Investigating Derivative Estimation as Applied to Smooth Surface Interpolation

by

J. Alexander Clake

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Master of Mathematics

in

Computer Science

Waterloo, Ontario, Canada, 2006

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

blah blah blah

Acknowledgements

I'd like to thank:

Stephen Mann

for supervising me through my fear and indecision.

Wayne and Margaret Clarke, Mom and Dad,

for financial and emotional support.

Shauna and Liam Clarke

for patience and long, longsuffering.

Microsoft and Nortel, OGS, and the University of Waterloo

for scholarship contributions.

The University of Waterloo

for accepting, employing, and putting up with me.

The CGL

for helping me expand my horizons.

The Music Department at UW

for doing the same, and keeping me sane too.

GNU and their GPL concept

for wonderful free, *free* software.

Dr. James Mason and the University of Regina

for providing a work environment away from UW.

Trademarks

Duh! is a trademark of ...

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis is concerned with the problem of smooth surface interpolation. More precisely, I am interested in working on a particular solution to the following problem: given a set of points in the plane with height values and 0 or more derivatives at each point, find a smooth surface that interpolates the heights and all derivatives given at each point. There are several fields that are interested in solutions to this problem including Computer Graphics which is the direction I am approaching the problem from. Different fields look for different things from such interpolants. I am looking primarily for good looking and predictable interpolation for sparse surfaces, the better to model with. Other interests are accurate surface reconstruction and continuity constraints.

Nearly every field of scientific endeavor is interested in representing some data as a surface. Whether the surface should be interpolated or approximated is a question of the reliability of the data. If the data is known to be precise to several significant figures then interpolation is called for. However if the data changes very rapidly and offers different measurements for the same point or very close points a surface estimation is called for. If you want to know the shape and direction of a concrete sidewalk you don't measure the sidewalk on a millimetre scale, interpolate the data and take a derivative. Rather, you might either take samples farther apart or, if you have no choice as to the input data you use a point smoothing scheme to find trends in the data.

A similar situation occurs in computer graphics. If an artist is using an input device to draw curves and sketch then high resolution input data will be approximated using a

A closely related area and one a lot of people ask, including myself, when they start looking at it is why are you doing interpolation instead of fitting? The answer to that straight from my supervisor was that somtimes you know the data is really really good and you don't want to be missing the data if the data's extremely good and your guess is worse than the data. So were looking to interpolate points that are extremely well defined – Say they come from extremely well measured geological surveys. Or say you're designing a car, CAGD, where you just want some thing to go exactly so at certin points. ¡pure bs¿Interpolation is also used in circumstances where data is sparse. That might be for compression reasons or whatever else and you want go right through that data.¡/pure bs¿ Surface fitting is used in cases where you have lots of data and you know the data is noisy so you're looking to smooth it out so you can work out trends in the data and the exact values are not so important even if the data is sparse and you can't really work out the noise you just want to figure out what the data's about. Used a lot in economics, and physical sciences like meterology. Surface interpolation work is also done in weather but they don't seem to desperately concerned with doing it. Summary, why use one over the other? Surface fitting is to simplify a lot of data, mostly, and to expose trends in the data. Interpolation is to upsample sparse data so that you can get good looking results out of work that you've invested a lot of effort in.

In Surface interpolation your model is ultimately going to be finely rendered in some way either its going to sculpted using machines or its going to be finely rendered on scree and used in an animation on the screen, CG. The idea is often that the interpolation should be smooth. So, specifically, I am working with smooth surface interpolation, Shortened SSI. The key is that when you get in close to what you're modelling it needs to keep its smooth aspect on that close examination. When you carve it, it should be smooth to the touch, no wobbles or wiggles on the surface that your finger will identify as being rough or odd. Your hands are good at telling you what's smooth. You can't have discontinuities at the edges, or the boundaries between triangles in the data that will make funny looking creases.

