. Several methods for estimating the twist vectors were reviewed in Chapter 2, and it was concluded that the Adini method [Barnhill, *et. al.*, 1978] was the best existing algorithm, despite the fact that its formulation contains the naïve assumption that all general cubic patches can be well represented using bilinearly-blended Coons patches.

This chapter presents a new twist vector estimation method that builds on the Adini method. It is shown that, for quadratic data, the true twist can be exactly recreated using the Adini twist and a bilinearly-blended Coons patch. The method gives excellent results for general cubic surfaces, and the resulting error is shown to diminish at a faster rate than Adini when the data is subdivided. Numerical tests reinforce that the Improved method consistently outperforms Adini.

6.1 Improved twist vector estimation method

6.1.1 Adini

If a bi-cubic surface is constructed by constrained interpolation with $(m-1)\times(n-1)$ data points, it can be exactly represented with $(m-2)\times(n-2)$ bi-cubic sub-patches using, for example, the Bézier or Ball basis. However, the four central control vertices of each sub-patch are influenced by the four corner twist vectors of the interpolated surface, which are generally not known. Recalling from Chapter 2, the Adini method [Barnhill, *et. al.*, 1978] represents the sub-patches using bilinearly-blended Coons patches, which are entirely defined by their twelve external vertices. As a consequence, the required four twist vectors can be sampled from the four corner Coons patches, and scaled according the parametric intervals of the surface ((2.50) – (2.51)). If the bilinearly-blended Coons patches are a good approximation of the 'true' surface, the four sampled twist vectors will also be good. However, where this is not the case, the twist vectors produced by Adini may be unacceptable.

When a surface is constructed using data points and cross boundary derivatives, it is impossible to know what the 'true' surface is, as it is a function of the twist vectors. It is possible to visually compare the performance of different twist vector estimation algorithms by considering the Gaussian curvature of the surfaces [Barnhill, *et. al.*, 1978], although this is a subjective approach. The performance of a twist vector estimation algorithm can be numerically assessed by sampling points, cross-boundary derivatives and twist vectors from an existing surface, and using the algorithm to estimate the twist vectors; the results can be compared to the sampled vectors to give a numerical measure of performance.

6.1.2 Hypothesis

The Adini twist [Barnhill, *et. al.*, 1978], when applied to all four corners of a single bicubic patch, produces a Coons patch by definition. When an Adini twist is used at the corner of a multi-patch interpolation, the resulting corner sub-patch is not identical to a Coons patch. This is because the twist vectors at the other three corners of the sub-patch do not, in general, correspond to Adini. There is however a relationship between the mid-point of this patch, and the mid-point of the corresponding Coons patch. It is hypothesised that, for quality surfaces [Cripps and Ball, 1998], there is an approximate linear relationship between these two mid-points and the mid-point that lies on the 'true' surface. This hypothesis implies that there is an approximate linear relationship between the corresponding twist vectors, which can be expressed in the following form for one corner and is the crux of the improved algorithm:

$$\boldsymbol{S}_{uv[A]}\left(0,0\right)=\left(1-\lambda\right)\boldsymbol{S}_{uv[C]}\left(0,0\right)+\lambda\boldsymbol{S}_{uv[T]}\left(0,0\right),\ 0\le\lambda\le 1 \qquad (6.1)$$

where $\boldsymbol{S}_{uv[A]}\left(0,0\right)$ is the Adini twist, $\boldsymbol{S}_{uv[T]}\left(0,0\right)$ the true twist, and $\boldsymbol{S}_{uv[\mathrm{C}]}\left(0,0\right)$ the twist that would cause the interpolated surface to pass through the parametric centre of the locally interpolated Coons patch, $\boldsymbol{C}^{0,0}\left(½,½\right)$. $\boldsymbol{S}_{uv[\mathrm{C}]}\left(0,0\right)$ cannot be calculated directly, but needs to be obtained numerically. Since, from linear algebra, the vector difference between any two given twist vectors is proportional to the vector difference between the parametric mid-points of their corresponding corner patches:

$$\boldsymbol{S}_{uv[\mathrm{C}]}\left(0,0\right)=\boldsymbol{S}_{uv[\mathrm{A}]}\left(0,0\right)+\gamma\left(\boldsymbol{C}^{0,0}\left(½,½\right)-\boldsymbol{B}^{0,0}_{[\mathrm{A}]}\left(½,½\right)\right) \qquad (6.2)$$

where γ is the scale factor of proportionality. The value of γ can be obtained by rearranging (6.2), and substituting for known vector quantities, e.g.:

$$\gamma=\frac{\left\|\boldsymbol{S}_{uv[\mathrm{Z}]}\left(0,0\right)-\boldsymbol{S}_{uv[\mathrm{A}]}\left(0,0\right)\right\|}{\left\|\boldsymbol{B}^{0,0}_{[\mathrm{Z}]}\left(½,½\right)-\boldsymbol{B}^{0,0}_{[\mathrm{A}]}\left(½,½\right)\right\|} \qquad (6.3)$$

where $[Z]$ indicates that the surface has been interpolated with zero twists.

The linear relationship (6.1) is illustrated graphically in Figure 6.1 for a cubic NUBS interpolation with open clamped knot vectors.

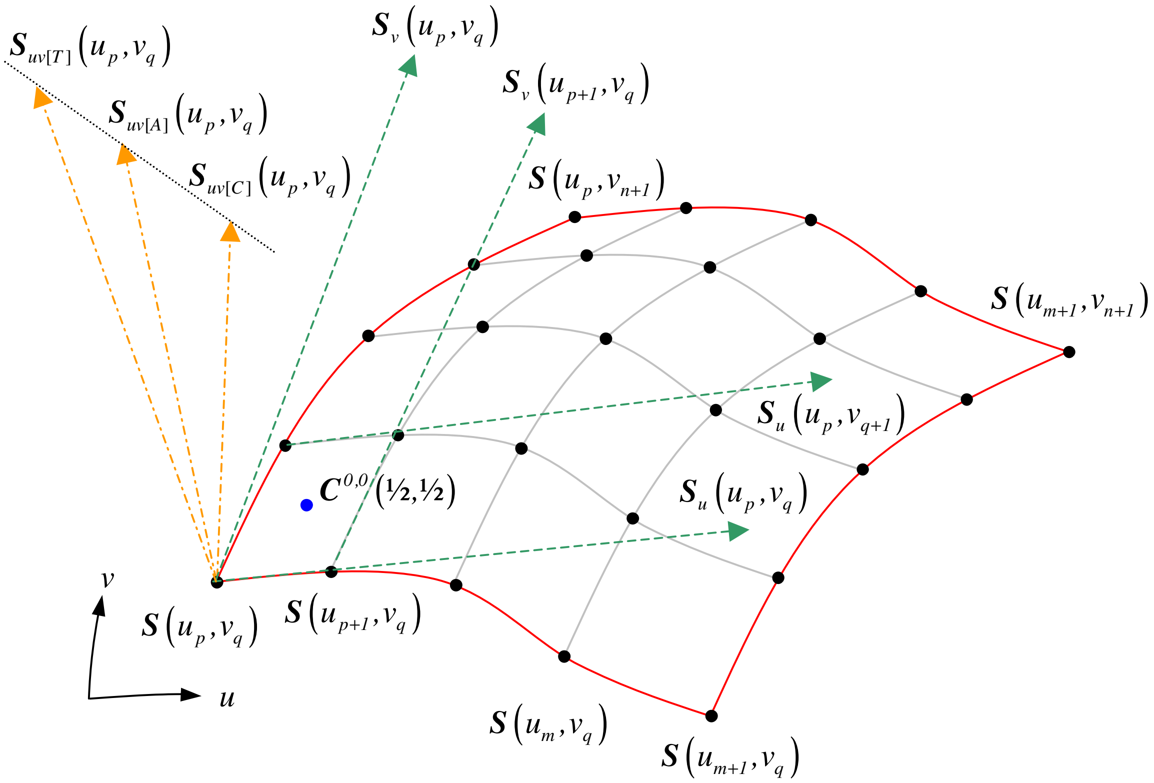

$\boldsymbol{S}_{uv[T]}(u_p,v_q)$
$\boldsymbol{S}_v(u_p,v_q)$
$\boldsymbol{S}_{uv[A]}(u_p,v_q)$
$\boldsymbol{S}_v(u_{p+1},v_q)$
$\boldsymbol{S}_{uv[C]}(u_p,v_q)$
$\boldsymbol{S}(u_p,v_{n+1})$
$\boldsymbol{S}(u_{m+1},v_{n+1})$
$\boldsymbol{S}_u(u_p,v_{q+1})$
$\boldsymbol{C}^{0,0}(½,½)$
$\boldsymbol{S}_u(u_p,v_q)$
v
$\boldsymbol{S}(u_p,v_q)$
$\boldsymbol{S}(u_{p+1},v_q)$
u
$\boldsymbol{S}(u_m,v_q)$
$\boldsymbol{S}(u_{m+1},v_q)$

Figure 6.1 – NURBS surface interpolation showing the relationship between twists

6.1.3 Method implementation

The algorithm takes as input a grid of points suitable for a constrained interpolation, corresponding cross-boundary derivatives, and both knot vectors. It has three discrete stages:

i. A bilinearly-blended Coons patch is calculated in each corner of the surface, from which the Adini twists are obtained.

ii. Twist vectors are then calculated that force the interpolated surface to pass through the mid-points of each of the four corner Coons patches using (6.2). This involves calculating the linear factor that relates a change in the twist to a change at the mid-point on the corner sub-patch (6.3).

iii. Lastly, the linear relationship (6.1) can then be used to yield the estimated 'true' twist. The value of λ is given by (6.21) in Section 6.2.4.

6.2 Analytical validation of the hypothesis

To establish (6.1) mathematically, it is necessary to consider analytical expressions for each of the vector quantities. Since the mid-point of any interpolated patch is a linear function of its points, derivatives and twist vectors, (6.1) can be reformulated, i.e.:

$$\boldsymbol{B}^{0,0}_{[A]}(½,½) = (1-\lambda)\boldsymbol{C}^{0,0}(½,½) + \lambda \boldsymbol{B}^{0,0}_{[T]}(½,½) \tag{6.4}$$

The three points, $\boldsymbol{B}^{0,0}_{[A]}(½,½)$, $\boldsymbol{C}^{0,0}(½,½)$ and $\boldsymbol{B}^{0,0}_{[T]}(½,½)$ therefore need to be expressed analytically for the hypothesis to be verified.

6.2.1 True mid-point

To simplify the analysis, the mid-point of a general bi-cubic patch can be expressed in terms of the cubic Ball basis [Ball, 1974]:

$$\boldsymbol{B}^{0,0}_{[T]}(½,½) = \frac{1}{16}\sum_{i=0}^{3}\sum_{j=0}^{3}\boldsymbol{P}^{0,0}_{i,j} \ , \ 0 \le i,j \le 3 \tag{6.5}$$

6.2.2 Adini mid-point

The mid-point of a patch that is part of a surface interpolated using the Adini twist $\boldsymbol{S}_{uv[A]}(0,0)$ can also be expressed analytically using the Ball basis. Initially, consider the corner bi-cubic sub-patch $\boldsymbol{B}^{0,0}_{[\mathrm{T}]}(u,v)$, which is part of an interpolated surface that contains 2×2 sub-patches, and has the true twists. It joins with C^2 continuity to adjacent sub-patches. If the twist vector for this surface were changed from the true value, $\boldsymbol{S}_{uv[T]}(0,0)$, to that estimated by Adini, $\boldsymbol{S}_{uv[A]}(0,0)$, the control point $\boldsymbol{P}_{1,1}^{0,0}$ would move by a vector amount Δ, Figure 6.2.

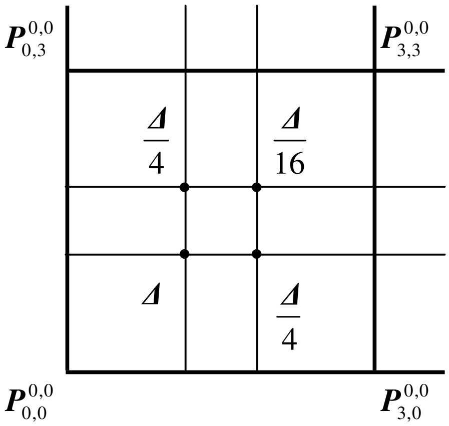

$\boldsymbol{P}_{0,3}^{0,0}$
$\boldsymbol{P}_{3,3}^{0,0}$
$\frac{\Delta}{4}$
$\frac{\Delta}{16}$
Δ
$\frac{\Delta}{4}$
$\boldsymbol{P}_{0,0}^{0,0}$
$\boldsymbol{P}_{3,0}^{0,0}$

Figure 6.2 – Effect on internal vertices when twist is changed from true to Adini

The effect on the remaining internal vertices, $\boldsymbol{P}_{1,2}^{0,0}$, $\boldsymbol{P}_{2,1}^{0,0}$ and $\boldsymbol{P}_{2,2}^{0,0}$ can be determined by considering the continuity conditions with adjacent patches. Each row and column of control points can represent a cubic Ball segment, all of which must join with C^2 continuity to their corresponding segments on the adjacent patches for the overall surface to be C^2 continuous. The *k*th cubic Ball segment, $\boldsymbol{B}^k(u)$, and its derivatives is given in terms of its control points, $\boldsymbol{P}_i^k$, $0 \le i \le 3$, by:

$$\boldsymbol{B}^k(u) = \left(1-2u+u^2\right)\boldsymbol{P}_0^k + \left(2u-4u^2+2u^3\right)\boldsymbol{P}_1^k + \left(2u^2-2u^3\right)\boldsymbol{P}_2^k + u^2\boldsymbol{P}_3^k \quad (6.6)$$

$$\frac{d\boldsymbol{B}^k}{du}(u) = \left(-2+2u\right)\boldsymbol{P}_0^k + \left(2-8u+6u^2\right)\boldsymbol{P}_1^k + \left(4u-6u^2\right)\boldsymbol{P}_2^k + 2u\boldsymbol{P}_3^k \quad (6.7)$$

$$\frac{d^2\boldsymbol{B}^k}{du^2}(u) = 2\boldsymbol{P}_0^k + \left(8+12u\right)\boldsymbol{P}_1^k + \left(4-12u\right)\boldsymbol{P}_2^k + 2\boldsymbol{P}_3^k \quad (6.8)$$

If two segments, having control points $\boldsymbol{P}_i^0$ and $\boldsymbol{P}_i^1$, $0 \le i \le 3$, respectively join with C^2 continuity, the following relationships are obtained by substituting $u = 0,1$ into (6.6) to (6.8):

$$\text{C}^0: \quad \boldsymbol{P}_3^0 = \boldsymbol{P}_0^1 \tag{6.9}$$

$$\text{C}^1: \quad -2\boldsymbol{P}_2^0 + 2\boldsymbol{P}_3^0 = -2\boldsymbol{P}_0^1 + 2\boldsymbol{P}_1^1 \tag{6.10}$$

$$\text{C}^2: \quad 2\boldsymbol{P}_0^0 + 4\boldsymbol{P}_1^0 - 8\boldsymbol{P}_2^0 + 2\boldsymbol{P}_3^0 = 2\boldsymbol{P}_0^1 - 8\boldsymbol{P}_1^1 + 4\boldsymbol{P}_2^1 + 2\boldsymbol{P}_3^1 \tag{6.11}$$

Eliminating $\boldsymbol{P}_0^1$ and $\boldsymbol{P}_1^1$ from (6.11) using (6.9) and (6.10) yields:

$$4\boldsymbol{P}_1^0 = -2\boldsymbol{P}_0^0 + 16\boldsymbol{P}_2^0 - 16\boldsymbol{P}_3^0 + 4\boldsymbol{P}_2^1 + 2\boldsymbol{P}_3^1 \tag{6.12}$$

Applying this result to the surface in Figure 6.2, it is evident that moving $\boldsymbol{P}_{1,1}^{0,0}$ by Δ will cause $\boldsymbol{P}_{1,2}^{0,0}$ to move by $\Delta/4$ as $\boldsymbol{P}_{1,0}^{0,0}$, $\boldsymbol{P}_{1,3}^{0,0}$, $\boldsymbol{P}_{1,2}^{0,1}$ and $\boldsymbol{P}_{1,3}^{0,1}$ must be fixed to maintain C^2 continuity. Likewise, moving $\boldsymbol{P}_{1,1}^{0,0}$ by Δ causes $\boldsymbol{P}_{2,1}^{0,0}$ to move by $\Delta/4$ and $\boldsymbol{P}_{2,2}^{0,0}$ to move by $\Delta/16$.

The mid-point of the Adini corner patch, $\boldsymbol{B}_{[A]}^{0,0}(½,½)$, can therefore be expressed analytically in terms of the true mid-point, $\boldsymbol{B}_{[T]}^{0,0}(½,½)$, and Δ:

$$\boldsymbol{B}_{[A]}^{0,0}(½,½) = \boldsymbol{B}_{[T]}^{0,0}(½,½) + \frac{1}{16}\left[\Delta + \frac{\Delta}{4} + \frac{\Delta}{4} + \frac{\Delta}{16}\right] = \boldsymbol{B}_{[T]}^{0,0}(½,½) + \frac{1}{16}\frac{25\Delta}{16} \tag{6.13}$$

6.2.3 Coons mid-point

In a similar way to the previous section, Figure 6.3 illustrates the difference in the internal control points between the patch $\boldsymbol{B}_{[T]}^{0,0}(u,v)$ and a bilinearly-blended Coons Patch, $\boldsymbol{C}^{0,0}(u,v)$, constructed using the true sub-patch boundaries. This sub-patch has only C^0 continuity with bounding patches. The control point $\boldsymbol{P}_{1,1}^{0,0}$ must move exactly the same amount as Adini, i.e. Δ, since Adini reproduces Coons. If the boundaries of the sub-patch $\boldsymbol{B}_{[T]}^{0,0}(u,v)$ are quadratic, the three remaining control points will be coincident with $\boldsymbol{P}_{1,1}^{0,0}$, and hence they will also move by Δ (this is a special property of the Ball basis). For cubic surfaces, this assumption is only an approximation of the movement of $\boldsymbol{P}_{1,2}^{0,0}$, $\boldsymbol{P}_{2,1}^{0,0}$ and $\boldsymbol{P}_{2,2}^{0,0}$; an associated error term is therefore produced which is examined in Section 6.3.

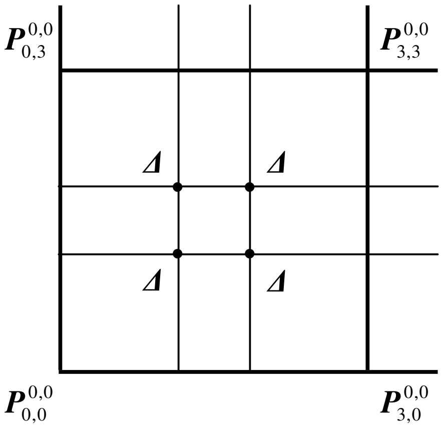

$\boldsymbol{P}_{0,3}^{0,0}$
$\boldsymbol{P}_{3,3}^{0,0}$
Δ
Δ
Δ
Δ
$\boldsymbol{P}_{0,0}^{0,0}$
$\boldsymbol{P}_{3,0}^{0,0}$

Figure 6.3 – Effect on internal vertices when the sub patch is approximated by Coons

The mid-point of the Coons corner patch, $\boldsymbol{C}^{0,0}(½,½)$, can therefore be expressed analytically in terms of the true mid-point, $\boldsymbol{B}_{[T]}^{0,0}(½,½)$, and Δ:

$$\boldsymbol{C}^{0,0}(½,½) = \boldsymbol{B}_{[T]}^{0,0}(½,½) + \frac{1}{16}4\Delta \tag{6.14}$$

6.2.4 Relationship between mid-points

For 2 x 2 sub-patches, a ratio λ_2 between the mid-points can be expressed by substituting (6.13) and (6.14) into (6.4), giving:

$$\lambda_2\left(\boldsymbol{B}_{[T]}^{0,0}(½,½) - \boldsymbol{C}^{0,0}(½,½)\right) = \boldsymbol{B}_{[A]}^{0,0}(½,½) - \boldsymbol{C}^{0,0}(½,½)$$

$$\lambda_2(0\Delta - 4\Delta) = \frac{25\Delta}{16} - 4\Delta$$

$$\lambda_2 = \frac{39}{64} \approx 0.60938$$

It is evident that there is a relationship between the three mid-points and, therefore, there must be a relationship between the three corresponding twist vectors. This relationship is exact for quadratic data, and approximate for cubic data. By considering how the continuity conditions (6.9) – (6.11) are affected by varying the number of patches, it is possible to generalise this process for s x s sub-patches and observe the behaviour of λ_s as $s \rightarrow \infty$. The relationship between the movements of any two internal adjacent control points is derived in Appendix A, and given by:

$$\frac{1}{4-f_s} \tag{6.15}$$

where $f_s = \begin{cases} 0 & s=2 \\ \left(\dfrac{1}{4-f_{s-1}}\right) & s \geq 3 \end{cases}$ (6.16)

Therefore λ_s can be expressed generally:

$$\lambda_s\left(\boldsymbol{B}^{0,0}_{[T]}(½,½)-\boldsymbol{C}^{0,0}(½,½)\right)=\boldsymbol{B}^{0,0}_{[A]}(½,½)-\boldsymbol{C}^{0,0}(½,½) \tag{6.17}$$

$$\lambda_s\left(\left(\boldsymbol{B}^{0,0}_{[T]}(½,½)\right)-\left(\boldsymbol{B}^{0,0}_{[T]}(½,½)+\frac{1}{16}4\boldsymbol{\Delta}\right)\right)=$$

$$\left(\boldsymbol{B}^{0,0}_{[T]}(½,½)+\frac{1}{16}\left[\left(\frac{1}{4-f_s}\right)^2\boldsymbol{\Delta}+2\left(\frac{1}{4-f_s}\right)\boldsymbol{\Delta}+\boldsymbol{\Delta}\right]\right)-\left(\boldsymbol{B}^{0,0}_{[T]}(½,½)+\frac{1}{16}4\boldsymbol{\Delta}\right)$$

$$\lambda_s=\frac{\left(\left(\frac{1}{4-f_s}\right)^2+2\left(\frac{1}{4-f_s}\right)+1\right)-4}{0-4}=\frac{39-22f_s+3f_s^{\,2}}{4\left(4-f_s\right)^2} \tag{6.18}$$

The value of f_s in (6.16) is given by a recurrence relationship; a more convenient formulation using a linear difference equation is derived in Appendix B, and is given by:

$$f_s=\frac{\left(2+\sqrt{3}\right)^{s-1}-\left(2-\sqrt{3}\right)^{s-3}}{\left(2+\sqrt{3}\right)^{s}-\left(2-\sqrt{3}\right)^{s-2}},\ s\geq 3 \tag{6.19}$$

In the limit:

$$\lim_{s\to\infty} f_s = \left(2-\sqrt{3}\right) \tag{6.20}$$

$$\lim_{s\to\infty} \lambda_s = ½\left(3\sqrt{3}-4\right) \approx 0.59808 \tag{6.21}$$

λ_s converges rapidly as s increases: with only 4 spans the relative error in λ_s is less than 0.01% – see Figure 6.4. This analysis makes the necessary assumption that only one twist is being estimated, and that the other three corners have true twists. This assumption is valid because the effect of an adjacent corner twist rapidly decreases as s increases. By considering the continuity conditions between adjacent patches, it can be shown that the effect of changing $\boldsymbol{P}_{1,1}^{0,0}$ by Δ on the position of the mid-point in adjacent patches is $\frac{25}{16 \cdot 4^h}\Delta$, where h is the span index, $0 \le h < s$.