Chapter 2

Background

2.1 Polynomials

Polynomials are the simplest form of function. Every student of algebra is first taught how to expand binomials, for example $(x+1)(x+1) = x^2+2x+1$. The expanded form is the polynomial basis called the Monomial, or power, basis. It is an orthogonal basis for polynomial functions. A polynomial basis consists of terms which are weighted with coefficients and summed to form the polynomial. The terms of an orthogonal basis cannot be expressed by any combination of any other terms. Any polynomial of degree n may be expressed by a weighted sum of the terms for the basis for that degree. Each polynomial is unique, is uniquely represented in each basis, and can be represented by all polynomial bases for that degree of polynomial. Some polynomial bases have specific constructions for the terms of a polynomial that, while similar, are distinct for each degree, while others have the terms of lower degree polynomials as subsets of the higher degree ones. Other orthogonal bases for the polynomials include Bernstein-Bézier, Taylor, Legendre, Chebyshev, and Lagrange. Pertinent to this thesis are the Monomial, Bernstein-Bézier, and Taylor bases.

2.1.1 Monomial Basis

The monomial basis is the simplest basis and so is the preferred basis for doing algebra and calculus. For a univariate polynomial of degree n the terms are:

$$\bigcup_{i=0}^{n} x^i.$$

Their d^{th} derivative can be computed as:

$$\frac{d}{dx^d}\ x^i = \begin{cases} \left(\prod_{j=i}^{i-d} i - j\right) x^{i-d} & \text{if } x \le i \\ 0 & \text{otherwise} \end{cases}$$

Similarly the bivariate terms are:

$$\bigcup_{i=0}^{n} \bigcup_{j=0}^{n-i} x^i y^j.$$

and their partial derivatives are:

$$\frac{\partial}{\partial x^k y^l}\ x^i y^j = \begin{cases} \frac{i!}{k!} \frac{j!}{l!} x^{i-k} y^{j-l} & \text{if } k < i, l < j \\ 0 & \text{otherwise} \end{cases}$$

The monomial basis is generally regarded is not optimally stable; other more stable bases such as the Chebyshev basis are preferred for numerical algorithms. The Numerical Algorithms Group, for instance, uses the latter for its fitters[**?**]. It is easy to calculate the derivative of a polynomial in the monomial basis, making it ideal for estimating derivatives. It also possesses the useful property that the terms of a degree n + 1 polynomial are a superset of the terms of a degree n polynomial. For an fitter this means it is easy to show that a high order fit accurately reproduces a low order polynomial.

2.1.2 Bernstein-Bézier Basis

The Bernstein-Bézier basis is the basis at the root of Bézier approzimation. There is a dynamic progaming approach available to evaluate the function known as the De Castlejau's algorithm. To evaluate f(t), the coefficients associated with "neighboring" basis function are blended together repeatedly, weighted by the relationship between t and the extremes of the domain, until a single value is arrived at. For a more detailed discussion of this technique refer to *Pyramid Algorithms*[1] pp. 194 – 198 for the univariate case, and pp. 279 – 287 for the bivariate triangular case.

The following information on explicit evaluation of the bivariate triangular domain version of this basis comes from the aforementioned text. The degree n triangular Bézier patch $T(s,t)$ with control points P_{ijk}, $i+j+k=n$ can be written as

$$T(s,t) = B^n_{ijk}(s,t)P_{ijk}$$

where

$$B^n_{ijk}(s,t) = \binom{n}{ijk} s^i t^j (1-s-t)^k$$

and

$$\binom{n}{ijk} = \frac{n!}{i!j!k!}.$$

The dynamic algorithm can be modified slightly to calculate derivatives. For univariate Bernstein-B'ezier polynomials refer to pp. 243 – 247 of *Pyramid Algorithms*[1]. The extension of this idea to bivariate triangular Bézier surfaces is on p. 283 of the same book.

2.1.3 Taylor Approximation

Given a point at a on a continuous function f and d derivatives at that point it is possible to construct a degree d interpolating polynomial.

$$p(x) = \sum_{n=0}^{d} \frac{f^n(a)}{n!}(x-a)^n \tag{2.1}$$

This approximation polynomial is called the Taylor polynomial. It approximates the curve with an error at distance $h = x - a$ bounded by $O(h^{d+1})$.