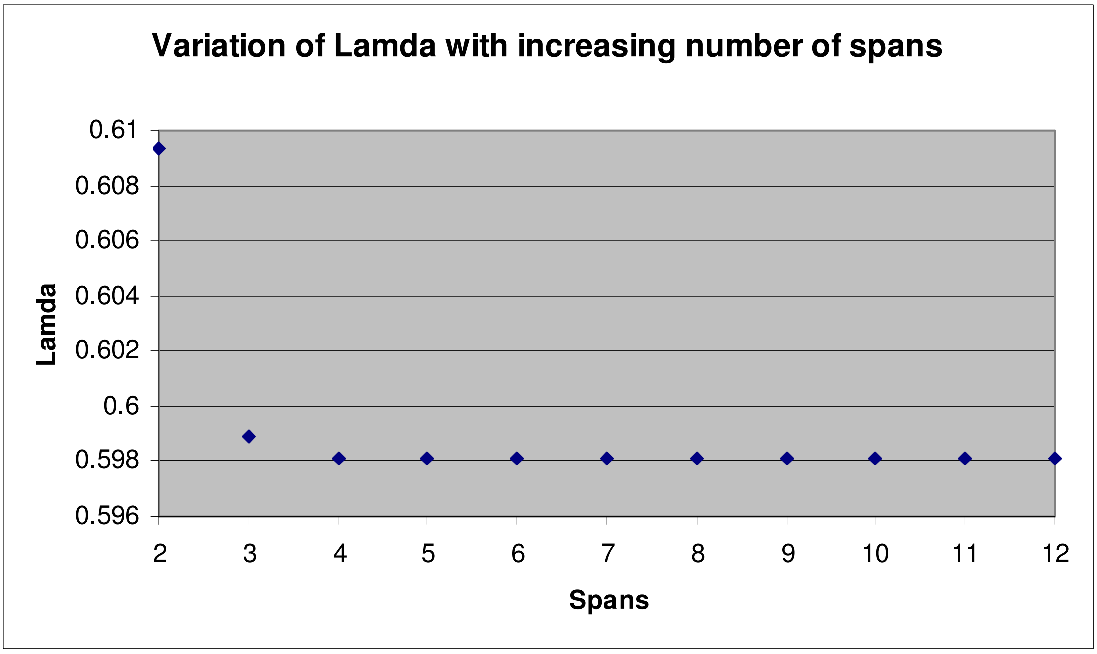


Figure 6.4 – Value of λ_s as the number of spans increase

6.3 Error analysis for cubic data

For quadratic data, the relationship (6.4) is exact assuming that λ is given by (6.21) and $s \to \infty$. It has been shown that the error associated with this assumption converges rapidly. The error that results from using cubic data with the Improved algorithm can be evaluated; this error can then be compared with the error produced by the Adini method. As twist vectors have no geometric interpretation, the error term is taken as the vector difference between the estimated and true mid-points of the corner sub-patches, i.e.:

$$\boldsymbol{E}_{[A]}^{0,0}\left(½,½\right) = \boldsymbol{S}_{[A]}^{0,0}\left(½,½\right) - \boldsymbol{S}_{[T]}^{0,0}\left(½,½\right)$$
$$\boldsymbol{E}_{[I]}^{0,0}\left(½,½\right) = \boldsymbol{S}_{[I]}^{0,0}\left(½,½\right) - \boldsymbol{S}_{[T]}^{0,0}\left(½,½\right)$$

The error term is developed to correspond with the sub-patch $\boldsymbol{B}_{[T]}^{0,0}\left(u,v\right)$; the error terms for the other corner sub-patches can be derived analogously. Analytical expressions for $\boldsymbol{B}_{[T]}^{0,0}\left(½,½\right)$, $\boldsymbol{C}^{0,0}\left(½,½\right)$, $\boldsymbol{B}_{[A]}^{0,0}\left(½,½\right)$ and $\boldsymbol{B}_{[I]}^{0,0}\left(½,½\right)$ are required in terms of the original 16 control points, $\boldsymbol{P}_{i,j}^{0,0}$, $0 \le i, j \le 3$.

6.3.1 True mid-point

The mid-point of the true sub-patch is given by (6.5), although it is more convenient to expand this expression to include all 16 control points:

$$\boldsymbol{B}_{[T]}^{0,0}\left(½,½\right) = \frac{1}{16}\begin{pmatrix} +1\boldsymbol{P}_{0,0}^{0,0} & +1\boldsymbol{P}_{0,1}^{0,0} & +1\boldsymbol{P}_{0,2}^{0,0} & +1\boldsymbol{P}_{0,3}^{0,0} \\ +1\boldsymbol{P}_{1,0}^{0,0} & +1\boldsymbol{P}_{1,1}^{0,0} & +1\boldsymbol{P}_{1,2}^{0,0} & +1\boldsymbol{P}_{1,3}^{0,0} \\ +1\boldsymbol{P}_{2,0}^{0,0} & +1\boldsymbol{P}_{2,1}^{0,0} & +1\boldsymbol{P}_{2,2}^{0,0} & +1\boldsymbol{P}_{2,3}^{0,0} \\ +1\boldsymbol{P}_{3,0}^{0,0} & +1\boldsymbol{P}_{3,1}^{0,0} & +1\boldsymbol{P}_{3,2}^{0,0} & +1\boldsymbol{P}_{3,3}^{0,0} \end{pmatrix} \qquad (6.22)$$

Note that this is *not* a matrix, but a sum of 16 vector quantities.

6.3.2 Coons mid-point

A bilinearly-blended Coons patch is constructed from the patch's boundary data alone. The internal vertices of the Coons patch using the Ball basis are given by:

$$\boldsymbol{P}_{i,j[C]} = \frac{1}{2}\left(\boldsymbol{P}_{0,j} + \boldsymbol{P}_{i,0} + \boldsymbol{P}_{i,3} + \boldsymbol{P}_{3,j}\right) - \frac{1}{4}\left(\boldsymbol{P}_{0,0} + \boldsymbol{P}_{0,3} + \boldsymbol{P}_{3,0} + \boldsymbol{P}_{3,3}\right), \qquad 1 \le i, j \le 2 \tag{6.23}$$

The mid-point of the coons patch, $\boldsymbol{C}^{0,0}\left(½,½\right)$, is obtained by substituting (6.23) into (6.22):

$$\boldsymbol{C}^{0,0}\left(½,½\right) = \frac{1}{16}\begin{pmatrix} +0\boldsymbol{P}^{0,0}_{0,0} & +2\boldsymbol{P}^{0,0}_{0,1} & +2\boldsymbol{P}^{0,0}_{0,2} & +0\boldsymbol{P}^{0,0}_{0,3} \\ +2\boldsymbol{P}^{0,0}_{1,0} & +0\boldsymbol{P}^{0,0}_{1,1} & +0\boldsymbol{P}^{0,0}_{1,2} & +2\boldsymbol{P}^{0,0}_{1,3} \\ +2\boldsymbol{P}^{0,0}_{2,0} & +0\boldsymbol{P}^{0,0}_{2,1} & +0\boldsymbol{P}^{0,0}_{2,2} & +2\boldsymbol{P}^{0,0}_{2,3} \\ +0\boldsymbol{P}^{0,0}_{3,0} & +2\boldsymbol{P}^{0,0}_{3,1} & +2\boldsymbol{P}^{0,0}_{3,2} & +0\boldsymbol{P}^{0,0}_{3,3} \end{pmatrix} \tag{6.24}$$

6.3.3 Adini mid-point

The Adini mid-point cannot be expressed directly in terms of the original 16 vertices as it is a function of *s*. When $s = 2$, the mid-point is given by (6.13). By considering the C^2 continuity conditions between adjacent patches, the relationship between adjacent internal control points is given by (6.15); the value in the limit is obtained by substituting (6.20):

$$\frac{1}{\left(2+\sqrt{3}\right)} \tag{6.25}$$

It is noted that (6.15) converges rapidly, allowing (6.13) to be rewritten for $s \to \infty$:

$$\boldsymbol{B}^{0,0}_{[A]}\left(½,½\right) = \boldsymbol{B}^{0,0}_{[T]}\left(½,½\right) + \frac{1}{16}\left(\boldsymbol{\Delta} + 2\left(\frac{1}{\left(2+\sqrt{3}\right)}\right)\boldsymbol{\Delta} + \left(\frac{1}{\left(2+\sqrt{3}\right)^2}\right)\boldsymbol{\Delta}\right)$$

$$\boldsymbol{B}^{0,0}_{[A]}\left(½,½\right) = \boldsymbol{B}^{0,0}_{[T]}\left(½,½\right) + \frac{6\left(2-\sqrt{3}\right)\boldsymbol{\Delta}}{16} \tag{6.26}$$

The vector $\boldsymbol{\Delta}$ is given by the difference between the corresponding $\boldsymbol{P}^{0,0}_{1,1}$ control points on the Coons and True surfaces, i.e.:

$$\mathit{\Delta} = \boldsymbol{P}^{0,0}_{1,1[C]} - \boldsymbol{P}^{0,0}_{1,1[\mathrm{T}]}$$

$$\mathit{\Delta} = \frac{1}{16}\begin{pmatrix} -1\boldsymbol{P}^{0,0}_{1,1} & +2\boldsymbol{P}^{0,0}_{1,1} & +0\boldsymbol{P}^{0,0}_{1,1} & -1\boldsymbol{P}^{0,0}_{1,1} \\ +2\boldsymbol{P}^{0,0}_{1,1} & -4\boldsymbol{P}^{0,0}_{1,1} & +0\boldsymbol{P}^{0,0}_{1,1} & +2\boldsymbol{P}^{0,0}_{1,1} \\ +0\boldsymbol{P}^{0,0}_{1,1} & +0\boldsymbol{P}^{0,0}_{1,1} & +0\boldsymbol{P}^{0,0}_{1,1} & +0\boldsymbol{P}^{0,0}_{1,1} \\ -1\boldsymbol{P}^{0,0}_{1,1} & +2\boldsymbol{P}^{0,0}_{1,1} & +0\boldsymbol{P}^{0,0}_{1,1} & -1\boldsymbol{P}^{0,0}_{1,1} \end{pmatrix} \tag{6.27}$$

Therefore the convergent Adini mid-point can be expressed in terms of the original control points of the true sub-patch $\boldsymbol{B}^{0,0}_{[T]}(u,v)$ by substituting (6.27) into (6.26):

$$\boldsymbol{B}^{0,0}_{[A]}(½,½) = \frac{1}{64}\begin{pmatrix} +\left(-8+6\sqrt{3}\right)\boldsymbol{P}^{0,0}_{0,0} & +\left(28-12\sqrt{3}\right)\boldsymbol{P}^{0,0}_{0,1} & +4\boldsymbol{P}^{0,0}_{0,2} & +\left(-8+6\sqrt{3}\right)\boldsymbol{P}^{0,0}_{0,3} \\ +\left(28-12\sqrt{3}\right)\boldsymbol{P}^{0,0}_{1,0} & +\left(-44+24\sqrt{3}\right)\boldsymbol{P}^{0,0}_{1,1} & +4\boldsymbol{P}^{0,0}_{1,2} & +\left(28-12\sqrt{3}\right)\boldsymbol{P}^{0,0}_{1,3} \\ +4\boldsymbol{P}^{0,0}_{2,0} & +4\boldsymbol{P}^{0,0}_{2,1} & +4\boldsymbol{P}^{0,0}_{2,2} & +4\boldsymbol{P}^{0,0}_{2,3} \\ +\left(-8+6\sqrt{3}\right)\boldsymbol{P}^{0,0}_{3,0} & +\left(28-12\sqrt{3}\right)\boldsymbol{P}^{0,0}_{3,1} & +4\boldsymbol{P}^{0,0}_{3,2} & +\left(-8+6\sqrt{3}\right)\boldsymbol{P}^{0,0}_{3,3} \end{pmatrix} \tag{6.28}$$

6.3.4 Improved mid-point

The Improved mid-point is obtained by rearranging (6.4):

$$\boldsymbol{B}^{0,0}_{[I]}(½,½) = \frac{\boldsymbol{B}^{0,0}_{[A]}(½,½) - (1-\lambda)\boldsymbol{C}^{0,0}(½,½)}{\lambda} \tag{6.29}$$

and assuming the value of λ from (6.21). The mid-point $\boldsymbol{B}^{0,0}_{[I]}(½,½)$ can be evaluated by substituting (6.24) and (6.28):

$$\boldsymbol{B}_{[I]}^{0,0}\left(½,½\right)=\frac{1}{64}\frac{2\left(3\sqrt{3}+4\right)}{11}$$

$$\begin{pmatrix} +\left(-8+6\sqrt{3}\right)\boldsymbol{P}_{0,0}^{0,0} & +4\boldsymbol{P}_{0,1}^{0,0} & +\left(-20+12\sqrt{3}\right)\boldsymbol{P}_{0,2}^{0,0} & +\left(-8+6\sqrt{3}\right)\boldsymbol{P}_{0,3}^{0,0} \\ +4\boldsymbol{P}_{1,0}^{0,0} & +\left(-44+24\sqrt{3}\right)\boldsymbol{P}_{1,1}^{0,0} & +4\boldsymbol{P}_{1,2}^{0,0} & +4\boldsymbol{P}_{1,3}^{0,0} \\ +\left(-20+12\sqrt{3}\right)\boldsymbol{P}_{2,0}^{0,0} & +4\boldsymbol{P}_{2,1}^{0,0} & +4\boldsymbol{P}_{2,2}^{0,0} & +\left(-20+12\sqrt{3}\right)\boldsymbol{P}_{2,3}^{0,0} \\ +\left(-8+6\sqrt{3}\right)\boldsymbol{P}_{3,0}^{0,0} & +4\boldsymbol{P}_{3,1}^{0,0} & +\left(-20+12\sqrt{3}\right)\boldsymbol{P}_{3,2}^{0,0} & +\left(-8+6\sqrt{3}\right)\boldsymbol{P}_{3,3}^{0,0} \end{pmatrix}$$

(6.30)

6.3.5 Error terms

Expressing the errors as the vector difference between the interpolated and true mid-points, the Adini error, $\boldsymbol{E}_{[A]}^{0,0}\left(½,½\right)$, is found using (6.4) and (6.28):

$$\boldsymbol{E}_{[A]}^{0,0}\left(½,½\right)=\boldsymbol{S}_{[A]}^{0,0}\left(½,½\right)-\boldsymbol{S}_{[T]}^{0,0}\left(½,½\right)$$

$$\boldsymbol{E}_{[A]}^{0,0}\left(½,½\right)=\frac{1}{64}\left(-12+6\sqrt{3}\right)\begin{pmatrix} +1\boldsymbol{P}_{0,0}^{0,0} & -2\boldsymbol{P}_{0,1}^{0,0} & +0\boldsymbol{P}_{0,2}^{0,0} & +1\boldsymbol{P}_{0,3}^{0,0} \\ -2\boldsymbol{P}_{1,0}^{0,0} & +4\boldsymbol{P}_{1,1}^{0,0} & +0\boldsymbol{P}_{1,2}^{0,0} & -2\boldsymbol{P}_{1,3}^{0,0} \\ +0\boldsymbol{P}_{2,0}^{0,0} & +0\boldsymbol{P}_{2,1}^{0,0} & +0\boldsymbol{P}_{2,2}^{0,0} & +0\boldsymbol{P}_{2,3}^{0,0} \\ +1\boldsymbol{P}_{3,0}^{0,0} & -2\boldsymbol{P}_{3,1}^{0,0} & +0\boldsymbol{P}_{3,2}^{0,0} & +1\boldsymbol{P}_{3,3}^{0,0} \end{pmatrix} \qquad (6.31)$$

The Improved error is found using (6.4) and (6.30):

$$\boldsymbol{E}_{[I]}^{0,0}\left(½,½\right)=\boldsymbol{S}_{[I]}^{0,0}\left(½,½\right)-\boldsymbol{S}_{[T]}^{0,0}\left(½,½\right)$$

$$\boldsymbol{E}_{[I]}^{0,0}\left(½,½\right)=\frac{1}{64}\frac{2\left(3\sqrt{3}+4\right)}{11}\left(-12+6\sqrt{3}\right)\begin{pmatrix} +0\boldsymbol{P}_{0,0}^{0,0} & -1\boldsymbol{P}_{0,1}^{0,0} & +1\boldsymbol{P}_{0,2}^{0,0} & +0\boldsymbol{P}_{0,3}^{0,0} \\ -1\boldsymbol{P}_{1,0}^{0,0} & +3\boldsymbol{P}_{1,1}^{0,0} & -1\boldsymbol{P}_{1,2}^{0,0} & -1\boldsymbol{P}_{1,3}^{0,0} \\ +1\boldsymbol{P}_{2,0}^{0,0} & -1\boldsymbol{P}_{2,1}^{0,0} & -1\boldsymbol{P}_{2,2}^{0,0} & +1\boldsymbol{P}_{2,3}^{0,0} \\ +0\boldsymbol{P}_{3,0}^{0,0} & -1\boldsymbol{P}_{3,1}^{0,0} & +1\boldsymbol{P}_{3,2}^{0,0} & +0\boldsymbol{P}_{3,3}^{0,0} \end{pmatrix}$$

(6.32)

It is impossible to meaningfully compare the two error terms directly; it could only be concluded that one error was smaller than another, in general, if the coefficient of every control point was smaller in one expression compared with the other. However, significant meaning can be obtained by observing the rate at which the error terms diminish when the original patch is subdivided at $\boldsymbol{B}^{0,0}_{[T]}(½,½)$. The 16 control points belonging to the subdivided patch, $\boldsymbol{P}^{0,0\prime}_{i,j}$, can be expressed in terms of the original 16 control points $\boldsymbol{P}^{0,0}_{i,j}$, $0 \le i, j \le 3$, by equating parametric points, derivatives and twists, at $\boldsymbol{S}^{0,0}(0,0)$, $\boldsymbol{S}^{0,0}(0,½)$, $\boldsymbol{S}^{0,0}(½,0)$ and $\boldsymbol{S}^{0,0}(½,½)$, i.e.:

$$\boldsymbol{S}^{0,0\prime}(0,0) = \boldsymbol{S}^{0,0}(0,0) \quad \Rightarrow \quad \boldsymbol{P}^{0,0\prime}_{0,0} = \boldsymbol{P}^{0,0}_{0,0},$$

$$\boldsymbol{S}^{0,0\prime}(0,1) = \boldsymbol{S}^{0,0}(0,½) \quad \Rightarrow \quad \boldsymbol{P}^{0,0\prime}_{0,1} = \frac{1}{2}\boldsymbol{P}^{0,0}_{0,0} + \frac{1}{2}\boldsymbol{P}^{0,0}_{0,1},$$

$$\boldsymbol{S}^{0,0\prime}_{u}(0,0) = ½\boldsymbol{S}^{0,0}_{u}(0,0) \quad \Rightarrow \quad \boldsymbol{P}^{0,0\prime}_{1,0} = \frac{1}{2}\boldsymbol{P}^{0,0}_{0,0} + \frac{1}{2}\boldsymbol{P}^{0,0}_{1,0}, \text{ etc..}$$

Note that derivatives must be scaled down by a factor of 2, and twists by a factor of 4.