The generalization of the Taylor polynomial to $\Re^n$ dimensions is available online from Mathworld[2]. Using notation similar to (2.1) I give a bivariate version of the Taylor polynomial here for d derivatives at $f(a,b)$,

$$p(x,y) = \sum_{n=0}^{d} \left\{ \frac{\frac{\partial^n}{\partial x} f(a,b)}{n!}(x-a) + \frac{\frac{\partial^n}{\partial y} f(a,b)}{n!}(y-b) \right\}^n .$$

Multivariate versions the Taylor polynomial are bounded by the same order of error as the univariate given the same number of full sets of derivatives.

A good polynomial approximation to a function at a point p should produce derivatives at p that closely match the function's derivatives. We should also expect its error to be bounded in a way consistent with the Taylor error. I use the first assumption in my validation and the second in one of my weighting schemes.

2.2 Least Squares Fitting

2.2.1 Design Matrices

2.2.2 Weighting

2.2.3 Singular Value Decomposition

2.3 Old Junk, being cleaned or relocated

Locally smooth patch based surface interpolation evolved out of CAGD. The tools of the trade were developped by the automotive industry in the late 50s and early 60s. It was at about the time when people were looking to use the new computers to simplify the process of engineering cars.

For most modelling cases B-splines are ideal. B-splines are a surface patch representation that approximates a set of control points. B-splines that share control points automatically join together smoothly. These patches are related to and easily translated to Bézier patches, another kind of spline that interpolates its corners and is easy to work with because the curve/spline always lies inside the convex hull formed by its control points.

B-splines are defined such that you can hook lots of spline patches together in a way that is continuous. Continuous means - in some way mathematically smooth. C^0 continuous means that the edges of the patches you are working with line up, no hole or gaps. C^1 meas that the first derivatives of the patches as you approach the edges at the corners are equal and C^2 continuous is the same for second derivatives and so on. C^2 continuity is about the most that you likely want because as you increase the constraints on how patches meet up you increase the degree of the polynomial you are using and the higher the degree of polynomial you are using the more likely you are to find wiggles which defeats the attempt to have things abstractly smooth.

B-splines are used for smooth surface interpolation because they have this continuity enforcing property and they can be made to fit certain kinds of data easily. You can do a simple least squares fit that forces the patches to interpolate your data. The constraint on your input data is that it should be rectilinear or at least map to a grid. Once you lose that, you have to start fudging things be cause B-splines are modelled on a grid. There are triangular B-patches, but they don't hook toghther nicely. Because B-splines map to a grid if you want to fit them to non rectangular shapes you end up having to collapse a side to make it into a triangle. With triangles you can model pretty much anything. The trouble is you end up with a singularity at one corner and you get bad, unsmooth behaviour out of the singularity. See figure 2.1.

Given any set of points sampled from a surface you can generate a triangulation of those points, so being able to smoothly interpolate a triangulated data set is sort of a holy grail. Although you can make degenerate B-spline or Bézier patches that are triangular, the shape of these patches is poor, and they are difficult to join together with anything more than c^0 continuity. Although some work has been done on triangular B-splines [ref DMS splines, Ingram], no reasonable method exists for constructing triangular patches that automatically join with C^1 or higher continuity in the general case.