Substituting the expressions for the subdivided control points into (6.31) and (6.32) gives the Adini and Improved errors following one level of subdivision:

$$\boldsymbol{E}^{0,0\prime}_{[A]}(½,½) = \frac{1}{4096}\left(-12+6\sqrt{3}\right)\begin{pmatrix} +4\boldsymbol{P}^{0,0}_{0,0} & -12\boldsymbol{P}^{0,0}_{0,1} & +4\boldsymbol{P}^{0,0}_{0,2} & +4\boldsymbol{P}^{0,0}_{0,3} \\ -12\boldsymbol{P}^{0,0}_{1,0} & +36\boldsymbol{P}^{0,0}_{1,1} & -12\boldsymbol{P}^{0,0}_{1,2} & -12\boldsymbol{P}^{0,0}_{1,3} \\ +4\boldsymbol{P}^{0,0}_{2,0} & -12\boldsymbol{P}^{0,0}_{2,1} & +4\boldsymbol{P}^{0,0}_{2,2} & +4\boldsymbol{P}^{0,0}_{2,3} \\ +4\boldsymbol{P}^{0,0}_{3,0} & -12\boldsymbol{P}^{0,0}_{3,1} & +4\boldsymbol{P}^{0,0}_{3,2} & +4\boldsymbol{P}^{0,0}_{3,3} \end{pmatrix} \tag{6.33}$$

$$\boldsymbol{E}^{0,0\prime}_{[I]}(½,½) = \frac{1}{4096}\frac{2\left(3\sqrt{3}+4\right)}{11}\left(-12+6\sqrt{3}\right)\begin{pmatrix} +0\boldsymbol{P}^{0,0}_{0,0} & -2\boldsymbol{P}^{0,0}_{0,1} & +2\boldsymbol{P}^{0,0}_{0,2} & +0\boldsymbol{P}^{0,0}_{0,3} \\ -2\boldsymbol{P}^{0,0}_{1,0} & +11\boldsymbol{P}^{0,0}_{1,1} & -7\boldsymbol{P}^{0,0}_{1,2} & -2\boldsymbol{P}^{0,0}_{1,3} \\ +2\boldsymbol{P}^{0,0}_{2,0} & -7\boldsymbol{P}^{0,0}_{2,1} & +3\boldsymbol{P}^{0,0}_{2,2} & +2\boldsymbol{P}^{0,0}_{2,3} \\ +0\boldsymbol{P}^{0,0}_{3,0} & -2\boldsymbol{P}^{0,0}_{3,1} & +2\boldsymbol{P}^{0,0}_{3,2} & +0\boldsymbol{P}^{0,0}_{3,3} \end{pmatrix} \tag{6.34}$$

The rate at which the Adini error is reducing cannot be determined because (6.33) has non-zero coefficients that are not present in (6.31). However, subdividing a second time overcomes this problem:

$$\boldsymbol{E}_{[A]}^{0,0''}\left(½,½\right)=\frac{1}{262144}\left(-12+6\sqrt{3}\right)\begin{pmatrix} +16\boldsymbol{P}_{0,0}^{0,0} & -56\boldsymbol{P}_{0,1}^{0,0} & +24\boldsymbol{P}_{0,2}^{0,0} & +16\boldsymbol{P}_{0,3}^{0,0} \\ -56\boldsymbol{P}_{1,0}^{0,0} & +196\boldsymbol{P}_{1,1}^{0,0} & -84\boldsymbol{P}_{1,2}^{0,0} & -56\boldsymbol{P}_{1,3}^{0,0} \\ +24\boldsymbol{P}_{2,0}^{0,0} & -84\boldsymbol{P}_{2,1}^{0,0} & +36\boldsymbol{P}_{2,2}^{0,0} & +24\boldsymbol{P}_{2,3}^{0,0} \\ +16\boldsymbol{P}_{3,0}^{0,0} & -56\boldsymbol{P}_{3,1}^{0,0} & +24\boldsymbol{P}_{3,2}^{0,0} & +16\boldsymbol{P}_{3,3}^{0,0} \end{pmatrix} \tag{6.35}$$

$$\boldsymbol{E}_{[I]}^{0,0''}\left(½,½\right)=\frac{1}{262144}\frac{2\left(3\sqrt{3}+4\right)}{11}\left(-12+6\sqrt{3}\right)\begin{pmatrix} +0\boldsymbol{P}_{0,0}^{0,0} & -4\boldsymbol{P}_{0,1}^{0,0} & +4\boldsymbol{P}_{0,2}^{0,0} & +0\boldsymbol{P}_{0,3}^{0,0} \\ -4\boldsymbol{P}_{1,0}^{0,0} & +27\boldsymbol{P}_{1,1}^{0,0} & -19\boldsymbol{P}_{1,2}^{0,0} & -4\boldsymbol{P}_{1,3}^{0,0} \\ +4\boldsymbol{P}_{2,0}^{0,0} & -19\boldsymbol{P}_{2,1}^{0,0} & +11\boldsymbol{P}_{2,2}^{0,0} & +4\boldsymbol{P}_{2,3}^{0,0} \\ +0\boldsymbol{P}_{3,0}^{0,0} & -4\boldsymbol{P}_{3,1}^{0,0} & +4\boldsymbol{P}_{3,2}^{0,0} & +0\boldsymbol{P}_{3,3}^{0,0} \end{pmatrix} \tag{6.36}$$

Considering equations (6.33), (6.34), (6.35) and (6.36), it is evident that each of the 16 components of the Improved error reduces at least twice as quickly compared with Adini. The ratios are:

$$\boldsymbol{E}_{[A]}^{0,0''}\left(½,½\right):\boldsymbol{E}_{[A]}^{0,0'}\left(½,½\right) \qquad\qquad \boldsymbol{E}_{[I]}^{0,0''}\left(½,½\right):\boldsymbol{E}_{[I]}^{0,0'}\left(½,½\right)$$

$$\frac{1}{64}\begin{bmatrix} 4 & {}^{14}\!/_{3} & 6 & 4 \\ {}^{14}\!/_{3} & {}^{49}\!/_{9} & 7 & {}^{14}\!/_{3} \\ 6 & 7 & 9 & 6 \\ 4 & {}^{14}\!/_{3} & 6 & 4 \end{bmatrix} \qquad\qquad \frac{1}{64}\begin{bmatrix} 0 & 2 & 2 & 0 \\ 2 & {}^{27}\!/_{11} & {}^{19}\!/_{7} & 2 \\ 2 & {}^{19}\!/_{7} & {}^{11}\!/_{4} & 2 \\ 0 & 2 & 2 & 0 \end{bmatrix}$$

Figure 6.5 shows how the error associated with the Improved method reduces at a much faster rate than Adini as *s* increases. The error is measured in terms of the Euclidean distance between the parametric mid-points of the interpolated and true corner sub-patches, and is directly proportional to the magnitude of the absolute difference between the corresponding twist vectors.

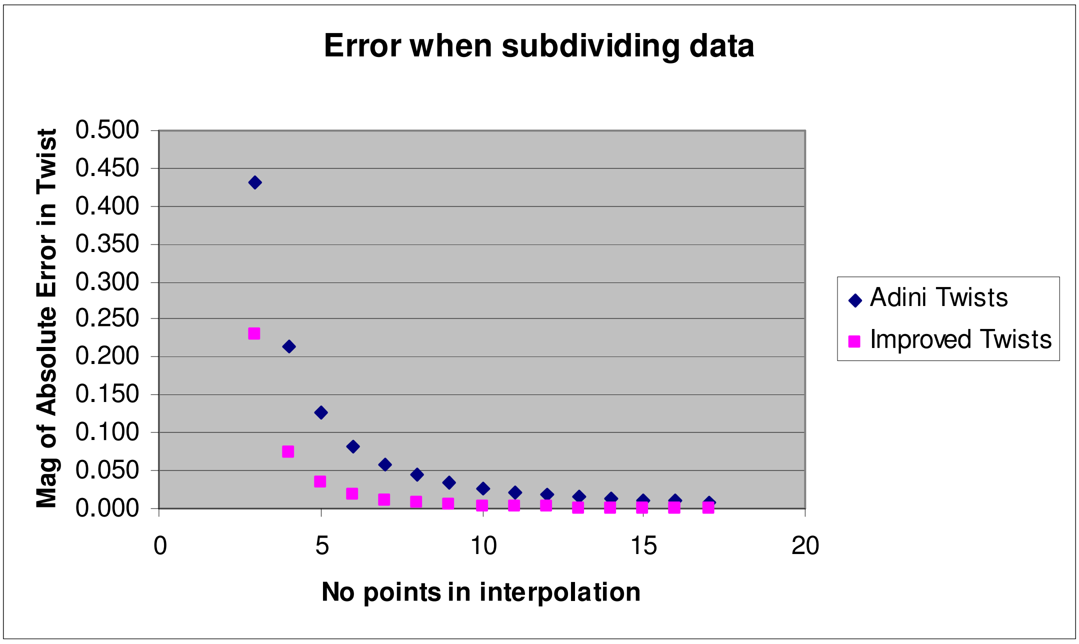


Figure 6.5 – Reducing error terms with increasing number of interpolated points

The Adini and the Improved algorithms produce identical errors when interpolating a single patch. As *s* increases, the Improved method will, in general, produce a smaller error. Whilst it is possible to construct cases where the Adini algorithm produces a smaller error for a specific *s*, the Improved algorithm would outperform Adini if the interpolation data were subdivided. The Improved algorithm is therefore a better choice than Adini when selecting a general-purpose twist vector estimation algorithm.

6.4 Numerical testing and performance

Generally, the true twist for an interpolated surface would not be known, but the performance of any algorithm can be assessed by sampling interpolation data – including the twist vector – from a range of explicitly defined surfaces, and assessing how well the algorithm performs in reproducing them. The original surfaces must be unbiased to any particular twist estimation algorithm, and ideally not reliant on any special parameterisation or derivative magnitude estimation algorithms. Four primitive surface types were selected (cylinders, cones, spheres and tori) as they characterise many engineering surfaces, the construction is not influenced by any factors that would bias the results, and the data sets can be easily reproduced. Figure 6.6 illustrates the four surface types. Each surface is created using 6 data points in each parametric direction, following the guidelines given in Chapter 5.

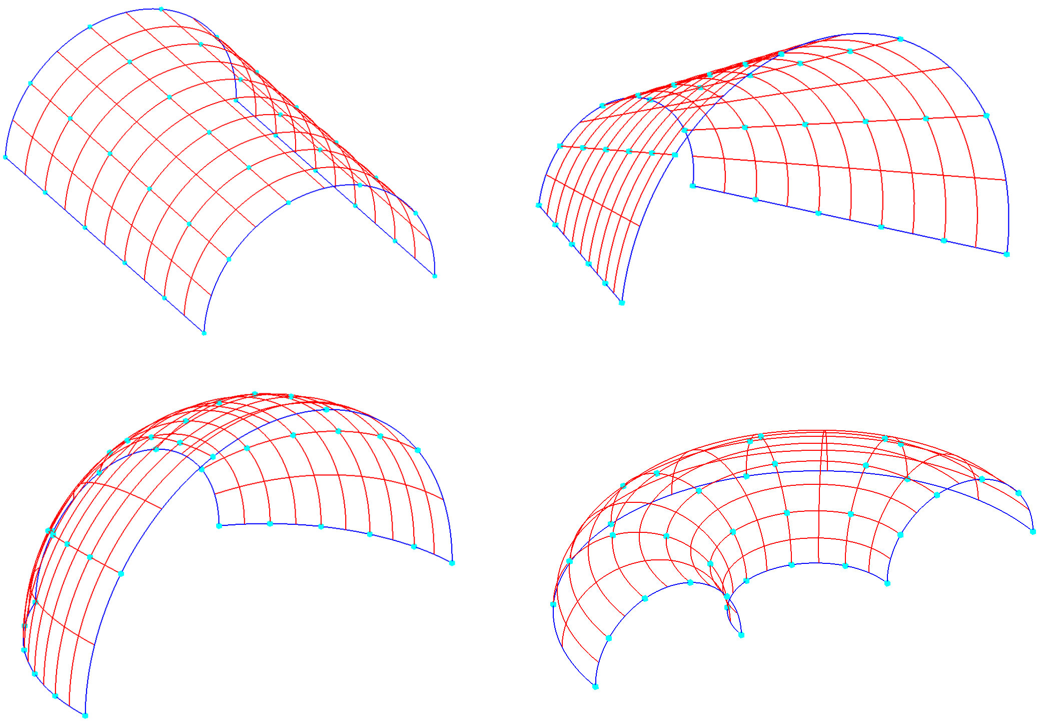
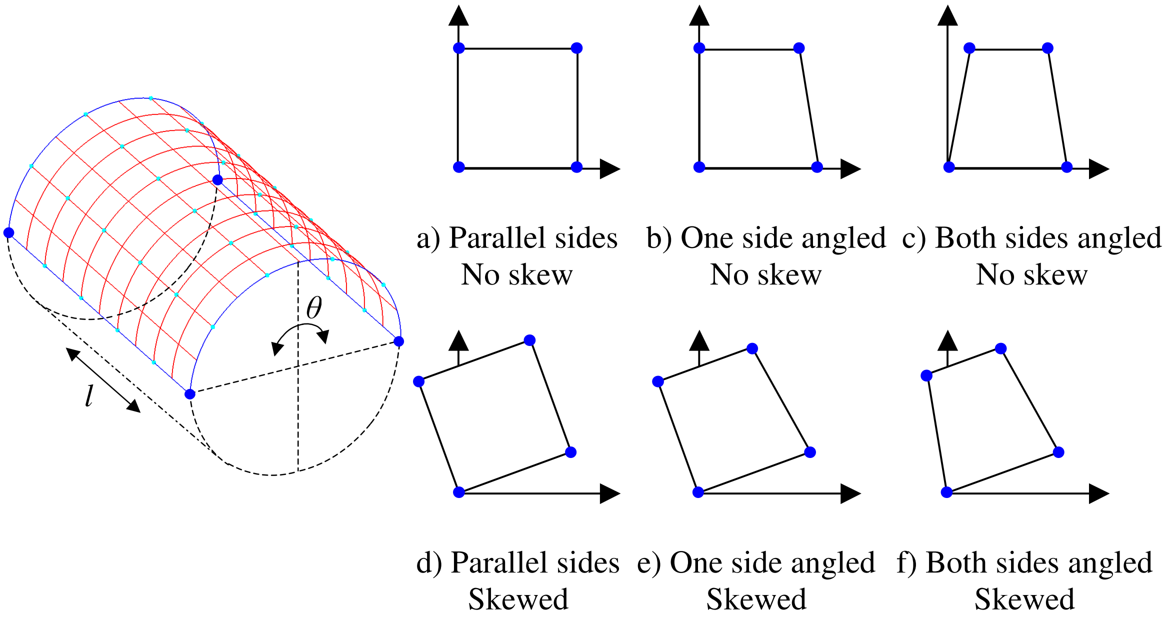


Figure 6.6 – Four surface types: Cylinder, Cone, Sphere, and Torus

To generate a wider range of unbiased surfaces, points were sampled from the four primitive types in six different configurations by skewing the data grids and angling their boundaries with respect to the 'natural' orientations illustrated in Figure 6.6. This is illustrated in Figure 6.7 for a cylinder, although the method is analogous for the other surfaces.



a) Parallel sides No skew

b) One side angled No skew

c) Both sides angled No skew

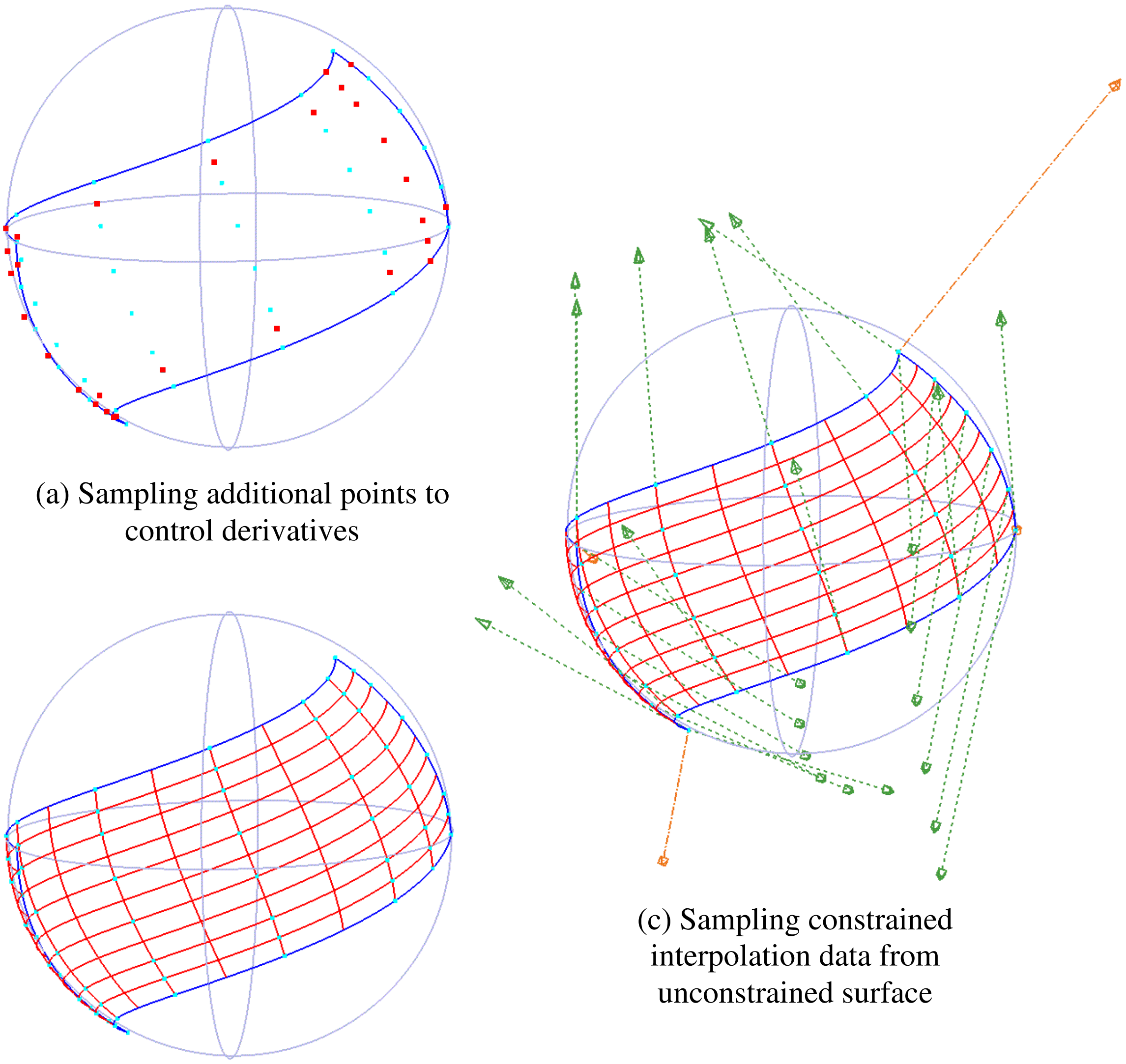
d) Parallel sides Skewed

e) One side angled Skewed

f) Both sides angled Skewed

Figure 6.7 – Six configurations for parametrically sampling points from a cylinder

The required interpolation data consists of points and derivatives. Whilst derivative directions could be sampled from the primitive data, their magnitudes would need to be estimated using an appropriate algorithm as they are a function of the parameterisation of a surface, not its geometry. To avoid introducing unnecessary ambiguity into the resulting surface, it was decided to construct the 'true' surfaces by performing an unconstrained interpolation of the data points. Additional points were sampled from the original surfaces to control the cross boundary derivatives and twists. These additional points are located at the parametric mid-point between the first and second, and penultimate and last, data points in each row and column – denoted by the red dots in Figure 6.8(a). Green arrows are used to represent derivatives, and orange arrows to represent twists in Figure 6.8(c).



(a) Sampling additional points to control derivatives

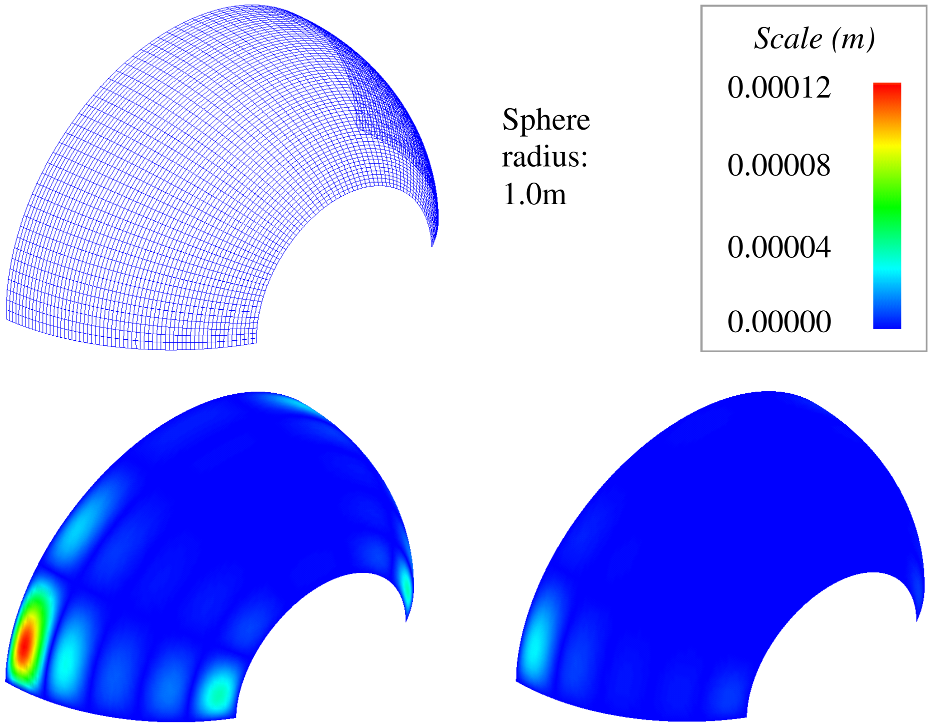
(b) Unconstrained interpolation: ‘true’ surface

(c) Sampling constrained interpolation data from unconstrained surface

Figure 6.8 – Obtaining the constrained interpolation data

Points and derivatives were sampled from each of the 24 ‘true’ surfaces, and used to create the Adini and Improved twist vector estimates. The performance of the algorithms could be measured as the absolute error between the estimated and ‘true’ twist vectors. However, the twist vectors have no geometric interpretation so the numerical values are difficult to interpret. Instead, the positional error between the corner patch mid-points was used, which is directly proportional to the absolute error between the true and interpolated twist vectors. Measuring the error as a distance does not affect the outcome of the results, but gives it geometric significance as it can be related to a physical tolerance.