Moving away from local patch based techniques, there are global solvers that can do the job. But they are very inefficient and they involve fitting high degree polynomial to the data and attempting to minimize the energy and therefore reduce wiggliness.I've been told that they can look good, but the problem is they are very inefficient. You have to recompute the whole thing,

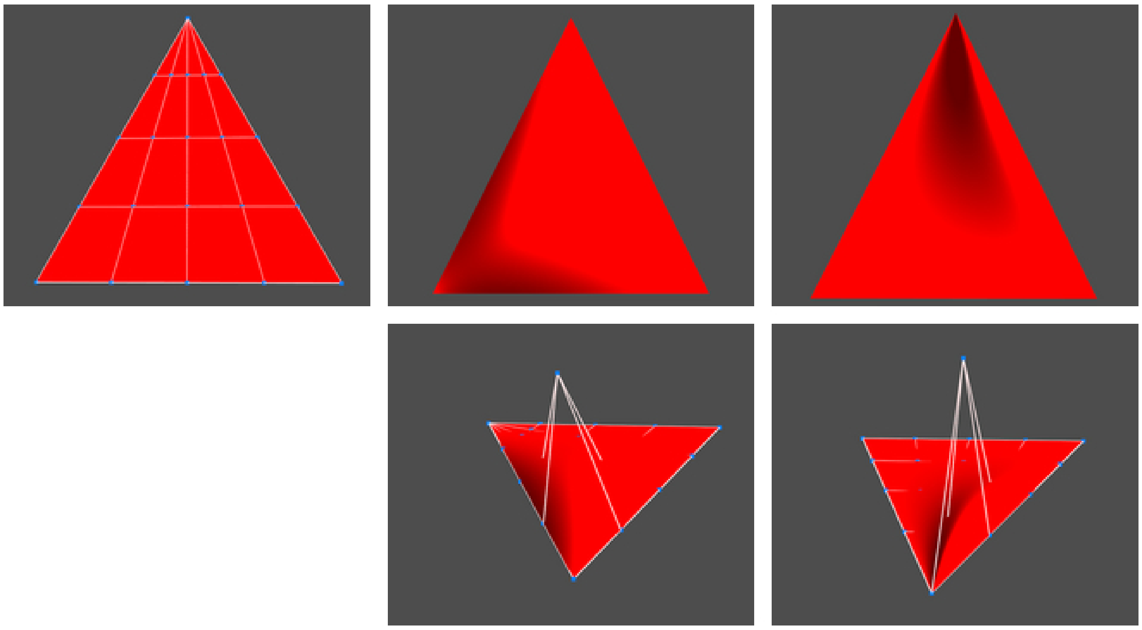

Figure 2.1: What happens when a square patch is used to fit a triangular area? Here we have a square bezier patch with a degenerate side that makes it a triangle. Top left to right: triangle with control points, triangle with non degenerate corner altered, triangle with degenerate corner altered. Bottom row: Hull + perspective view of above patches. The patches have been positioned so that the modified corner is pointing toward the viewer. Both modified patches have had a point near one corner moved 2 units off the plane. Notice how a crease appears near the degenerate corner.

at great expense, anytime you change any part of the surface. The calculation for the surfce itself is complex. The beauty of say cubic B-splines is that the surface is divided up into patches that rely on no more than $4^2 = 16$ points at a time.... conversely one point influences only 16 patches and so recomputation is done in constant time, independant of the size of the dataset. You can do efficient local recomputations for changes to the geometry.

Other techniques I have heard of: Thin plate splines, Iterative B-spline surface refinement, Gaussian weighted blending stuff. (Look it up, don't make it up.)

The local techniques are highly desired for interactive fitting, and also anykind of fitting, really. Low order (cubic) local techniques are probably best. Low order polynomials behave better. But because they are low order, if you have a complex surface that isn't sampled densely enough you will oversimplify it. Cubic splines are generally agreed to fit the bill. If yo are given just points you don't have much data to work with to fit cubic patches. They require much more data that just positions... need 5 partial derivatives $\partial x, \partial y, \partial xx, \partial xy, \partial yy$. You can linearly interpolate the points but you end up with a faceted surface - not smooth at all.

More background - approaches to the local problem and high continuity Farin: Clough-Tocher Asks for normal to the surface must be provided. Triangles in surface are subdivided into 3 triangular patches. Sets up the interior control points for these patches using the normals provided at the vertices of the dataset then fakes the missing interior information. I don't know if it borrows information from surrounding triangles or if it sets them in some way he found to be good enough through theory and experimentation.