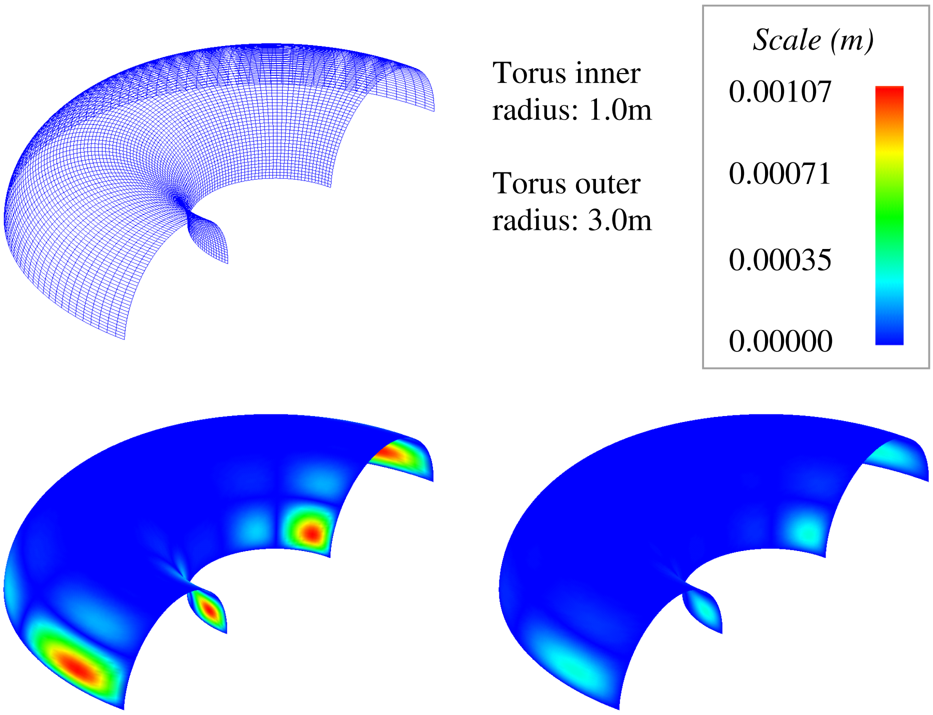
The results for each of the 96 cases (24 surfaces, each with 4 corners) are given in Appendix C. The Improved method outperforms Adini in all 96 cases, and on average produces an error that is 4.95 times smaller, with a minimum of 0.0 and a maximum of 61.8. It is possible to visualise the effect that the twist vectors have over the entire surface by considering the positional error at all points on the surface. Figures 6.9 and 6.10 show the errors for a sphere and a torus respectively. The colour indicates the error in terms of the Euclidean distance between the surfaces with the true and estimated twists, calculated using a geometric arc length approach similar to Chong [2006], although modified to parameterise each patch individually as suggested for curve segments in Chapter 4.



a) Adini twists

b) Improved twists

Figure 6.9 – Position error for sphere data when using a) Adini b) Improved Twists



a) Adini twists

b) Improved twists

Figure 6.10 – Position error for torus data when using a) Adini b) Improved Twists

6.5 Conclusion

It has been shown that the Improved algorithm has a theoretical foundation, and for quadratic surfaces it reproduces twists that tend towards the exact solution. For cubic data, the Improved algorithm produces an identical result to Adini for single patch interpolations, and, in general, produces a better estimate than Adini for multi-patch interpolations. It has been shown that the error produced by the Improved algorithm reduces at a faster rate than Adini when the data is sub-divided. A wide range of numerical tests was conducted to practically compare the algorithms, and in all cases the Improved algorithm outperformed Adini. The Improved algorithm is therefore recommended as the best method for estimating twist vectors for general purpose interpolation of engineering data.

Chapter 7

Arbitrary topologies

The primary focus of this thesis is to control the interpolation process, with the aim of constructing high quality surfaces; however, ‘real’ surfaces rarely exist in isolation, and are frequently required to join with other surfaces. This chapter considers the conditions for joining two surfaces with parametric and geometric continuity.

An interpolated NUBS surface has exactly four sides, or boundaries, assuming that the surface is not degenerate (i.e. with a zero-length side). Certain geometries arise that require the use of *N*-sided surfaces. Various *N*-sided surface definitions have been proposed, although none are widely accepted and most CAD software will only support rectangular surface types, i.e. $N = 4$. As a consequence, it is helpful to be able to represent a non-rectangular surface using an assembly of rectangular patches. Kahmann [1983] proposed a generalised algebraic method for joining two adjacent patches with geometric continuity. This technique has been developed and reformulated to allow an *N*-sided assembly of patches to be constructed with G^1 continuity. G^2 continuity is desirable but it is proven that this is, in general, mathematically impossible using Kahmann’s method applied to arbitrary topologies.

7.1 Arbitrary topologies in CAGD

Most surfaces in CAGD can be represented using rectangular surfaces, i.e. $N = 4$, that join with an acceptable level of continuity. Certain geometries arise which require non-rectangular surfaces; these surfaces cannot be assembled from rectangular patches that join with parametric continuity. Examples of such circumstances are illustrated in Figure 7.1.

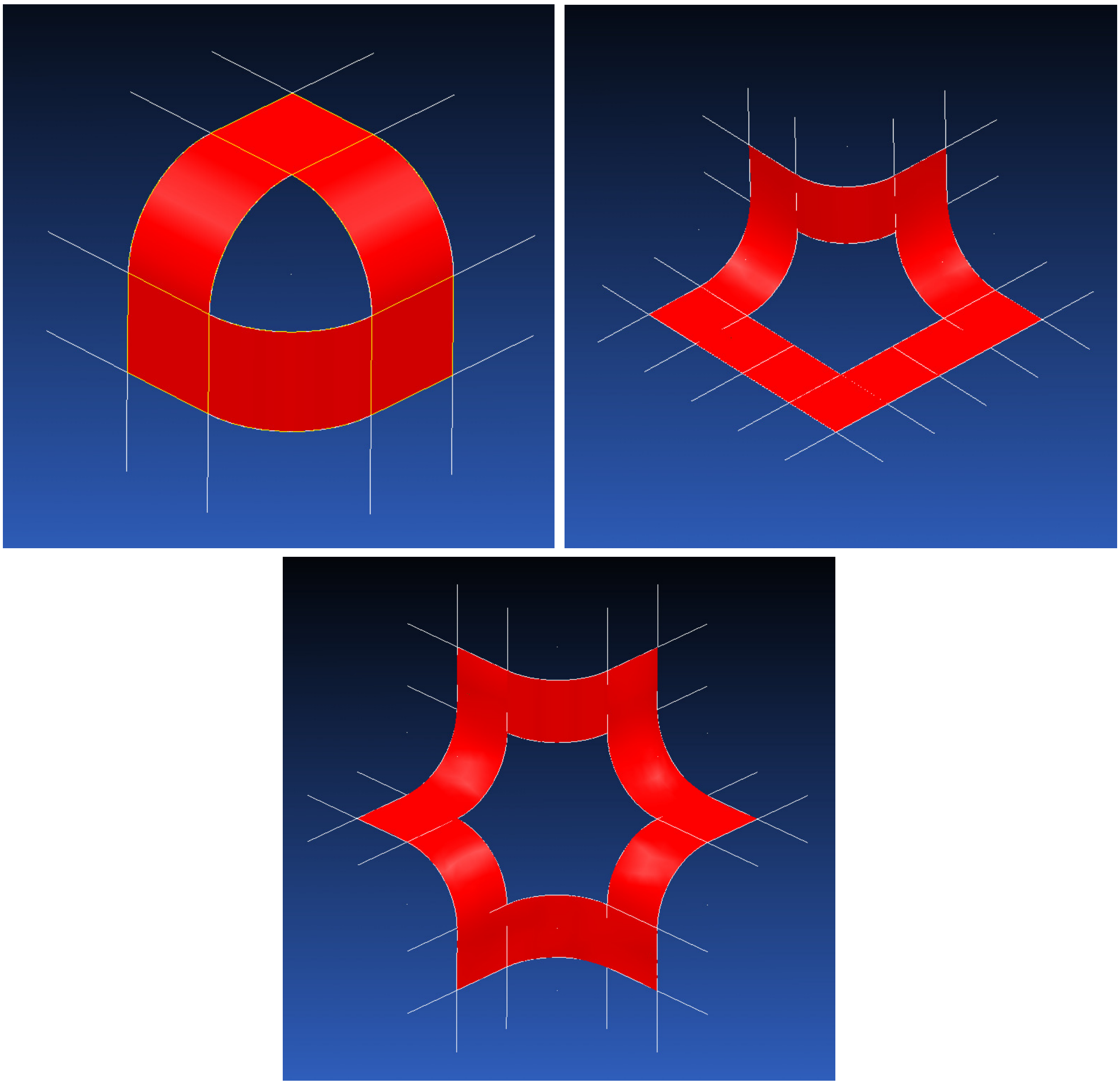


Figure 7.1 – Geometries that have areas with 3,5 and 6 sides

It is evident that being able to properly handle surfaces of arbitrary topology is advantageous. Much of the work published in this field actually defines new types of patches that have an arbitrary number of sides; these patches have non-standard parameterisations and constructions (see [Sabin, 1983], [Hosaka and Kimura, 1984], [Lee, *et. al.*, 1995], [Zheng and Ball, 1997], [Goldman, 2004]). CAD software manufacturers must specifically implement these schemes for them to be of any use, which is a significant restriction. Even when implemented, many problems can occur, i.e. when transferring CAD models to other software for analysis or machine tool path generation.

It is therefore desirable that any solution should be assembled from a collection of parametrically defined patches with rectangular topology. These patches could be originally represented using either the Bézier or B-spline basis, but any B-spline patches can be converted to Bézier. This allows a more straightforward analysis of the assembly conditions. If the B-spline surfaces contain multiple sub-patches, just the *N* sub-patches with the common central vertex are assembled; the remaining sub-patches have rectangular topology, and so it is assumed that they can be subsequently modified to attain the desired continuity with adjoining patches. The non-rational form is used, as these *N*-sided surface assemblies are to be integrated with other surfaces produced by interpolation; such surfaces very rarely employ rational form [Piegl and Tiller, 1997].

7.2 Joining Bézier patches along a common boundary

7.2.1 Parametric continuity

Two Bézier [Farin, 2002] patches $\boldsymbol{S}^A(u,v)$ and $\boldsymbol{S}^B(u,v)$, $0 \le u,v \le 1$ with control points $\boldsymbol{P}^A_{i,j}$, $0 \le i \le m_A$, $0 \le j \le n_A$, and $\boldsymbol{P}^B_{i,j}$, $0 \le i \le m_B$, $0 \le j \le n_B$, respectively, can be joined with parametric continuity along a common boundary. If the common boundary curves are $\boldsymbol{S}^{A}(1,v)$ and $\boldsymbol{S}^{B}(0,v)$, C^0 continuity is obtained when:

$$\boldsymbol{S}^{B}(0,v)=\boldsymbol{S}^{A}(1,v) \qquad 0\le v\le 1 \tag{7.1}$$

Assuming that $n_A = n_B$, C^0 continuity is attained when the common boundary control points are identical, i.e.:

$$\boldsymbol{P}^{B}_{0,j}=\boldsymbol{P}^{A}_{m_A,j} \qquad 0\le j\le n_A=n_B \tag{7.2}$$

C^1 continuity implies C^0, and in addition requires parametric derivative continuity across the boundary:

$$\boldsymbol{S}^{B}_{u}(0,v)=\boldsymbol{S}^{A}_{u}(1,v) \qquad 0\le v\le 1$$

This condition is obtained when the control points satisfy:

$$\boldsymbol{P}^{B}_{1,j}-\boldsymbol{P}^{B}_{0,j}=\boldsymbol{P}^{A}_{m_A,j}-\boldsymbol{P}^{A}_{m_A-1,j} \qquad 0\le j\le n_A=n_B \tag{7.3}$$

C^2 continuity implies C^1, and in addition requires second order parametric derivative continuity:

$$\boldsymbol{S}^{B}_{uu}(0,v)=\boldsymbol{S}^{A}_{uu}(1,v) \qquad 0\le v\le 1$$

This condition is satisfied when:

$$\boldsymbol{P}^{B}_{2,j}-2\boldsymbol{P}^{B}_{1,j}+\boldsymbol{P}^{B}_{0,j}=\boldsymbol{P}^{A}_{m_A,j}-2\boldsymbol{P}^{A}_{m_A-1,j}+\boldsymbol{P}^{A}_{m_A-2,j} \qquad 0\le j\le n_A=n_B \tag{7.4}$$

7.2.2 Geometric continuity

It is possible to relax the strong requirements of *parametric* continuity to that of *geometric* continuity. G^0 continuity requires only that the two boundary curves be common, and need not require the control points to be identical as for C^0. However, C^0 continuity (7.1) is used for simplicity of analysis, although it is noted that this does not reduce any degrees of freedom assuming that the curves have the same degree and do not represent a straight line. G^1 continuity only requires tangent continuity across the boundary, instead of requiring parametric derivative continuity. The simplest way of achieving this is to maintain the parametric direction of the cross boundary derivatives, but allow the magnitude to differ by a constant positive factor α, i.e.:

$$\boldsymbol{S}_u^B(0,v) = \alpha \boldsymbol{S}_u^A(1,v) \qquad 0 \le v \le 1 \qquad (7.5)$$

This condition is attained when the control points satisfy the relationship:

$$\boldsymbol{P}_{1,j}^B - \boldsymbol{P}_{0,j}^B = \alpha\left(\boldsymbol{P}_{m_A,j}^A - \boldsymbol{P}_{m_A-1,j}^A\right) \qquad 0 \le j \le n_A = n_B \qquad (7.6)$$

It is noted that C^1 always implies G^1, assuming $\left\|\boldsymbol{S}_u^B(0,v)\right\| > 0$, but the converse is not always true.

In a similar fashion, G^2 does not require second derivative continuity, rather curvature continuity across the boundary. The simplest way to achieve this is given by Barsky and DeRose [1989]:

$$\boldsymbol{S}_{uu}^B(0,v) = \alpha^2 \boldsymbol{S}_{uu}^A(1,v) + \eta \boldsymbol{S}_u^A(1,v) \qquad 0 \le v \le 1$$

where α is from (7.5), and η is an arbitrary constant. This leads to the control point condition:

$$\boldsymbol{P}_{2,j}^B - 2\boldsymbol{P}_{1,j}^B + \boldsymbol{P}_{0,j}^B = \alpha^2\left(\boldsymbol{P}_{m_A,j}^A - 2\boldsymbol{P}_{m_A-1,j}^A + \boldsymbol{P}_{m_A-2,j}^A\right) + \frac{\eta}{m_A - 1}\left(\boldsymbol{P}_{m_A,j}^A - \boldsymbol{P}_{m_A-1,j}^A\right)$$
$$0 \le j \le n_A = n_B \qquad (7.7)$$

However, geometric continuity does not require that the direction of any parametric derivatives be maintained; tangent and curvature continuity can be preserved even when two patches join such that their parameter lines do not have tangent continuity. This is easily seen to be true by considering two planes that meet as in Figure 7.2. They must have both tangent and curvature continuity at all points along the boundary, yet their parameter lines join at an angle.

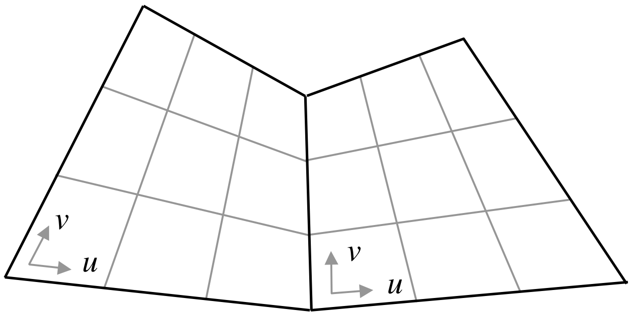


Figure 7.2 – Two planes that join with G^2, but not parametric, continuity

Kahmann [1983] proposed a general algebraic method for joining two surfaces together along a common boundary, with G^1 and then G^2 continuity. The mathematical continuity conditions are applied to Bézier patches, resulting in a set of relationships between the control points on adjoining patches. The formulation assumes that one patch, ***A***, is predefined, and that the adjoining patch, ***B***, is modified to attain the required continuity. Alternative formulations can be constructed where ***A*** is modified and ***B*** predefined, or both ***A*** and ***B*** are modified, although as neither increase the chances of obtaining a successful assembly, these formulations are not considered.

To obtain G^0 continuity between two patches, Kahmann assumes that the row of common control points between two adjacent patches must be identical. It is noted that this is actually the C^0 condition, but does not reduce the freedom in the system unless the boundary is a straight line. The next row of control points on patch ***B*** must be modified for G^1 continuity, i.e. control points $\boldsymbol{P}^B_{1,j}$, $0 \le j \le n_A = n_B$. The third row, control points $\boldsymbol{P}^B_{2,j}$, $0 \le j \le n_A = n_B$, are required for G^2 continuity. On patch ***A***, control points $\boldsymbol{P}^A_{i,j}$, $m_A - 2 \ge i \ge m_A$, $0 \le j \le n_A = n_B$, are predefined and must not be changed after the control points on patch ***B*** have been determined. Control points $\boldsymbol{P}^A_{i,j}$, $i < m_A - 2$, $0 \le j \le n_A = n_B$, and $\boldsymbol{P}^B_{i,j}$, $i > 2$, $0 \le j \le n_A = n_B$, have no influence on the continuity. See Figure 7.3.

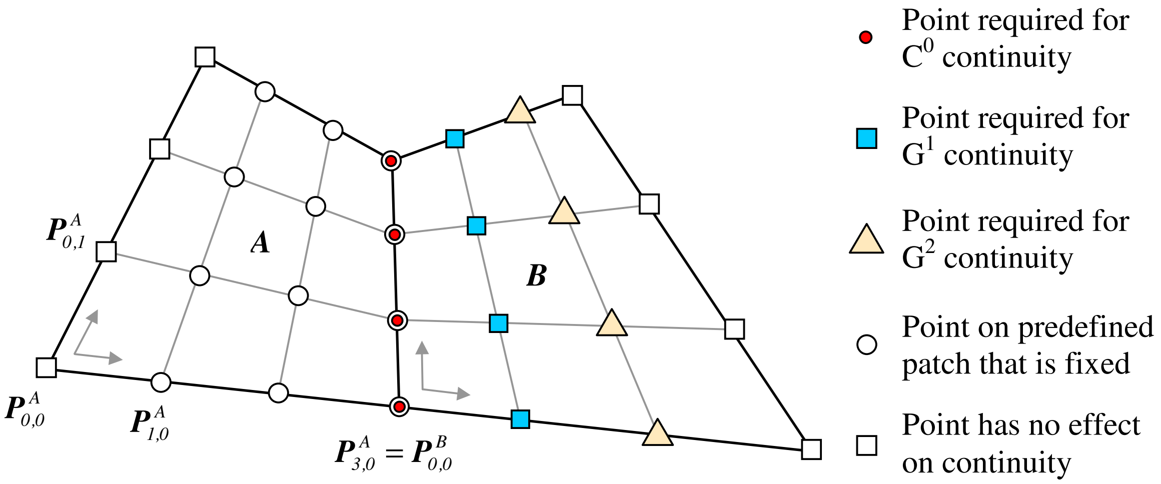

$P^A_{0,1}$
A
B
$P^A_{0,0}$
$P^A_{1,0}$
$P^A_{3,0} = P^B_{0,0}$
Point required for C^0 continuity
Point required for G^1 continuity
Point required for G^2 continuity
Point on predefined patch that is fixed
Point has no effect on continuity

Figure 7.3 – The control points affecting different levels of continuity when joining two patches

For G^1 continuity between the two patches, Kahmann assumes both patches must have coincident tangent planes at every point on the common boundary, as well as (7.1). Kahmann [1983] uses the following conditions:

$$\boldsymbol{S}_u^B(0,v) = p(v)\,\boldsymbol{S}_u^A(1,v) \; + \; q(v)\,\boldsymbol{S}_v^A(1,v) \qquad 0 \le v \le 1 \qquad (7.8)$$

where $p(v)$ is a positive constant, and $q(v)$ is a linear function. This condition implies that the unit normal must be common at all points along the boundary. Defining $p(v) = \alpha > 0$, as in (7.6), and $q(v) = \beta(1-v) + \gamma v$, where β, γ are arbitrary constants, the control point condition for G^1 continuity can be derived by comparing coefficients in (7.8):

$$\boldsymbol{P}_{1,j}^B = \alpha\left(\boldsymbol{P}_{m_A,j}^A - \boldsymbol{P}_{m_A-1,j}^A\right) + \beta\frac{n_A - j}{m_A}\left(\boldsymbol{P}_{m_A,j+1}^A - \boldsymbol{P}_{m_A,j}^A\right) + \gamma\frac{j}{m_A}\left(\boldsymbol{P}_{m_A,j}^A - \boldsymbol{P}_{m_A,j-1}^A\right) + \boldsymbol{P}_{0,j}^B$$

$$0 \le j \le n_A = n_B \qquad (7.9)$$

It is noted that (7.9) is equivalent to (7.6) when $\beta = \gamma = 0$, and also equivalent to (7.3) when $\alpha = 1, \beta = \gamma = 0$, indicating C^1 $\subset$ G^1.