His good enough isn't good enough though. You end up with very flat looking patches, very lifeless in the middle. It tends to kill what could be some nice looking functions.

Continuity issues! Bad Curvature!

Dr. Stephen Mann proposed a novel non-split domain scheme with continuity adjustments. Rather than trying to fit 3 cubics to one patch he took a high degree surface (D5 and D9 worked well) then he used Farin's approach to setting interior control points using $\{d =$order of patch $|\lfloor 2d/3 \rfloor\}$ full sets of partial derivatives. By setting the interior parts based on the partial derivatives he ended up with C0 continuity between patches, but C1 continuity and C2 continuity were not achieved. To set center control points along triangle edges, he used point blending. This point blending breaks C1 and C2 continuity. C1 is only absent in one or two patches at the centre of the side where blending occured. To adjust continuity he used a geometric technique to blend control points across patch boundaries

For $\geq$D5 patches he could do a full C1 adjustment. For $\geq$D8 patches could do C2 adjustments.

Mann's results are much better than Farin/Clough-tocher. However a lot of derivative information is needed that probably can't be provided with the data, either because the surface is modelled by hand and the derivatives are hard to adjust or understand or because they are difficult to measure. That was my job - see how well I could automatically set/estimate the derivatives for a point mesh or a point normal mesh. Getting results similar to or better that Clough Tocher results was the desired standard. Getting it from point data only was the ideal because it would be simplest to model.

Why is the number of complete derivatives required by Mann's scheme a problem?

People don't understand derivatives beyond the first derivative when they look at them and don't know how to set them intuitively. The Twist Vector (∂xy) was especially problematic in early CAGD. There was a car that was designed using the twist vectors set to 0 because they were too difficult to get to behave. The car was ugly and did not get far (find nameof car for anecdote, procure a picture if possible).

Also there is the problem of overlarge datasets. Clough-Tocher - 5 fp values. Mann D5 - 12 fp values, Mann D9 - 30 fp values. 2-6x bigger data sets.

And lastly it is unreasonable to expect people to get so much derivative data when making local measurements of real world phenomena. Mostly the real world is not shaped in such a way that one can do that. Altitude of a coordinate, depth of a sample, or temperature is one thing. Measuring how it changes in local directions is a fair bit more, but not unreasonable. But getting even as little as $\lfloor \frac{2*5}{3} \rfloor = 3$

Chapter 3

Derivative Estimation

My research objective was to find a way to smoothly interpolate a mesh using only positional information without a split domain scheme, such as that used in the Clough/Tocher scheme. I had at my disposal the first interpolation scheme proposed and implemented by Stephen Mann in 2000 and described above [3, 4]. This scheme requires $\lfloor 2n/3 \rfloor$ derivatives. Therefor my goal was to estimate derivative information at each point up to the required number of derivatives.

3.1 Methods

One thesis I read covered several derivation techniques[5], but most of them assume either that the function is known but difficult to differentiate, or that more points may be acquired from a black box. My research assumes that data acquisition is complete.

3.2 Simple Estimates

There are a few ways of arriving at estimates of the derivative of a function from discreet samples. A look at any calculus text will demonstrate differencing as a way of estimating the tangent to a function. In computer graphics the average of the directions polygons face surrounding a point may be averaged to produce a normal to the implied surface. This normal can be used to get directional derivatives. Simple polynomials are often fitted to data.

3.2.1 Finite Differencing

Formulae for first derivatives are easily found in standard numerical analysis texts[6], and other references[find some to put here, dummy]. the basic formulae are forward differencing, backward differencing, central difference.