For G^2 continuity between the two patches, the principal directions and curvatures must coincide at all points along the boundary, in addition to (7.1) and (7.8). Kahmann [1983] states that an equivalent formulation requires that the Dupin indicatrices must also coincide at all points along the common boundary, leading to the following condition:

$$\begin{aligned}&\boldsymbol{S}^{B}_{uu}(0,v)=p(v)^2\,\boldsymbol{S}^{A}_{uu}(1,v)\;+2p(v)q(v)\,\boldsymbol{S}^{A}_{uv}(1,v)\;+\;q(v)^2\,\boldsymbol{S}^{A}_{vv}(1,v)\\&+\;r(v)\boldsymbol{S}^{A}_{u}(1,v)+\;s(v)\boldsymbol{S}^{A}_{v}(1,v)\qquad\qquad 0\le v\le 1\end{aligned}$$

where $r(v)$ is a constant, and $s(v)$ is a linear function [Kahmann, 1983]. For simplicity, Kahmann assumes $r(v)=s(v)=0$, which yields the G^2 continuity condition in terms of the Bézier control points:

$$\begin{aligned}\boldsymbol{P}^{B}_{2,j}&=\alpha^2\left(\boldsymbol{P}^{A}_{m_A,j}-2\boldsymbol{P}^{A}_{m_A-1,j}+\boldsymbol{P}^{A}_{m_A-2,j}\right)\\&-2\alpha\beta\left(\frac{n_A-j}{m_A-1}\right)\left(\boldsymbol{P}^{A}_{m_A,j+1}-\boldsymbol{P}^{A}_{m_A,j}-\boldsymbol{P}^{A}_{m_A-1,j+1}+\boldsymbol{P}^{A}_{m_A-1,j}\right)\\&-2\alpha\gamma\left(\frac{j}{m_A-1}\right)\left(\boldsymbol{P}^{A}_{m_A,j}-\boldsymbol{P}^{A}_{m_A,j-1}-\boldsymbol{P}^{A}_{m_A-1,j}+\boldsymbol{P}^{A}_{m_A-1,j-1}\right)\\&+\beta^2\frac{(n_A-j)(n_A-j-1)}{m_A(m_A-1)}\left(\boldsymbol{P}^{A}_{m_A,j+2}-2\boldsymbol{P}^{A}_{m_A,j+1}+\boldsymbol{P}^{A}_{m_A,j}\right)\\&+2\beta\gamma\frac{(n_A-j)\,j}{m_A(m_A-1)}\left(\boldsymbol{P}^{A}_{m_A,j+1}-2\boldsymbol{P}^{A}_{m_A,j}+\boldsymbol{P}^{A}_{m_A,j-1}\right)\\&+\gamma^2\frac{j(j-1)}{m_A(m_A-1)}\left(\boldsymbol{P}^{A}_{m_A,j}-2\boldsymbol{P}^{A}_{m_A,j-1}+\boldsymbol{P}^{A}_{m_A,j-2}\right)\\&+2\boldsymbol{P}^{B}_{1,j}-\boldsymbol{P}^{B}_{0,j},\\0\le j\le n_A&=n_B\end{aligned}\tag{7.10}$$

It is noted that (7.10) is equivalent to (7.7) when $\beta=\gamma=\eta=0$, and also equivalent to (7.4) when $\alpha=1,\beta=\gamma=0$, indicating C^2 $\subset$ G^2. It is also noted that assuming $r(v)=s(v)=0$ reduces the degrees of freedom in the system.

7.3 Joining patches within an arbitrary topology region

7.3.1 Configuration of patches

There are two common approaches for joining patches together to form an arbitrary topology region, illustrated in Figure 7.4

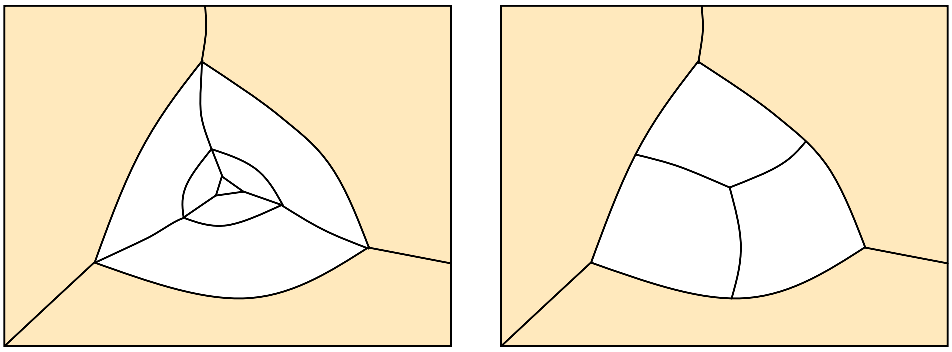
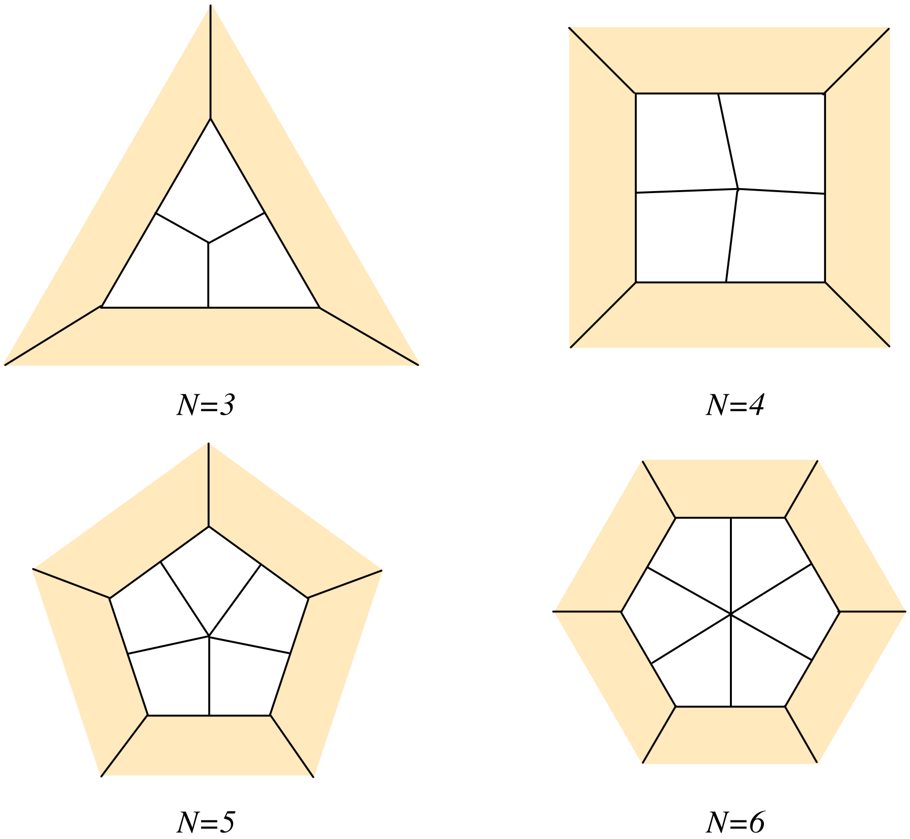


Figure 7.4 – Two methods to fill an arbitrary topology region with rectangular patches

The first method is a recursive method that fits a strip of rectangular patches around the arbitrary topology region, leaving another smaller arbitrary topology region in the middle. After several recursions, the 'hole' becomes smaller than a given tolerance, and can be ignored. The second method directly fills the arbitrary topology region without recursion by assembling patches that all join at a single point within the region. The focus of this research is based around the latter method, as the former can lead to data proliferation, which is very undesirable.

The starting point for the method, therefore, is a collection of *N* patches in one of the forms illustrated in Figure 7.5. It is impossible for these patches to all join together with parametric continuity when $N \neq 4$ without forcing at least one patch to have an angle between *u* and *v* of $0°$ or $180°$ at the central vertex. However, if the patches join with geometric continuity they need only meet with tangent and curvature continuity, allowing the parameter lines on adjacent patches to meet at an angle.



N=3

N=4

N=5

N=6

Figure 7.5 – Configuration of patches for $N = 3, 4, 5$, and 6

A restriction is placed on the geometry of each patch within the assembly such that the angle between the *u* and *v* parametric derivatives at the central vertex is greater than 0° and less than 180°. This ensures that the normal is well defined at the centre, which must be common to all patches. When $N = 2$, at least one patch has a reflex or collinear angle between parametric derivatives at the mid-point, as illustrated in Figure 7.6, thus only $N \geq 3$ is considered.

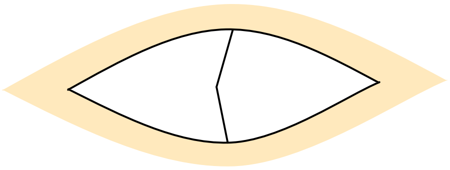


Figure 7.6 – Configuration for $N = 2$

Each of the N-sided regions is filled with N patches. It is worth noting that the $N = 4$ configuration illustrated is often superfluous, as a single patch also has four sides and would guarantee G^2 (in fact C^2) continuity everywhere within the boundaries without the need for assembly. Likewise, the $N = 6$ configuration could be replaced with 3 patches; this is outlined in Figure 7.7. Note that there are two different configurations possible, which will almost certainly result in different surfaces. Any even N-sided surface assembly, $N \geq 6$, in the form of those in Figure 7.2 can be replaced with $N/2$ patches in the form of those in Figure 7.7.

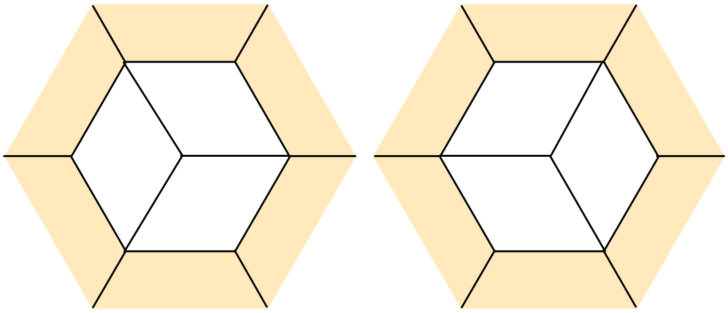


Figure 7.7 – Alternative configurations with 3 patches N=6

It is assumed that the patches are fully defined (i.e. position of all control points known), but need to be modified to attain the desired level of internal continuity. This is because the patches are most likely to be constructed by interpolation. The corner control points of each patch are fixed, and all others are allowed to move; however, how they move will be subject to constraints.

7.3.2 Patch orientation

Irrespective of the particular configuration employed, the Bézier patches are constructed such that the parametric origin for each patch is at the central vertex, and the u-direction of any patch corresponds to the v-direction of the adjacent patch along the same boundary. Such a construction requires (7.2), (7.9) and (7.10) to be reformulated, but has the advantage that each patch then joins to its neighbour in exactly the same way. This configuration is illustrated for the case $N=3$ in Figure 7.8.

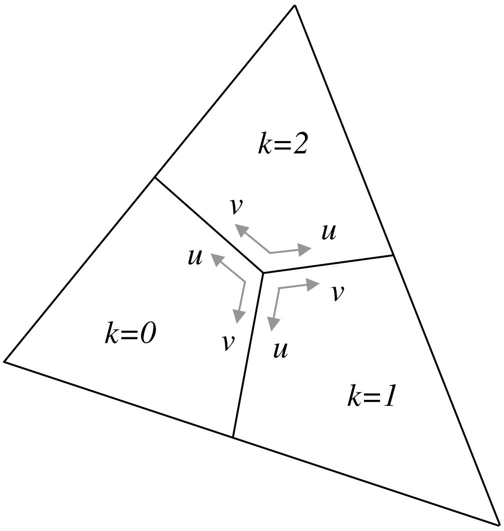


Figure 7.8 – Construction of Bézier patches to form an arbitrary topology

The control points for the patches are denoted as $\boldsymbol{P}_{i,j}^{k}$, where the subscripts i and j represent the indices of the control point in the parametric u and v-directions respectively, and the superscript k represents the patch number, $0 \le k < N$. For convenience when joining adjacent patches, the degree along the common boundary is required to be the same for both patches, i.e.:

$$m_k = n_c,\ 0 \le k < N,\ c = k-1\ (mod\ N)$$

It is noted that all patches within the assembly could be degree elevated, in both parametric directions, to the highest degree in the collection of N patches without any compromise in geometry. However, this is not a requirement for assembly, and does not increase the chances of a successful construction.

7.3.3 Initial constraints

To ensure the overall geometry of the patches is maintained, the corner vertices of each patch, $\boldsymbol{P}_{0,0}^{k}$, $\boldsymbol{P}_{0,n_k}^{k}$, $\boldsymbol{P}_{m_k,0}^{k}$, and $\boldsymbol{P}_{m_k,n_k}^{k}$, $0 \le k < N$, are not modified.

To enforce C^0 continuity along the internal boundaries, each pair of corresponding boundary points are constrained such that they are identical:

$$\boldsymbol{P}^k_{i,0} = \boldsymbol{P}^c_{0,i} \qquad 0 \le k < N,\ 0 \le i \le m_k,\ c = k - 1\ (mod\ N) \tag{7.11}$$

For G^1 continuity, all the patches must meet with tangent plane continuity at the central vertex. This constraint can be expressed in terms of the control points:

$$\boldsymbol{P}^k_{0,1} - \boldsymbol{P}_{0,0} = -\alpha_k \left(\boldsymbol{P}^c_{1,0} - \boldsymbol{P}_{0,0}\right) + \beta_k \frac{n_c}{m_c}\left(\boldsymbol{P}^c_{0,1} - \boldsymbol{P}_{0,0}\right) \tag{7.12}$$

$$0 \le k < N,\ c = k - 1\ (mod\ N)$$

where α_k and β_k are numerical constants. (7.11) is derived from (7.9), with $j = 0$, but is reformulated to allow for the change in parametric origin.

For assemblies using *N* patches, the boundary of the *N*-sided region must consist of *N* G^1 curves, otherwise the surface is not guaranteed to have *N* sides. This constraint can be expressed in terms of the control points:

$$\boldsymbol{P}^k_{m_k,1} - \boldsymbol{P}^c_{0,n_c} = -\alpha_k \left(\boldsymbol{P}^c_{1,n_c} - \boldsymbol{P}^c_{0,n_c}\right) \qquad 0 \le k < N,\ c = k - 1\ (mod\ N) \tag{7.13}$$

where α_k are the same as those in (7.12). (7.13) is also derived from (7.9), with $j = n_c$, and $\gamma_k = 0$, $0 \le k < N$, again reformulated to allow for the change in parametric origin.

No continuity conditions are imposed with regard to surfaces outside the boundary, except the C^0 condition at patch corners. Patches that join with the *N*-sided surface assembly must be subsequently constructed with the appropriate level of continuity.

7.4 Adapting Kahmann's method for arbitrary topologies

After the initial constraints have been imposed, the geometric continuity conditions (7.2), (7.9) and (7.10) are required. When they are applied to arbitrary topologies, a closed loop will be formed; see Figure 7.9. The assumption that one patch can be predefined whilst another is modified to obtain the required continuity is no longer valid because there are cyclic dependencies on certain control points. To obtain C^0 between all adjacent patches within the arbitrary topology region, all control points on adjacent boundaries must be identical, i.e. (7.11). The second row of control points must be modified for G^1 continuity, i.e. control points $\boldsymbol{P}^k_{i,1}$, $0 \le k < N$, $0 \le i \le m_k$. The third row, control points $\boldsymbol{P}^k_{i,2}$, $0 \le k < N$, $0 \le i \le m_k$, are required for G^2 continuity.

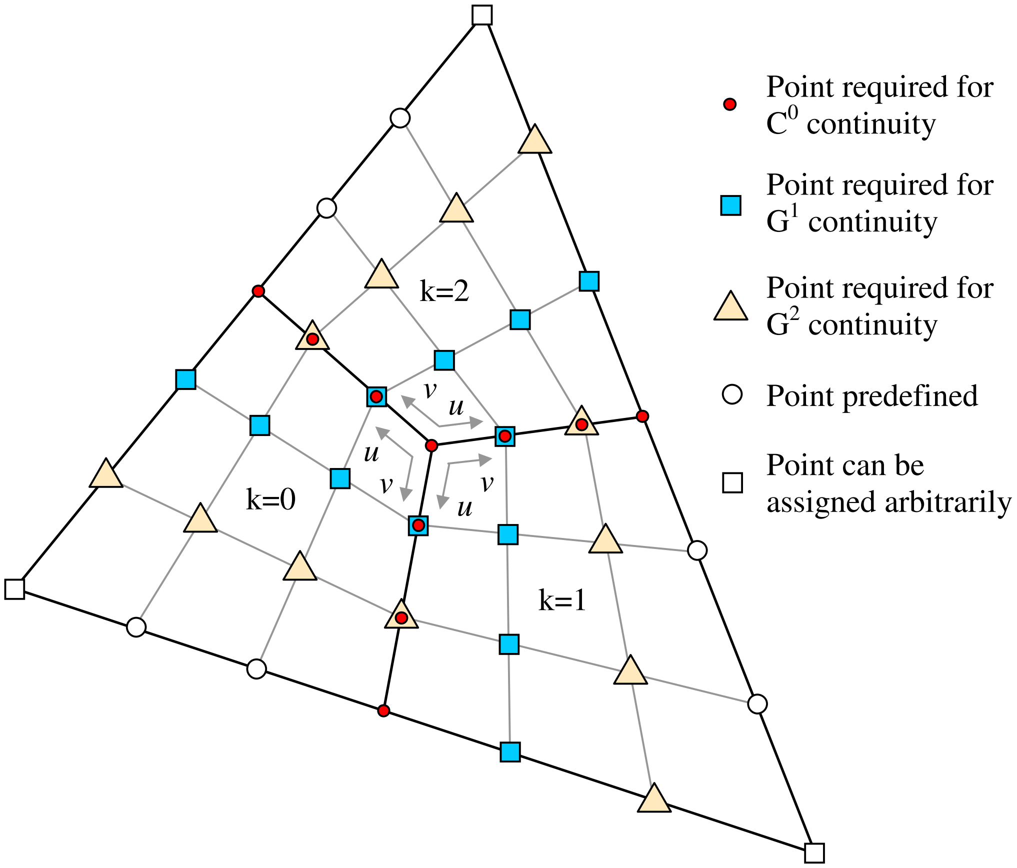

k=2
k=0
k=1
v
u
u
v
v
u
● Point required for C⁰ continuity
■ Point required for G¹ continuity
△ Point required for G² continuity
○ Point predefined
□ Point can be assigned arbitrarily

Figure 7.9 – The control points affecting different levels of continuity, N=3

Every patch, index *k*, has to be considered as both the predefined patch when calculating the control points on patch *k+1*, and the patch to be modified to achieve continuity with patch $k-1\,(mod\ N)$. As a consequence, certain groups of points need to be determined by systems of linear equations. If linear equations are consistent, there may be a unique or infinite number of solutions. Control points $\boldsymbol{P}_{i,j}^{k}$, $0 \le k < N$, $i > 2$, $j > 2$, have no influence on the continuity, and can be assigned arbitrarily.

7.5 Modifying Kahmann's conditions to achieve G^1 continuity

The starting point for assembling a collection of patches with G^1 continuity to create an *N*-sided surface is one of the forms in Figure 7.5. The patch corner points must be fixed, and C^0 continuity along boundaries must be enforced (7.11). Two other initial conditions are imposed: (7.12) ensures that there is a common normal at the central vertex, and (7.13) ensures that the *N*-sided surface boundary consists of N G^1 curves.

Kahmann's condition for G^1 continuity (7.9) must be reformulated to be compatible with the change made to the parametric origin of each patch. It becomes:

$$\boldsymbol{P}_{j,1}^{k} = -\alpha_k \left(\boldsymbol{P}_{1,j}^{c} - \boldsymbol{P}_{0,j}^{c}\right) + \beta_k \frac{n_c - j}{m_c}\left(\boldsymbol{P}_{0,j+1}^{c} - \boldsymbol{P}_{0,j}^{c}\right) + \gamma_k \frac{j}{m_c}\left(\boldsymbol{P}_{0,j}^{c} - \boldsymbol{P}_{0,j-1}^{c}\right) + \boldsymbol{P}_{0,j}^{c},$$

$$0 \le k < N,\ 0 \le j \le n_c,\ m_k = n_c,\ c = k - 1 \pmod{N} \qquad (7.14)$$

It is noted that (7.14) simplifies to the initial constraints (7.12) and (7.13) when $j = 0$, and $j = n_c$, respectively. All control points beyond the second row, i.e. $\boldsymbol{P}_{i,j}^{k}$, $0 \le k < N$, $0 \le i \le m_k$, $2 \le j \le n_k$, can be assigned arbitrarily, as they have no effect on the continuity between adjacent patches.

Before (7.14) can be used to determine any control points, α_k, β_k and γ_k, $0 \le k < N$, must be determined. Recalling that the unit normal at the central vertex must be common to all patches, $\boldsymbol{P}_{0,0}$ and $\boldsymbol{P}_{0,1}^{k}$, $0 \le k < N$, must lie in a plane. $\boldsymbol{P}_{0,1}^{k}$ can be chosen to maintain the desired normal thereby defining α_k and β_k. γ_k, $0 \le k < N$, can be defined such that the direction of $\left(\boldsymbol{P}^k_{m_k,1} - \boldsymbol{P}^k_{m_k,0}\right)$, $0 \le k < N$, is maintained and the initial constraint (7.13) satisfied; the magnitude cannot be prescribed because the G^1 condition completely defines $\boldsymbol{P}^k_{m_k,1}$. If N patches are used, then $\gamma_k = 0$, $0 \le k < N$, ensures that the assembly of N patches remains N sided.

$\boldsymbol{P}^k_{1,1}$, $0 \le k < N$ must now be found; however, $\boldsymbol{P}^k_{1,1}$ is a function of $\boldsymbol{P}^c_{1,1}$, $c = k - 1 \ (mod \ N)$, so they can not be solved independently. See Figure 7.10.