Forward Differencing

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{3.1}$$

Backward Differencing

$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} = \frac{y_i - y_{i-1}}{x_{i+1} - x_{i-1}} \tag{3.2}$$

Central Differencing

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{x_{i+1} - x_{i-1}} = \frac{y_{i+1} - y_{i-1}}{x_{i+1} - x_{i-1}} \tag{3.3}$$

Second derivatives are as follows More equations

3.2.2 Normal and Tangent Approximation

Given a well triangulated mesh it is possible to approximate the normal at each vertex by averaging the cross product of all neighboring edges that are connected to the vertex. This technique is used to perform smooth shading of arbitrary meshes in ray tracers, for example. Once one has the normal to a surface at a point the tangents in any direction are easy to calculate by taking the cross product of the normal and one of the axes,x or y.

3.3 Higher Order Estimates

There are methods for estimating higher order derivatives that follow from the lower order methods.

Repeated Differentiation

Calculating higher order derivatives can be done by iterating any of the simple methods on the derivatives arrived at by the previous steps.

It is possible to arrive at "reasonable" estimates of the derivatives by repeatedly performing central differencing on the centre of a rectilinear mesh and doing either forward or backward differencing on the edges. As a proof of concept I designed a matlab test and extracted results for some surfaces. (If I have time I could generate a few samples of this.) Edge conditions are a problem. Only forward and backward differencing can be used here. They have a higher error that central differencing and this error propagates in from the edges as higher derivatives are taken because the results from one step are fed into the next.

Ames[7] suggested that doing this was inherently unstable and ought to be avoided.

3.3.1 Atoms

The numerical analysis text I referred to for information on finite differencing suggested using a schematic technique for calculating approximations of partial derivatives called atoms. The approximations to partial derivatives are calculated using a diagram that represents the relative placements of values in a discrete mesh. Neither that text nor its reference[7] was clear on how atoms were constructed. My numerical analysis text did provide some examples.

3.3.2 High Order Interpolation

The idea here is to interpolate the neighbourhood of a point with a polynomial, then take directional partial derivatives. The easiest way to interpolate points with a polynomial is to do a Lagrange interpolation. The construction is simple and is presented in detail in chapter 2 of

Figure 3.1: *left*: Degree x polynomial interpolation of points. *right*: Equal degree Bézier approximation of the same points. Notice that the polynomial interpolation bears no resemblance to the shape suggested by the points where the Bézier curve does. The polynomial interpolation, however, interpolates very nicely near the middle of its curve.

Pyramid Algorithms [1]. However there is no way indicated in that text, or in any other I found, to easily find the directional derivatives of the resulting surface. This is not a problem as they could be derived with some effort. There was a bigger problem, one that effort would not easily solve. Lagrange polynomial interpolanting surfaces can only be constructed for points that map to a rectangular or triangular array of points. There is no guarantee that the nearest neighbours to a point in a scattered data set will map to either one of those.

3.3.3 High Order Approximation

One problem with polynomial interpolants is that as the order of the interpolant increases, the interpolation becomes increasingly wiggly. The variation diminishing property of Bézier curves would be a boon - although there is no known equivalent to the property for Bézier patches. Consider figure 3.1 to see why I might have thought this. B-splines could also do the job.

I tried implementing this but the implementation did not go well. Portions of the code may still be present in my code base, but I am uncertain as to how close to complete it is or how long it would take to fix it up.

3.3.4 Least Squares Fitting

The method of choice for approximating functions is least squares fitting. Its use in United States geological surveys in the early days of computing was noted in my introductory linear algebra text[8] Found a paper by Hiroshi Akima, an early researcher in the field, that used it in a context very similar to my own[9].

Least squares fitting allows a close enough approximation to the interpolating polynomial to be made. This is what attracted me to the method at first. As I examined it further I saw that the data points being used could be in any configuration around the point of interest so long as they did not degenerate into a line.

3.4 Implementation of the Least Squares Based Estimator

The derivative estimation method I eventually settled on was least squares. There were three stages that I would need to do the fits. First I needed the nearest neighbours to the point of interest. Next I needed to find polynomial of the degree I wanted that best represented those points. Lastly I needed to differentiate the polynomial and calculate the derivatives at that point.

3.4.1 Nearest Neighbours

The design matrix for this application of LSF requires the solution of the k nearest neighbours problem.