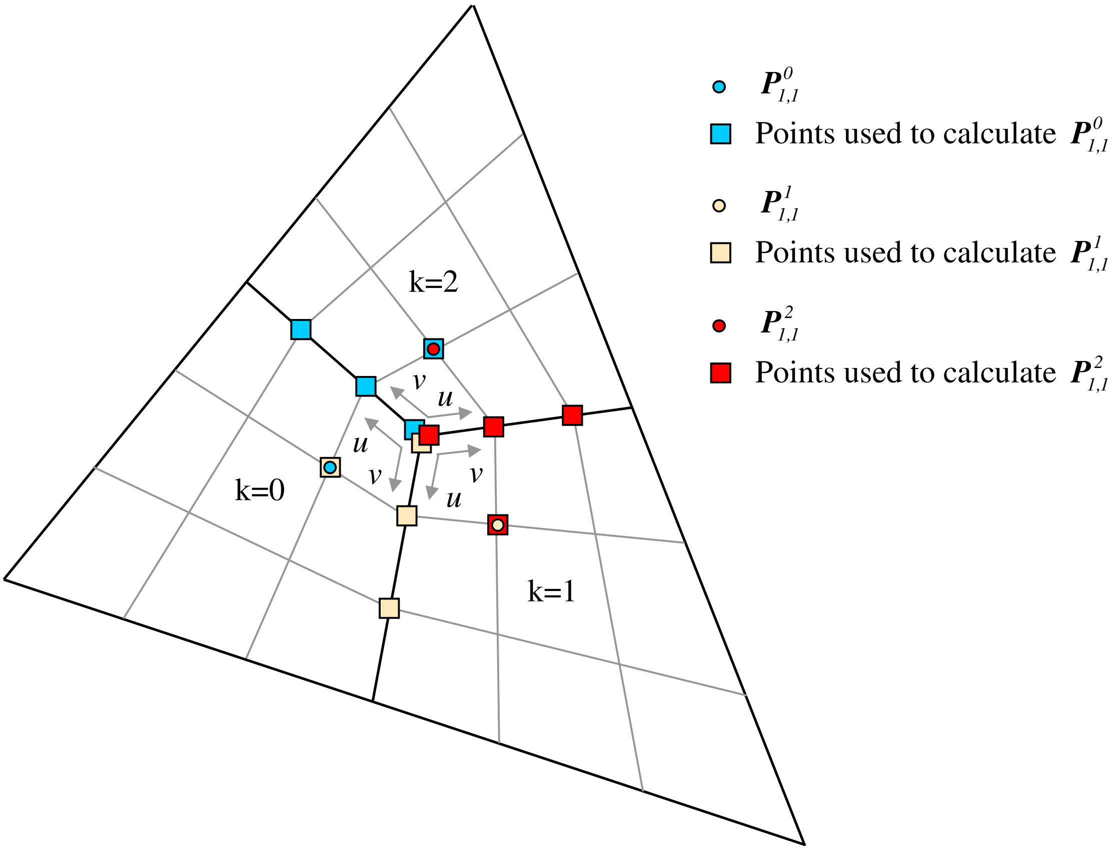

● $\boldsymbol{P}^0_{1,1}$
■ Points used to calculate $\boldsymbol{P}^0_{1,1}$
○ $\boldsymbol{P}^1_{1,1}$
□ Points used to calculate $\boldsymbol{P}^1_{1,1}$
● $\boldsymbol{P}^2_{1,1}$
■ Points used to calculate $\boldsymbol{P}^2_{1,1}$
k=2
k=0
k=1
v
u
u
v
v
u

Figure 7.10 – Control points used when calculating $\boldsymbol{P}^k_{1,1}$, $N = 3$

To solve for $\boldsymbol{P}^k_{1,1}$, a system of N linear equations is required:

$$\boldsymbol{P}^k_{1,1} + \alpha_k \boldsymbol{P}^c_{1,1} = \boldsymbol{E}_c = \alpha_k \boldsymbol{P}^c_{0,1} + \beta_k \left(\frac{n_c - 1}{m_c}\right)\left(\boldsymbol{P}^c_{0,2} - \boldsymbol{P}^c_{0,1}\right) + \frac{\gamma_k}{m_c}\left(\boldsymbol{P}^c_{0,1} - \boldsymbol{P}^c_{0,0}\right) + \boldsymbol{P}^c_{0,1}$$

$$0 \le k < N, \ c = k - 1 \ (mod \ N)$$

For $N=3$, the system of equations can be written as:

$$\begin{bmatrix} 1 & 0 & \alpha_0 \\ \alpha_1 & 1 & 0 \\ 0 & \alpha_2 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{P}_{1,1}^{0} \\ \boldsymbol{P}_{1,1}^{1} \\ \boldsymbol{P}_{1,1}^{2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{E}_2 \\ \boldsymbol{E}_0 \\ \boldsymbol{E}_1 \end{bmatrix}$$

This system of equations has the determinant:

$$\begin{vmatrix} 1 & 0 & \alpha_0 \\ \alpha_1 & 1 & 0 \\ 0 & \alpha_2 & 1 \end{vmatrix} = 1 + \alpha_0\alpha_1\alpha_2 = 2$$

because $\alpha_0\alpha_1\alpha_2 = 1$ (Appendix D). Therefore, a solution is always guaranteed.

For $N=4$, the system of equations has the determinant:

$$\begin{vmatrix} 1 & 0 & 0 & \alpha_0 \\ \alpha_1 & 1 & 0 & 0 \\ 0 & \alpha_2 & 1 & 0 \\ 0 & 0 & \alpha_3 & 1 \end{vmatrix} = 1 - \alpha_0\alpha_1\alpha_2\alpha_3 = 0$$

because experimental evidence suggests $\alpha_0\alpha_1\alpha_2\alpha_3 = 1$. Therefore, the system of linear equations either has no solution, or an infinite number of solutions. It is noted that at least one solution must be possible, when $\alpha_k = 1$, $\beta_k = 0$ and $\gamma_k = 0$, $0 \le k < N$, as this is the condition for C^1 continuity, which is a special subset of G^1. Further research is required to classify all the geometries that lead to an under-specified system, and will therefore yield a solution.

For $N=5$, the system of equations has the determinant:

$$\begin{vmatrix} 1 & 0 & 0 & 0 & \alpha_0 \\ \alpha_1 & 1 & 0 & 0 & 0 \\ 0 & \alpha_2 & 1 & 0 & 0 \\ 0 & 0 & \alpha_3 & 1 & 0 \\ 0 & 0 & 0 & \alpha_4 & 1 \end{vmatrix} = 1 + \alpha_0\alpha_1\alpha_2\alpha_3\alpha_4 = 2$$

because experimental evidence suggests $\alpha_0\alpha_1\alpha_2\alpha_3\alpha_4 = 1$. Therefore, a solution is always guaranteed. After observing the trend for $N = 3,4,5,6,7$, it is noted that any arbitrary topology region with an odd *N* should be solvable, and any arbitrary topology region with an even *N* will have a determinant$=0$ for $\boldsymbol{P}_{1,1}^{k}$.

Assuming a solution can be found for $\boldsymbol{P}_{1,1}^{k}$, the next step is to calculate $\boldsymbol{P}_{i,1}^{k}$, $0 \le k < N$, $2 \le i < m_k$. This is achieved by simply applying the G^1 condition (7.14), and completes the construction of *k* patches to form a G^1 arbitrary topology region.

7.6 Modifying Kahmann's conditions to achieve G^2 continuity

To achieve G^2 continuity between two adjacent patches, Kahmann [1983] requires both the tangent planes (for G^1) and the Dupin indicatrices (for G^2) of both patches to coincide at every point along the common boundary curve. The method assumes that moving control points to attain G^2 continuity on one patch will not affect the other; however, when assembling patches that join up in a closed loop this assumption is no longer valid. Certain control points that were predefined for the G^1 construction now need to become variables, therefore the starting point for the G^2 method is a collection of *N* patches in one the forms illustrated in Figure 7.5.

The condition for G^2 (7.10) must be reformulated to be compatible with the changes made to the parametric origin of each patch. In the original derivation, Kahmann assumed $r(v) = s(v) = 0$ for simplicity. To maximise the chances of finding a solution, any such simplifications have been removed. Defining $r(v) = \eta$, and $s(v) = (1-v)\sigma + v\upsilon$ yields three additional terms with coefficients η_k, σ_k, and υ_k. Reformulating and generalising yields:

$$
\begin{aligned}
\boldsymbol{P}^{k}_{j,2} = {} & +\alpha_k^2\left(\boldsymbol{P}^{c}_{2,j}-2\boldsymbol{P}^{c}_{1,j}+\boldsymbol{P}^{c}_{0,j}\right)-2\alpha_k\beta_k\left(\frac{n_c-j}{m_c-1}\right)\left(\boldsymbol{P}^{c}_{1,j+1}-\boldsymbol{P}^{c}_{1,j}-\boldsymbol{P}^{c}_{0,j+1}+\boldsymbol{P}^{c}_{0,j}\right)\\
& -2\alpha_k\gamma_k\left(\frac{j}{m_c-1}\right)\left(\boldsymbol{P}^{c}_{1,j}-\boldsymbol{P}^{c}_{1,j-1}-\boldsymbol{P}^{c}_{0,j}+\boldsymbol{P}^{c}_{0,j-1}\right)\\
& +\beta_k^2\frac{(n_c-j)(n_c-j-1)}{m_c(m_c-1)}\left(\boldsymbol{P}^{c}_{0,j+2}-2\boldsymbol{P}^{c}_{0,j+1}+\boldsymbol{P}^{c}_{0,j}\right)\\
& +2\beta_k\gamma_k\frac{(n_c-j)\,j}{m_c(m_c-1)}\left(\boldsymbol{P}^{c}_{0,j+1}-2\boldsymbol{P}^{c}_{0,j}+\boldsymbol{P}^{c}_{0,j-1}\right)\\
& +\gamma_k^2\frac{j(j-1)}{m_c(m_c-1)}\left(\boldsymbol{P}^{c}_{0,j}-2\boldsymbol{P}^{c}_{0,j-1}+\boldsymbol{P}^{c}_{0,j-2}\right)\\
& -\frac{\eta_k}{(m_c-1)}\left(\boldsymbol{P}^{c}_{1,j}-\boldsymbol{P}^{c}_{0,j}\right)+\sigma_k\frac{(n_c-j)}{m_c(m_c-1)}\left(\boldsymbol{P}^{c}_{0,j+1}-\boldsymbol{P}^{c}_{0,j}\right)\\
& +\frac{\upsilon_k j}{m_c(m_c-1)}\left(\boldsymbol{P}^{c}_{0,j}-\boldsymbol{P}^{c}_{0,j-1}\right)\\
& +2\boldsymbol{P}^{k}_{j,1}-\boldsymbol{P}^{k}_{j,0}
\end{aligned}
$$

$$0\le k<N,\ 0\le j\le n_c,\ m_k=n_c,\ c=k-1\ (mod\ N) \qquad (7.15)$$

As before, the corner vertices of each patch are fixed, i.e. control points $\boldsymbol{P}^{k}_{0,0}$, $\boldsymbol{P}^{k}_{0,n_k}$, $\boldsymbol{P}^{k}_{m_k,0}$, $\boldsymbol{P}^{k}_{m_k,n_k}$, $0\le k<N$, and C^0 continuity is enforced along internal boundaries (7.11). Constraints are imposed such that the normal at the centre is common to all patches (7.12), and the *N*-sided surface assembly has *N* G^1 sides (7.13). All control points beyond the third row, i.e. $\boldsymbol{P}^{k}_{i,j}$, $0\le k<N$, $0\le i\le m_k$, $3\le j\le n_k$, can be assigned arbitrarily, as they have no affect on the continuity between adjacent patches.

$\boldsymbol{P}^{k}_{0,1}$, $0 \le k < N$, are solved in the same way as for G^1. The implications of the patches joining in a closed loop are greater for G^2 than G^1 because the dependencies on each control point are increased. $\boldsymbol{P}^{k}_{1,1}$, $0 \le k < N$, is now a function of $\boldsymbol{P}^{c}_{1,1}$ and $\boldsymbol{P}^{c}_{0,2}$, $c = k - 1 \ (mod \ N)$, so they must all be solved simultaneously. See Figure 7.11.

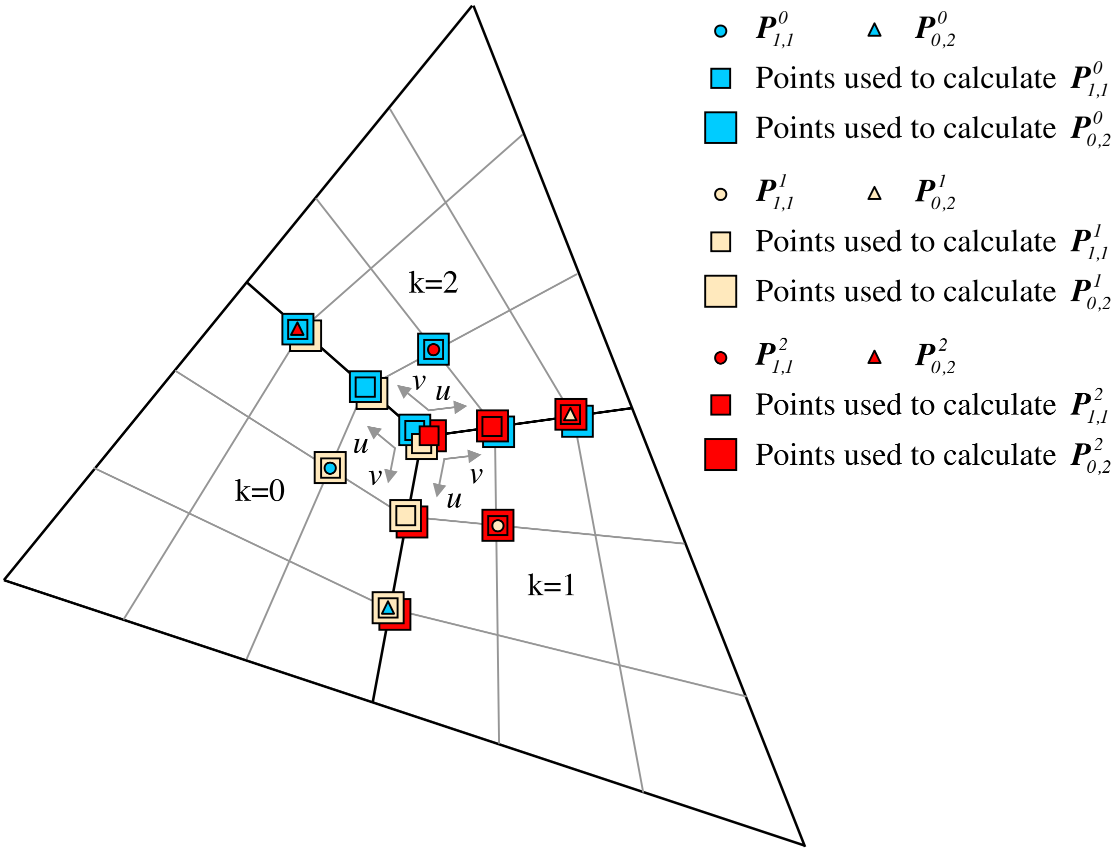

k=2
k=0
k=1
v
u
u
v
v
u
○ $\boldsymbol{P}^{0}_{1,1}$ △ $\boldsymbol{P}^{0}_{0,2}$
□ Points used to calculate $\boldsymbol{P}^{0}_{1,1}$
□ Points used to calculate $\boldsymbol{P}^{0}_{0,2}$
○ $\boldsymbol{P}^{1}_{1,1}$ △ $\boldsymbol{P}^{1}_{0,2}$
□ Points used to calculate $\boldsymbol{P}^{1}_{1,1}$
□ Points used to calculate $\boldsymbol{P}^{1}_{0,2}$
● $\boldsymbol{P}^{2}_{1,1}$ ▲ $\boldsymbol{P}^{2}_{0,2}$
🟥 Points used to calculate $\boldsymbol{P}^{2}_{1,1}$
🟥 Points used to calculate $\boldsymbol{P}^{2}_{0,2}$

Figure 7.11 – Control points used when calculating $\boldsymbol{P}^{k}_{1,1}$ and $\boldsymbol{P}^{k}_{0,2}$, $N=3$

Setting up a system of $2N$ linear equations using (7.14) as the condition on $\boldsymbol{P}^{k}_{1,1}$ and (7.15) as the condition on $\boldsymbol{P}^{k}_{0,2}$ yields:

$$
\begin{aligned}
&\boldsymbol{P}^{k}_{1,1} + \alpha_k \boldsymbol{P}^{c}_{1,1} - \beta_k \left(\frac{n_c - 1}{m_c} \right) \boldsymbol{P}^{c}_{0,2} = \boldsymbol{E}_c \\
&= \quad \alpha_k \boldsymbol{P}^{c}_{0,1} - \beta_k \left(\frac{n_c - 1}{m_c} \right) \boldsymbol{P}^{c}_{0,1} + \frac{\gamma_k}{m_c} \left(\boldsymbol{P}^{c}_{0,1} - \boldsymbol{P}^{c}_{0,0} \right) + \boldsymbol{P}^{c}_{0,1} \\
&0 \le k < N ,\ c = k - 1 \ (mod\ N)
\end{aligned}
$$

and

$$
\begin{aligned}
&\boldsymbol{P}^{k}_{0,2} - \alpha_k^2 \boldsymbol{P}^{d}_{0,2} - \beta_k^2 \frac{(n_c)(n_c - 1)}{m_c (m_c - 1)} \boldsymbol{P}^{c}_{0,2} + 2\alpha_k \beta_k \left(\frac{n_c}{m_c - 1} \right) \boldsymbol{P}^{c}_{1,1} = \boldsymbol{H}_c \\
&= \quad \alpha_k^2 \left(-2\boldsymbol{P}^{c}_{1,0} + \boldsymbol{P}^{c}_{0,0} \right) - 2\alpha_k \beta_k \left(\frac{n_c}{m_c - 1} \right) \left(-\boldsymbol{P}^{c}_{1,0} - \boldsymbol{P}^{c}_{0,1} + \boldsymbol{P}^{c}_{0,0} \right) \\
&\qquad + \beta_k^2 \frac{(n_c)(n_c - 1)}{m_c (m_c - 1)} \left(-2\boldsymbol{P}^{c}_{0,1} + \boldsymbol{P}^{c}_{0,0} \right) \\
&\qquad - \frac{\eta_k}{(m_c - 1)} \left(\boldsymbol{P}^{c}_{1,0} - \boldsymbol{P}^{c}_{0,0} \right) + \sigma_k \frac{n_c}{m_c (m_c - 1)} \left(\boldsymbol{P}^{c}_{0,1} - \boldsymbol{P}^{c}_{0,0} \right) \\
&\qquad + 2\boldsymbol{P}^{k}_{0,1} - \boldsymbol{P}^{k}_{0,0} \\
&0 \le k < N ,\ c = k - 1 \ (mod\ N) ,\ d = k - 2 \ (mod\ N)
\end{aligned}
$$

For $N = 3$, the resulting system of linear equations is:

$$
\begin{bmatrix}
1 & 0 & \alpha_0 & 0 & 0 & \beta_0 \left(\frac{n_2 - 1}{m_2} \right) \\
\alpha_1 & 1 & 0 & \beta_1 \left(\frac{n_0 - 1}{m_0} \right) & 0 & 0 \\
0 & \alpha_2 & 1 & 0 & \beta_2 \left(\frac{n_1 - 1}{m_1} \right) & 0 \\
0 & 0 & 2\alpha_0 \beta_0 \left(\frac{n_2}{m_2 - 1} \right) & 1 & -\alpha_0^2 & -\beta_0^2 \frac{n_2 (n_2 - 1)}{m_2 (m_2 - 1)} \\
2\alpha_1 \beta_1 \left(\frac{n_0}{m_0 - 1} \right) & 0 & 0 & -\beta_1^2 \frac{n_0 (n_0 - 1)}{m_0 (m_0 - 1)} & 1 & -\alpha_1^2 \\
0 & 2\alpha_2 \beta_2 \left(\frac{n_1}{m_1 - 1} \right) & 0 & -\alpha_2^2 & -\beta_2^2 \frac{n_1 (n_1 - 1)}{m_1 (m_1 - 1)} & 1
\end{bmatrix}
\begin{bmatrix}
\boldsymbol{P}^{0}_{1,1} \\ \boldsymbol{P}^{1}_{1,1} \\ \boldsymbol{P}^{2}_{1,1} \\ \boldsymbol{P}^{0}_{0,2} \\ \boldsymbol{P}^{1}_{0,2} \\ \boldsymbol{P}^{2}_{0,2}
\end{bmatrix}
=
\begin{bmatrix}
\boldsymbol{E}_2 \\ \boldsymbol{E}_0 \\ \boldsymbol{E}_1 \\ \boldsymbol{H}_2 \\ \boldsymbol{H}_0 \\ \boldsymbol{H}_1
\end{bmatrix}
$$

The determinant of the coefficient matrix is:

$$\begin{vmatrix} 1 & 0 & \alpha_0 & 0 & 0 & \beta_0\left(\dfrac{n_2-1}{m_2}\right) \\ \alpha_1 & 1 & 0 & \beta_1\left(\dfrac{n_0-1}{m_0}\right) & 0 & 0 \\ 0 & \alpha_2 & 1 & 0 & \beta_2\left(\dfrac{n_1-1}{m_1}\right) & 0 \\ 0 & 0 & 2\alpha_0\beta_0\left(\dfrac{n_2}{m_2-1}\right) & 1 & -\alpha_0^2 & -\beta_0^2\dfrac{n_2(n_2-1)}{m_2(m_2-1)} \\ 2\alpha_1\beta_1\left(\dfrac{n_0}{m_0-1}\right) & 0 & 0 & -\beta_1^2\dfrac{n_0(n_0-1)}{m_0(m_0-1)} & 1 & -\alpha_1^2 \\ 0 & 2\alpha_2\beta_2\left(\dfrac{n_1}{m_1-1}\right) & 0 & -\alpha_2^2 & -\beta_2^2\dfrac{n_1(n_1-1)}{m_1(m_1-1)} & 1 \end{vmatrix}$$

After expanding this determinant and substituting the relationship $\alpha_0\alpha_1\alpha_2 = 1$ (Appendix D), all terms cancel out leaving the result of the determinant $= 0$. The system of linear equations has therefore either no solution, or an infinite number of solutions. Solving the system of linear equations using Gaussian elimination and back substitution with an arbitrary example yields a non-zero right hand side. The system of linear equations has, in general therefore, no solution, although certain geometries may lead to a zero right hand side resulting in an under-specified system. It is noted that any assumptions made with regard to $r(v)$, $s(v)$ have no bearing on this result.

For $N=4$ and $N=5$, experimental evidence suggests that $\alpha_0\alpha_1\alpha_2\alpha_3 = 1$ and $\alpha_0\alpha_1\alpha_2\alpha_3\alpha_4 = 1$ respectively. When these relationships are substituted into their corresponding systems of equations, the resulting determinants $= 0$.

It seems evident, therefore, that a collection of *N* patches cannot be assembled with G^2 continuity for the general case, although it is known that certain specific geometries will produce a zero right hand side and therefore a solution can be found in those circumstances. An example of such geometries is when $N=4$ with $\alpha_k = 1$, $\beta_k = \gamma_k = \eta_k = \sigma_k = \upsilon_k = 0$, $0 \le k < N$, as this is the condition for C^2 continuity, which is a special subset of G^2. Also, it is noted that an assembly of planar patches must join with G^2 for any *N*. The classification of all geometries that do lead to a G^2 solution is the subject of future research.

Where a G^2 solution is not possible, it may be possible to construct the assembly of patches such that the maximum discrepancy in normal curvature, ε, is minimised between all boundary curves, i.e. $G^2-\varepsilon$ continuity. This is also the subject of further research.

7.7 Summary

This chapter has demonstrated that arbitrary topology regions with an odd *N* can always be filled with *N* patches joining with G^1 internal continuity. For even *N*, the system of equations used to solve for $\boldsymbol{P}^k_{1,1}$ has a zero determinant; further work is required to classify all the geometries that lead to an under-specified system, and can therefore yield a solution.

For G^2, it has been shown that the system of linear equations used to calculate $\boldsymbol{P}^k_{1,1}$ and $\boldsymbol{P}^k_{0,2}$ will always have a zero determinant for $N = 3,4,5$. Further work is required to prove:

$$\prod_{i=0}^{N-1} \alpha_i = 1$$

for all $N > 3$, and identify any specific geometries where a G^2 solution is possible. Where it is not possible, a method needs to be developed for obtaining a G^2 solution to within a maximum normal curvature discrepancy ε, i.e. $G^2-\varepsilon$ continuity.

Chapter 8

Conclusions and further work

The primary focus of this thesis has been to determine the best methods for controlling the interpolation process. Chapter 1 provided an outline of how NUBS curves and surfaces are constructed by interpolation, and the various factors that affect the quality of the resulting interpolant. Chapter 2 gave a detailed description of the interpolation process, and summarised a variety of existing methods for controlling the parameterisation, derivative magnitude estimation, knot vector generation, parameter reconciliation and twist vector estimation.

Chapter 3 introduced a new technique, proposed by Ball [2004], which calculates the parameter values and derivative magnitudes for a spline curve. Given points and tangents, the method constructs a piecewise C^1 curve with orthogonal first and second derivatives at specified parametric locations. The parameter values and end derivative magnitudes are then used to interpolate a C^2 NUBS curve. By varying the location of where the orthogonal conditions are satisfied within each C^1 span, three algorithms were suggested, corresponding to $W = 0,1,2$. An analytical study was conducted on the new methods, and concluded that the method which satisfies the orthogonality condition at

$t = \left(3 \pm \sqrt{5}\right) / 6$ within each C^1 segment, $W = 1$, has the greatest practical interest. This method is stable for all data configurations, is independent of the angle between tangents, and does not need to restrict applications to ensure numerical robustness or reasonable performance.

All of the existing and orthogonal parameterisation methods were tested numerically in Chapter 4, along with their corresponding derivative magnitude estimation methods. The case studies highlighted that all of the methods were virtually identical for evenly spaced data sampled from a unit semi-circle; however, many of the existing methods were shown to be very poor when the data is not evenly spaced, or is sampled from profiles with non-constant curvature. The orthogonal parameterisations were seen to perform very well: the $W = 0$ and 1 parameterisations both outperformed the other methods repeatedly, including the chord length method, which Piegl and Tiller [1997] state is the most common parameterisation. Considering the analytical and numerical investigations, it was concluded that the $W = 1$ parameterisation and derivative magnitude estimation methods are the most appropriate for general interpolation.

Chapter 5 discussed how data points should be distributed to control the shape of the interpolant. It was identified that distributing points evenly by arc length along the curve under-represented areas of higher curvature, and distributing points linearly with either winding angle or curvature under-represented areas of low curvature. Three new methods were proposed, of which the constant projected distance method outperformed all others in the numerical tests conducted. The method can be applied to analytic or parametric functions, or be used for data reduction when given a dense string of points. Further work is required to reformulate this method for selecting the optimal interpolation points for a surface; it is anticipated that points will need to be sampled in strips across and along the surface, resolving the final location of each point using a geometric arc length parameterisation method such as that proposed by Chong [2006]. The chapter also provide a guideline to the number of data points required to produce an acceptable error, based on the total winding angle of the curve.

An improved method for twist vector estimation is considered in Chapter 6, building on the Adini [Barnhill, *et. al.*, 1978] method. A hypothetical relationship is presented, which states that the true twist is a linear combination of the Adini and Coons twists. This relationship is proved to be true for quadratic data, and the general error term for cubic data is evaluated. This error is shown to diminish at a faster rate than the Adini error when the data is subdivided, indicating superior performance. Numerical tests conducted on a wide range of surface types illustrate that the Improved method consistently outperforms Adini, and therefore the Improved method is recommended as the most appropriate technique for interpolating constrained surfaces.

Chapter 7 considers the problem of joining interpolated surfaces together with parametric and geometric continuity constraints. Specific attention is given to joining *N* surfaces together, with geometric continuity, to form a closed loop such that an *N*-sided surface is created. Using Kahmann's [1983] conditions for G^1 and G^2 continuity, it is shown that surfaces with odd *N* can always be joined with G^1 continuity, but only certain geometries can lead to a G^1 solution with even *N*. These conclusions rely in part on experimental evidence, which indicates that $\prod_{i=0}^{N-1} \alpha_i \equiv 1$, although it has only be proven for $N = 3$; further work is required to develop a general proof. Only certain geometries can lead to a G^2 solution for either odd or even *N*. Further work is required to classify all the geometries that can lead to G^1 and G^2 solutions and, where a solution is not possible, to identify a method for obtaining the solution with minimum error.

Appendix A

Relationship between internal Ball vertices

Given s cubic Ball segments, $\boldsymbol{B}^{j}\left(0 \le t \le 1\right)$, $0 \le j \le s-1$, with control points $\boldsymbol{P}_{i}^{j}$, $0 \le i \le 3$, $0 \le j \le s-1$, that join with C^2 continuity, fix the continuity conditions and move $\boldsymbol{P}_{1}^{0}$ (i.e. the curve's start derivative) by a vector amount Δ. The amount that $\boldsymbol{P}_{2}^{0}$ moves as a result is a function of the number of segments, s. The relationship between $\boldsymbol{P}_{1}^{0}$ and $\boldsymbol{P}_{2}^{0}$ is calculated for $s = 2,3,4$, then the general relationship is deduced.

A point on a cubic Ball segment is given by:

$$\boldsymbol{B}^{j}\left(t\right) = \left(1-2t+t^{2}\right)\boldsymbol{P}_{0}^{j} + \left(2t-4t^{2}+2t^{3}\right)\boldsymbol{P}_{1}^{j} + \left(2t^{2}-2t^{3}\right)\boldsymbol{P}_{2}^{j} + t^{2}\boldsymbol{P}_{3}^{j}$$

The continuity conditions between segments are obtained by equating $\boldsymbol{B}_{(c)}^{j}\left(1\right) = \boldsymbol{B}_{(c)}^{j+1}\left(0\right)$, $0 \le j \le s-2$, where $(c) = \{0,1,2\}$ denotes the parametric point, first derivative and second derivative respectively:

C^0: $\boldsymbol{P}_{3}^{j} = \boldsymbol{P}_{0}^{j+1}$ (A1)

C^1: $-2\boldsymbol{P}_{2}^{j} + 2\boldsymbol{P}_{3}^{j} = -2\boldsymbol{P}_{0}^{j+1} + 2\boldsymbol{P}_{1}^{j+1}$ (A2)

C^2: $2\boldsymbol{P}_{0}^{j} + 4\boldsymbol{P}_{1}^{j} - 8\boldsymbol{P}_{2}^{j} + 2\boldsymbol{P}_{3}^{j} = 2\boldsymbol{P}_{0}^{j+1} - 8\boldsymbol{P}_{1}^{j+1} + 4\boldsymbol{P}_{2}^{j+1} + 2\boldsymbol{P}_{3}^{j+1}$ (A3)

For two segments, $s=2$, the relationship between $\boldsymbol{P}_1^0$ and $\boldsymbol{P}_2^0$ is obtained by substituting (A1) into (A2), $j=0$, to eliminate $\boldsymbol{P}_0^1$, and the result into (A3) to eliminate $\boldsymbol{P}_1^1$:

$$4\boldsymbol{P}_1^0 = -2\boldsymbol{P}_0^0 + 16\boldsymbol{P}_2^0 - 16\boldsymbol{P}_3^0 + 4\boldsymbol{P}_2^1 + 2\boldsymbol{P}_3^1 \tag{A4}$$

If the curve segments $\boldsymbol{B}^j(t)$ represent constrained interpolation data, then $\boldsymbol{P}_0^0$, $\boldsymbol{P}_3^0 = \boldsymbol{P}_0^1$ and $\boldsymbol{P}_3^1$ are fixed as they represent data points, and $\boldsymbol{P}_2^1$ is also fixed as it represents an end derivative. Ignoring fixed terms for clarity, the relationship (A4) becomes:

$$4\boldsymbol{P}_1^0 = 16\boldsymbol{P}_2^0 \tag{A5}$$

i.e. moving $\boldsymbol{P}_1^0$ by Δ causes $\boldsymbol{P}_2^0$ to move by $\Delta/4$.

For $s=3$, the continuity conditions (A1)-(A3), with $j=1$, must be used to eliminate $\boldsymbol{P}_2^1$ in (A4), yielding the control point relationship:

$$4\boldsymbol{P}_1^0 = -2\boldsymbol{P}_0^0 + 16\boldsymbol{P}_2^0 - 16\boldsymbol{P}_3^0 + 4\left[\frac{10\boldsymbol{P}_3^0 - 4\boldsymbol{P}_2^0 + 16\boldsymbol{P}_3^1 - 4\boldsymbol{P}_2^2 - 2\boldsymbol{P}_3^2}{16}\right] + 2\boldsymbol{P}_3^1$$

The control points $\boldsymbol{P}_0^0$, $\boldsymbol{P}_3^0 = \boldsymbol{P}_0^1$, $\boldsymbol{P}_3^1 = \boldsymbol{P}_0^2$ and $\boldsymbol{P}_3^2$ are fixed because they represent data points, and $\boldsymbol{P}_2^2$ represents the fixed derivative. Again, ignoring fixed terms, the relationship becomes:

$$4\boldsymbol{P}_1^0 = 16\boldsymbol{P}_2^0 + 4\left[-\frac{1}{4}\right]\boldsymbol{P}_2^0 \tag{A6}$$

For $s=4$, a relationship between $\boldsymbol{P}_2^2$ and $\boldsymbol{P}_2^1$ can be obtained by substituting (A1) and (A2) into (A3), with $j=2$:

$$-4\boldsymbol{P}_2^1 - 8\boldsymbol{P}_2^2 = 8\boldsymbol{P}_2^2 + 4(0)$$

$$\Rightarrow \qquad \boldsymbol{P}_2^2 = -\frac{1}{4}\boldsymbol{P}_2^1 \tag{A7}$$

$\boldsymbol{P}_0^0$, $\boldsymbol{P}_3^0 = \boldsymbol{P}_0^1$, $\boldsymbol{P}_3^1 = \boldsymbol{P}_0^2$, $\boldsymbol{P}_3^2 = \boldsymbol{P}_0^3$ and $\boldsymbol{P}_3^3$ are fixed because they represent data points, and $\boldsymbol{P}_2^3$ represents the fixed derivative. Rearranging (A3), $j = 1$, substituting in (A1) and (A2), $j = 1$, and ignoring fixed terms, the following can be deduced:

$$-4\boldsymbol{P}_2^0 - 8\boldsymbol{P}_2^1 = 8\boldsymbol{P}_2^1 + 4\boldsymbol{P}_2^2 \tag{A8}$$

Hence a relationship for $\boldsymbol{P}_2^1$ can formulated in terms of $\boldsymbol{P}_2^0$ using (A7) and (A8):

$$\Rightarrow \qquad \boldsymbol{P}_2^1 = -\frac{4}{15}\boldsymbol{P}_2^0 = -\frac{1}{\left(4 - \frac{1}{4}\right)}\boldsymbol{P}_2^0$$

Thus substituting into (A4) and ignoring fixed terms gives:

$$4\boldsymbol{P}_1^0 = 16\boldsymbol{P}_2^0 + 4\left[-\frac{1}{4 - \frac{1}{4}}\right]\boldsymbol{P}_2^0 \tag{A9}$$

(A5), (A6) and (A9) can be expressed in the following form:

$$s = 2 \qquad 4\boldsymbol{P}_1^0 = \left(16 - 4(0)\right)\boldsymbol{P}_2^0$$

$$s = 3 \qquad 4\boldsymbol{P}_1^0 = \left(16 - 4\left(\frac{1}{4 - (0)}\right)\right)\boldsymbol{P}_2^0$$

$$s = 4 \qquad 4\boldsymbol{P}_1^0 = \left(16 - 4\left(\frac{1}{4 - \left(\frac{1}{4 - (0)}\right)}\right)\right)\boldsymbol{P}_2^0$$

The effect of adding additional spans is known because their effect on $\boldsymbol{P}_2^0$ is determined by equations of the same form as (A8). Therefore, in general:

$$4\boldsymbol{P}_1^0 = \left(16 - 4f_s\right)\boldsymbol{P}_2^0, \tag{A10}$$

where $$f_s = \begin{cases} 0 & s = 2 \\ \left(\dfrac{1}{4 - f_{s-1}}\right) & s \geq 3 \end{cases} \tag{A11}$$

Equation (A11) is a recurrence relation, which is difficult to manipulate. The limit as $s \to \infty$ can be found by considering (A11) in terms of a linear difference equation (Appendix B).

Appendix B

Establishing the limit of f_s

Appendix A defined the recurrence relationship:

$$f_s = \begin{cases} 0 & s = 2 \\ \left(\dfrac{1}{4 - f_{s-1}} \right) & s \geq 3 \end{cases}$$

A more convenient formulation is in terms of a linear difference equation. Assume that such a relationship exists in the form:

$$f_s = \frac{a_s - b_s f_2}{c_s - d_s f_2}, \quad k \geq 2$$

where a_s, b_s, c_s, and d_s are integers that yield a fraction in its lowest form.

Since $f_2 = 0$, this becomes:

$$f_s = \frac{a_s}{c_s}, \quad s \geq 2 \qquad \text{(B1)}$$

It is possible to observe how f_s varies with s and establish the coefficients a_s and c_s – see Table B1.

s	f_s	a_s	c_s
2	$f_2 = \frac{0}{1}$	0	1
3	$f_3 = \frac{1}{4-f_2} = \frac{1}{4}$	1	4
4	$f_4 = \frac{1}{4-f_3} = \frac{1}{4-\left(\frac{1}{4}\right)} = \frac{4}{15}$	4	15
5	$f_5 = \frac{1}{4-f_4} = \frac{1}{4-\left(\frac{4}{15}\right)} = \frac{15}{56}$	15	56
6	$f_5 = \frac{1}{4-f_4} = \frac{1}{4-\left(\frac{15}{56}\right)} = \frac{56}{209}$	56	209

Table B1 – Finding the coefficients for the linear difference equation

The following relationships are evident:

$$c_s = 4c_{s-1} - c_{s-2}$$

$$a_s = c_{s-1} \tag{B2}$$

Using (B2), a_s can be eliminated from (B1) to give:

$$f_s = \frac{c_{s-1}}{c_s}, \quad s \geq 2 \tag{B3}$$

The general solution for c_s is:

$$c_s = A\gamma_1^s + B\gamma_2^s$$

and the characteristic equation is:

$$\gamma^2 - 4\gamma + 1 = 0$$

with roots:

$$\gamma = \frac{4 \pm \sqrt{16-4}}{2} = \left(2 \pm \sqrt{3}\right)$$

Thus the specific solution can be formulated:

$$c_s = A\left(2-\sqrt{3}\right)^s + B\left(2+\sqrt{3}\right)^s \tag{B4}$$

When $s = 1$, $c_1 = 0$

$$\Rightarrow \qquad A\left(2-\sqrt{3}\right) + B\left(2+\sqrt{3}\right) = 0 \tag{B5}$$

When $s = 2$, $c_2 = 1$

$$\Rightarrow \qquad A\left(2-\sqrt{3}\right)^2 + B\left(2+\sqrt{3}\right)^2 = 1 \tag{B6}$$

Eliminating *B* using (B5) and (B6) yields:

$$A = \frac{1}{\left(2-\sqrt{3}\right)^2 - 1} \tag{B7}$$

Eliminating *A* using (B5) and (B6) yields:

$$B = \frac{1}{\left(2+\sqrt{3}\right)^2 - 1} \tag{B8}$$

Manipulating (B7) gives:

$$A = \frac{1}{\left(2-\sqrt{3}\right)^2 - 1} = -\frac{\left(2+\sqrt{3}\right)^2}{\left(2+\sqrt{3}\right)^2 - 1} \tag{B9}$$

The value of c_s is then obtained by substituting (B8) and (B9) into (B4):

$$c_s = -\frac{\left(2+\sqrt{3}\right)^2}{\left[\left(2+\sqrt{3}\right)^2 - 1\right]}\left(2-\sqrt{3}\right)^s + \frac{1}{\left[\left(2+\sqrt{3}\right)^2 - 1\right]}\left(2+\sqrt{3}\right)^s$$

$$\Rightarrow \qquad c_s = \frac{1}{\left[\left(2+\sqrt{3}\right)^2 - 1\right]}\left[-\left(2+\sqrt{3}\right)^2\left(2-\sqrt{3}\right)^s + \left(2+\sqrt{3}\right)^s\right]$$

$$\Rightarrow \qquad c_s = \frac{1}{\left[\left(2+\sqrt{3}\right)^2 - 1\right]}\left[\left(2+\sqrt{3}\right)^s - \left(2-\sqrt{3}\right)^{s-2}\right],\ s \geq 3 \qquad \text{(B10)}$$

Finally, f_s is derived by substituting (B10) into (B3):

$$f_s = \frac{c_{s-1}}{c_s} = \frac{\left[\left(2+\sqrt{3}\right)^{s-1} - \left(2-\sqrt{3}\right)^{s-3}\right]}{\left[\left(2+\sqrt{3}\right)^s - \left(2-\sqrt{3}\right)^{s-2}\right]},\ \ s \geq 3 \qquad \text{(B11)}$$

This is verified using proof by induction: checking for the smallest case:

$$s = 3 \qquad f_3 = \frac{\left[\left(2+\sqrt{3}\right)^2 - \left(2-\sqrt{3}\right)^0\right]}{\left[\left(2+\sqrt{3}\right)^3 - \left(2-\sqrt{3}\right)^1\right]} = \frac{6+4\sqrt{3}}{24+16\sqrt{3}} = \frac{1}{4}$$

which is correct.

Assuming (B11) to be true for *s*, test for $s+1$:

$$\frac{\left[\left(2+\sqrt{3}\right)^s - \left(2-\sqrt{3}\right)^{s-2}\right]}{\left[\left(2+\sqrt{3}\right)^{s+1} - \left(2-\sqrt{3}\right)^{s-1}\right]} \equiv f_{s+1} \quad \text{q.e.d.}$$

Hence (B11) is true when $s \geq 3$.

From (B11), the limit of f_s as $s \to \infty$ is:

$$\lim_{s\to\infty} f_s = \lim_{s\to\infty} \frac{\left[\left(2+\sqrt{3}\right)^{s-1} - \left(2-\sqrt{3}\right)^{s-3}\right]}{\left[\left(2+\sqrt{3}\right)^{s} - \left(2-\sqrt{3}\right)^{s-2}\right]}$$

$$\Rightarrow \qquad \lim_{s\to\infty} f_s = \lim_{s\to\infty} \frac{\left(2+\sqrt{3}\right)^{s-1}}{\left(2+\sqrt{3}\right)^{s} - \left(2-\sqrt{3}\right)^{s-2}} - \lim_{k\to\infty} \frac{\left(2-\sqrt{3}\right)^{s-3}}{\left(2+\sqrt{3}\right)^{s} - \left(2-\sqrt{3}\right)^{s-2}}$$

Noting that $\left(2+\sqrt{3}\right)^{-1} \equiv \left(2-\sqrt{3}\right)$, this can be simplified to:

$$\Rightarrow \qquad \lim_{s\to\infty} f_s = \lim_{s\to\infty} \frac{\left(2-\sqrt{3}\right)}{1-\left(2-\sqrt{3}\right)^{2s-2}} - \lim_{k\to\infty} \frac{\left(2+\sqrt{3}\right)}{\left(2+\sqrt{3}\right)^{2s-2} - 1}$$

$$\Rightarrow \qquad \lim_{s\to\infty} f_s = \left(2-\sqrt{3}\right) \qquad \text{(B12)}$$

Appendix C

Numerical results from twist vector tests

The results for each of the 96 cases (24 surfaces, each with 4 corners) from Section 6.4 are given in Tables C1 to C4, and record the absolute positional error between the true and interpolated mid-points of the corner sub-patches. In all cases, the Improved method is either the same or better than Adini. $[*]$ indicates the method.

Case	Error	$\boldsymbol{B}^{0,0}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$	$\boldsymbol{B}^{5,0}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$	$\boldsymbol{B}^{0,5}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$	$\boldsymbol{B}^{5,5}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$
(a)	Adini New	0.000956038 0.000181146	0.000956038 0.000181146	0.000956038 0.000181146	0.000956038 0.000181146
(b)	Adini New	0.001031893 0.000197260	0.001297937 0.000443613	0.000397021 0.000102989	0.000299838 0.000076281
(c)	Adini New	0.001098750 0.000290537	0.001098750 0.000290537	0.000324420 0.000079729	0.000324420 0.000079729
(d)	Adini New	0.000773581 0.000178811	0.000318235 0.000086833	0.000603612 0.000098228	0.000141436 0.000062483
(e)	Adini New	0.000838934 0.000208494	0.000652596 0.000135367	0.000305905 0.000056944	0.000219863 0.000040049
(f)	Adini New	0.001172420 0.000365068	0.000482422 0.000081385	0.000455922 0.000069065	0.000085379 0.000008433

Table C1 – Positional errors from the Adini and Improved surfaces for torus cases

Case	Error	$\boldsymbol{B}^{0,0}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$	$\boldsymbol{B}^{5,0}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$	$\boldsymbol{B}^{0,5}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$	$\boldsymbol{B}^{5,5}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$
(a)	Adini	0.000108664	0.000061187	0.000108664	0.000061187
	New	0.000019354	0.000011741	0.000019354	0.000011741
(b)	Adini	0.000117450	0.000068789	0.000044551	0.000038093
	New	0.000022550	0.000018869	0.000011072	0.000008888
(c)	Adini	0.000118246	0.000065756	0.000043506	0.000031569
	New	0.000023727	0.000015507	0.000010761	0.000007377
(d)	Adini	0.000064059	0.000043192	0.000038921	0.000063800
	New	0.000011982	0.000008382	0.000008196	0.000011993
(e)	Adini	0.000074664	0.000027738	0.000023727	0.000029107
	New	0.000014945	0.000011506	0.000005171	0.000006342
(f)	Adini	0.000083344	0.000031570	0.000029355	0.000030603
	New	0.000017509	0.000009550	0.000005792	0.000006720

Table C2 – Positional errors from the Adini and Improved surfaces for sphere cases

Case	Error	$\boldsymbol{B}^{0,0}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$	$\boldsymbol{B}^{5,0}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$	$\boldsymbol{B}^{0,5}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$	$\boldsymbol{B}^{5,5}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$
(a)	Adini	0.000000001	0.000000001	0.000000000	0.000000000
	New	0.000000001	0.000000001	0.000000000	0.000000000
(b)	Adini	0.000233211	0.000463585	0.000045244	0.000309330
	New	0.000042162	0.000148289	0.000019415	0.000025247
(c)	Adini	0.000283850	0.000283850	0.000135911	0.000135911
	New	0.000077629	0.000077629	0.000015865	0.000015865
(d)	Adini	0.000243377	0.000264913	0.000308712	0.000340592
	New	0.000018236	0.000061690	0.000071321	0.000033255
(e)	Adini	0.000233141	0.000804766	0.000174168	0.000555398
	New	0.000011212	0.000251036	0.000045966	0.000055713
(f)	Adini	0.000201466	0.000438890	0.000054802	0.000349498
	New	0.000022427	0.000121332	0.000018436	0.000028270

Table C3 – Positional errors from the Adini and Improved surfaces for cone cases

Case	Error	$\boldsymbol{B}^{0,0}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$	$\boldsymbol{B}^{5,0}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$	$\boldsymbol{B}^{0,5}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$	$\boldsymbol{B}^{5,5}_{[*]}\left(\tfrac{1}{2},\tfrac{1}{2}\right)\ (m)$
(a)	Adini	0.000000000	0.000000000	0.000000000	0.000000000
	New	0.000000000	0.000000000	0.000000000	0.000000000
(b)	Adini	0.000320571	0.000053097	0.000155522	0.000050326
	New	0.000100376	0.000027460	0.000004098	0.000015569
(c)	Adini	0.000132662	0.000132662	0.000081021	0.000081021
	New	0.000044060	0.000044060	0.000010460	0.000010460
(d)	Adini	0.000092357	0.000092357	0.000092357	0.000092358
	New	0.000010340	0.000020390	0.000020391	0.000010340
(e)	Adini	0.000069826	0.000160757	0.000052376	0.000089735
	New	0.000026754	0.000048559	0.000009973	0.000006873
(f)	Adini	0.000071865	0.000337963	0.000053817	0.000157444
	New	0.000024204	0.000103068	0.000015714	0.000002506

Table C4 – Positional errors from the Adini and Improved surfaces for cylinder cases

Appendix D

Geometric relationships for *N = 3*

Hypothesis for *N = 3*

Given $N = 3$ Bézier patches that join with a common vertex $\boldsymbol{P}^k_{0,0}$, $0 \le k < N$, to form a 3-sided surface, then:

$$\prod_{i=0}^{3} \alpha_i = -\prod_{i=0}^{3} \beta_i = 1$$

Proof

The vertices $\boldsymbol{P}^k_{0,1}$, $0 \le k < N$, are all related geometrically in terms of α_k and β_k. The governing equations are the G^1 constraints given by (7.12) in Chapter 7, assuming $m_k = n_k$, $0 \le k < N$:

$$\boldsymbol{P}^k_{0,1} - \boldsymbol{P}_{0,0} = -\alpha_k \left(\boldsymbol{P}^c_{1,0} - \boldsymbol{P}_{0,0}\right) + \beta_k \left(\boldsymbol{P}^c_{0,1} - \boldsymbol{P}_{0,0}\right) \qquad 0 \le k < N,\ c = k - 1 \ \left(mod\ \ N\right)$$

These relationships are illustrated in Figure D1.

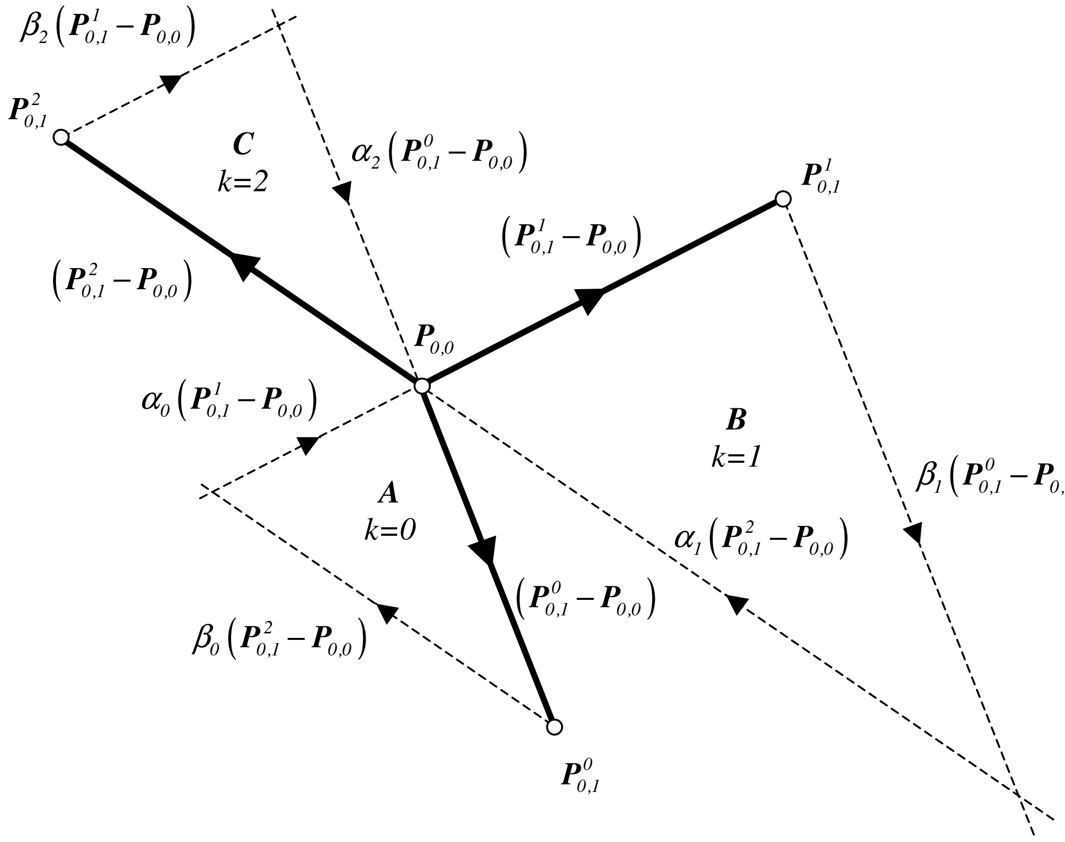

$\beta_2\left(\boldsymbol{P}^{1}_{0,1}-\boldsymbol{P}_{0,0}\right)$
$\boldsymbol{P}^{2}_{0,1}$
$\boldsymbol{C}$
$k=2$
$\alpha_2\left(\boldsymbol{P}^{0}_{0,1}-\boldsymbol{P}_{0,0}\right)$
$\boldsymbol{P}^{1}_{0,1}$
$\left(\boldsymbol{P}^{1}_{0,1}-\boldsymbol{P}_{0,0}\right)$
$\left(\boldsymbol{P}^{2}_{0,1}-\boldsymbol{P}_{0,0}\right)$
$\boldsymbol{P}_{0,0}$
$\alpha_0\left(\boldsymbol{P}^{1}_{0,1}-\boldsymbol{P}_{0,0}\right)$
$\boldsymbol{B}$
$k=1$
$\beta_1\left(\boldsymbol{P}^{0}_{0,1}-\boldsymbol{P}_{0,0}\right)$
$\boldsymbol{A}$
$k=0$
$\alpha_1\left(\boldsymbol{P}^{2}_{0,1}-\boldsymbol{P}_{0,0}\right)$
$\left(\boldsymbol{P}^{0}_{0,1}-\boldsymbol{P}_{0,0}\right)$
$\beta_0\left(\boldsymbol{P}^{2}_{0,1}-\boldsymbol{P}_{0,0}\right)$
$\boldsymbol{P}^{0}_{0,1}$

Figure D1 – Geometric relationships for $\boldsymbol{P}^{k}_{0,1}$, $N=3$

It is assumed in Section 7.3.1 that the angle between $\left(\boldsymbol{P}^{k}_{0,1}-\boldsymbol{P}_{0,0}\right)$ and $\left(\boldsymbol{P}^{c}_{0,1}-\boldsymbol{P}_{0,0}\right)$, $c=k-1\ \left(mod\ N\right)$, $0\le k<N$, is ≤180°, therefore $\alpha_k>0$ and $\beta_k<0$.

Consider triangles ***A*** and ***B***. The ratios of corresponding sides are:

$$\frac{\left|\alpha_0\right|\left\|\boldsymbol{P}^{1}_{0,1}-\boldsymbol{P}_{0,0}\right\|}{\left\|\boldsymbol{P}^{1}_{0,1}-\boldsymbol{P}_{0,0}\right\|}=\frac{\left\|\boldsymbol{P}^{0}_{0,1}-\boldsymbol{P}_{0,0}\right\|}{\left|\beta_1\right|\left\|\boldsymbol{P}^{0}_{0,1}-\boldsymbol{P}_{0,0}\right\|}=\frac{\left|\beta_0\right|\left\|\boldsymbol{P}^{2}_{0,1}-\boldsymbol{P}_{0,0}\right\|}{\left|\alpha_1\right|\left\|\boldsymbol{P}^{2}_{0,1}-\boldsymbol{P}_{0,0}\right\|}$$

i.e. $$\left|\alpha_0\right|=\frac{1}{\left|\beta_1\right|}=\frac{\left|\beta_0\right|}{\left|\alpha_1\right|} \qquad \text{(C1)}$$

Since $\alpha_k>0$ and $\beta_k<0$, $0\le k<N$:

$$\left|\alpha_0\right|\left|\beta_1\right|=1 \qquad \Rightarrow \qquad \alpha_0\beta_1=-1$$

Considering triangles ***A*** and ***C***, and ***B*** and ***C*** in turn gives:

$$\alpha_1\beta_2 = -1 \tag{C2}$$

$$\alpha_2\beta_0 = -1 \tag{C3}$$

Now from (C1), $\frac{|\beta_0|}{|\alpha_1|} = |\alpha_0|$, and from (C3), $\beta_0 = -\frac{1}{\alpha_2}$, therefore:

$$|\alpha_0| = \frac{1}{|\alpha_1||\alpha_2|} \qquad \Rightarrow \qquad \alpha_0\alpha_1\alpha_2 = 1$$

since $\alpha_k > 0$, $0 \le k < N$.

Similarly, from (C1), $\frac{|\beta_0|}{|\alpha_1|} = \frac{|1|}{|\beta_1|}$, and from (C2), $\alpha_1 = -\frac{1}{\beta_2}$, therefore:

$$\frac{|\beta_0||\beta_2|}{1} = \frac{1}{|\beta_1|} \qquad \Rightarrow \qquad \beta_0\beta_1\beta_2 = -1$$

since $\beta_k < 0$, $0 \le k < N$, which completes the proof.

References

Ahlberg, J.H., Nilson, E.N. and Walsh, J.L. [1967], *The theory of splines and their applications*, Academic Press, New York, USA.

Ali, J. [1994], *Geometric Control of Planar Curves*, Ph. D. Thesis, Geometric Modelling Group, The University of Birmingham.

Ali, J., Tookey, R.M., Ball, J.V., Ball, A.A. [1999], The generalised Cornu spiral and its application to span generation, *Journal of Computational and Applied Mathematics*, **102**, pp. 37-47.

Ball, A.A. [1974], CONSURF. Part one: introduction of the conic lofting tile, *Computer-Aided Design*, **6**(4), pp. 243.

Ball, A.A. [2004], Orthogonal C^1 and C^2 piecewise cubic interpolation, Unpublished paper, Geometric Modelling Group, The University of Birmingham.

Ball, A.A. [2005], Derivative magnitude estimation using orthogonality conditions, Unpublished algorithm, Geometric Modelling Group, The University of Birmingham.

Barnhill, R.E., Brown, J. and Klucewicz, I. [1978], A new twist in CAGD, *Computer Graphical Image Processing*, **8**, pp. 78-91.

Barnhill, R.E., Farin, G., Fayard, L. and Hagen, H. [1988], Twists, curvatures and surface interrogation, *Computer-Aided Design*, **20**(6), pp. 341-346.

Barsky, B.A. [1981], *The Beta-spline: A Local Representation Based on Shape Parameters and Fundamental Geometric Measures*, Ph. D. Thesis, Department of Computer Science, University of Utah.

Barsky, B.A. and Beatty, J.C. [1983], Local control of Bias and Tension in Beta-splines, *ACM Transactions on Graphics*, **2**(2), pp. 109-134.

Barsky, B.A. and DeRose, T.D. [1989], Geometric Continuity of Parametric Curves: Three Equivalent Characterizations, *IEEE Computer Graphics and Applications*, **9**(6), pp. 60-68.

de Boor, C. [1972], On calculation with B-Splines, *Journal of Approximation Theory*, **6**, p.50-62.

de Boor, C. [1978], *A practical guide to splines*, Springer, Berlin.

de Boor, C. [2001], *A practical guide to splines*, Revised Edition, Springer, New York, ISBN 0-387-95366-3.

Brodlie, K.W. [1980], A review of methods for curve and function drawing in Mathematical methods, in *Computer Graphics and Design*, Edited Brodlie, K.W., Academic Press, London.

Chong, P.L. [2006], *Visual Assessment of Free-form Surface Quality*, Ph. D. Thesis, Geometric Modelling Group, The University of Birmingham.

Cohen, E. [1987] A new local basis for designing with tensioned splines, *ACM Transactions on Graphics*, **6**, pp.81-122.

Conte, S.D. and de Boor, C. [1981], *Elementary numerical analysis: An algorithmic approach*, Third Edition, McGraw-Hill, ISBN 0-07-066228-2.

Cox, M.G. [1972], The numerical evaluation of B-Splines, *Institute of Mathematics and its Applications*, **10**, pp. 134-149.

Cripps, R.J. [2003], Algorithms to support point-based CADCAM, *International Journal of Machine Tools and Manufacture*, **43**(4), pp. 425-432.

Cripps, R.J. and Ball, A.A. [1998], Visual assessment of free-form Surfaces in CAD/CAM, *Proceedings of the IMechE*, **212, Part B**, pp. 207-214.

Cripps, R.J. and Ball, A.A. [2003], Orthogonal Cubic C^2 B-Splines, *Proceedings of the 10th SIAM International Conference on Geometric Design*, Seattle.

Cripps, R.J. and Lockyer, P.S. [2005], Circle approximation for CADCAM using orthogonal C^2 cubic B-splines, *International Journal of Machine Tools and Manufacture*, **45**, pp. 1222-1229.

Czerkawski, A.M. [1996], *Fitting procedures for curves and surfaces*, Ph. D. Thesis, Geometric Modelling Group, The University of Birmingham.

Farin, G. [1988], *Curves and Surfaces for CAGD, A Practical Guide*, First Edition, Academic Press, London, ISBN 0-12-249050-9.

Farin, G. [2002], *Curves and Surfaces for CAGD, A Practical Guide*, Fifth Edition, Morgan-Kaufmann, ISBN 1-55860-737-4.

Farin, G. and Hagen, H. [1992], A local twist estimator, in *Topics in Surface Modeling*, Edited Hagen, H., SIAM, Philadelphia, PA, pp. 79-84.

Faux, I.D. and Pratt, M.J. [1979], *Computational Geometry for Design and Manufacture*, Ellis Horwood, ISBN 0-470-27069-1.

Ferguson, J. [1964], Multivariable curve interpolation, *Journal of the Association for Computing Machinery*, **11**(2), pp. 221-228.

Goldman, R.N. [2004], Multisided arrays of control points for multisided Bézier Patches, *Computer Aided Geometric Design*, **21**(3), pp. 243-261.

Gordon, W.J. and Riesenfeld, C.J. [1974], Bernstien-Bézier methods for the computer aided design of free-form curves and surfaces, *Journal of the Association for Computing Machinery*, **21**(2), pp. 293-310.

Hagen, H. [1985], Geometric spline curves, *Computer Aided Geometric Design*, **2**(1-3), pp. 223-227.

Hagen, H. and Schulze, G. [1987], Automatic smoothing with geometric surface patches, *Computer Aided Geometric Design*, **4**(3), pp. 231-236.

Hosaka, M. and Kimura, F. [1984], Non-four-sided patch expressions with control points, *Computer Aided Geometric Design*, **1**(1), pp. 75-86.

Hoschek, J. [1992], Circular splines, *Computer-Aided Design*, **24**(11), pp. 611-618.

Kahmann, J. [1983], Continuity of curvature between adjacent Bézier patches, in *Surfaces in CAGD*, ed. Barnhill, R.E. and Boehm, W., North-Holland Publishing Company, Amsterdam, pp 65-75.

Lee, E.T.Y. [1989], Choosing nodes in parametric curve interpolation, *Computer-Aided Design*, **21**(6), pp. 363-370.

Lee, S.L., Tan, H.H. and Majid, A.A. [1995], Smooth piecewise biquartic surfaces from quadrilateral control polyhedra with isolated *n*-side faces, *Computer-Aided Design*, **27**(10), 741-758.

Ma, X. [2006], *Geometry based data reduction for planar curve points and surface points*, Ph. D. Thesis, Geometric Modelling Group, The University of Birmingham, pp. 59-70.

Nielson, G. [1984], A locally controllable spline with tension for interactive curve design, *Computer Aided Geometric Design*, **1**(3), pp. 199-205.

Nowacki, H. and Reese, D. [1983], Design and fairing of ship surfaces, in *Surfaces in CAGD*, ed. Barnhill, R.E. and Boehm, W., North-Holland Publishing Company, Amsterdam, pp. 121-134.

O'Neill, E.F. [1993], *Geometry based constructions for curves and surfaces*, Ph. D. Thesis, Geometric Modelling Group, The University of Birmingham.

Park, H. [2001], Choosing nodes and knots in closed B-spline curve interpolation to point data, *Computer-Aided Design*, **33**(13), pp. 967-974.

Partridge, M.F. [1968], Algorithm for drawing ellipses or hyperbola with a digital plotter (Letter to the Editor), *The Computer Journal*, **11**(2), pp. 119-120.

Piegl, L. and Tiller, W [1997], *The NURBS Book*, Second Edition, Springer-Verlag, ISBN 3-540-61545-8.

Pitteway, M.L.V. [1967], Algorithm for drawing ellipses or hyperbola with a digital plotter, *The Computer Journal*, **10**(3), pp. 282-289.

Rogers, David F. [2001], *An introduction to NURBS: with historical perspective*, Morgan Kaufmann Publishers, ISBN 1-55860-669-6.

Rogers, D.F. and Adams, J.A. [1990], *Mathematical Elements for Computer Graphics*, Second Edition, McGraw-Hill, ISBN 0-07-100289-8.

Sabin, M.A. [1983], Non-rectangular surfaces suitable for inclusion in a B-spline surface, in *Eurographics '83*, ed. Hagen, T., pp. 57-69.

Schoenberg, I.J. [1946], Contributions to the problem of approximation of equidistant data by analytical functions, *Applied Mathematics*, **4**(1), pp. 45-99; 112-141.

Selesnick, S.A. [1981], Local invariants and twist vectors in CAGD, *Computer Graphical Image Processing*, **17**, pp. 145-160.

Smith, F. [1999], Machining Impossible Shapes, *Proceedings of IFIP TC5 WG5.3 International Conference on Sculptured Surface Machining*, ed. Olling, G.J., Choi, B.K. and Jerard, R.B., Kluwer Academic Publishers, Michigan, USA, pp. 74-81.

Smith, L.B. [1971], Drawing ellipses, hyperbolas or parabolas with a fixed number of points and maximum inscribed area, *The Computer Journal*, **14**(1), pp. 81-86.

Versprille, K.J. [1975], *Computer-aided Design Applications of the Rational B-Spline Approximation Form*, Ph. D. Thesis, Syracuse University, Syracuse, NY.

Zheng, J.J. and Ball, A.A. [1997], Control point surfaces over non-four-sided areas, *Computer Aided Geometric Design* **14**(9), pp. 807-821.