Brute Force Method

The easiest way to implement k nearest neighbours is to scan all the data and keep track of the ten closest points found. This is a $O(n)$ algorithm and for large datasets this becomes the largest cost of the algorithm, though for small data sets (< 1000) it should be fine.

K-D Tree

The generally recognized efficient way to find the nearest neighbours to a point is to construct a K-D tree. This solution has a worst case of $O(n)$ but its average case is $O(logn)$ which is much faster.

My Hack

What I did was take advantage of the fact that I knew I was getting ordered arrays of points and perform a pair of binary searches for the point I wanted then took a box of points from around it using array indices. This is a $O(log\sqrt{n})$ method and is adequate until I need something more robust to support real scattered meshes.

3.4.2 Least Squares Fitting

Least squares fitting looked like my best option so I implemented it using LAPACK and the monomial or power polynomial basis:

$$P(x,y) = \sum_{i=0}^{d} \sum_{j=0}^{d-i} c_{ij} x^i y^j$$

Once I had worked a few things out, the estimator produced pleasing shapes when fit to polynomial data, and looked like it was doing well even on some of the Franke functions. In some places it even seemed to do better that the Clough-Tocher scheme, but patches near sharp features echo that feature. This ringing continues until the points used to estimate the surface no longer include that feature. This meant that the fits I was doing needed to focus on the immediate neighbourhood as much as possible.

The most obvious way to keep the focus narrow is to use no more points than necessary. For a polynomial surface that would be just enough points to uniquely determine each of the coefficients for the basis function being used to fit the data. For an orthonormal basis of a degree d polynomial there are $\frac{d(d+1)}{2} = \frac{d^2+d}{2}$ basis functions, so that many points are needed at a minimum. My first implementation used a box around the point of interest to construct the fitting matrix. The box was originally d points on a side, which gave $\{d > 3 | d^2 > \frac{d^2+d}{2}\}$ points. For fits of degree 3 and lower I used a box with $d+1$ points on a side. My first fits only worked for $d < 4$ surfaces and it wasn't until I saw how ill conditioned the fitting matrices were that I realized I need to seriously overconstrain to get good results.

Th problem is in the sampling pattern. If your data is sampled too regularly - say evenly spaced on a line for univariate data, or on a rectilinear grid for bivariate data, you end up with an ill-conditioned matrix. Overconstraining can bring conditioning numbers down from 10^{17} to 10^4, but that involves considering points that lay far from the point of interest. Using randomly scattered points improves conditioning - I made a small application in Octave that generated a specified number of random points and found that I could easily not overconstrain at all and get conditioning numbers in the order of 10^3 which may be adequate. Random data, however, highlights one of the bugaboos of spline based smooth interpolation schemes. Long thin triangles produce artifacts. For a real world implementation it might be best to try constructing a nearly square fitting matrix first, checking the condition number and overconstraining only to the point where the matrix falls below a certain value.

Akima's method to solve the conditioning problem was to use a lower degree fit with the exact number of points needed to define the coefficients if the fitting matrix was bad. If that failed the algorithm used the next lower degree polynomial. If a planar fit still produced an ill-conditioned design matrix, a surface perpendicular to the triangle formed by the point of interest and its two nearest neighbours was used as the interpolating polynomial. Akima had adhered too closely to the datasets suggested by Franke[10] and so his method failed on regularly spaced data. The rest of what I did was in response to these problems. Later I found a follow up on the paper that corrected the problems[11] using solutions I had already explored.

Weighting To Improve Local Fit

If the reason for the ringing behaviour is that the surface behaved somewhere in a way that, though possibly continuous, could not be easily represented with the basis I was using, to reduce how far the ringing propagated I could try either to reduce the the importance of outliers in the design matrix.

I weighted so as to emphasize the environment around the point I was examining above all other points, so that there was at least an interpolation of the point in question and, hopefully, all of its immediate neighbours. In my initial implementation of least squares the surface that is fit to the data approaches but does not interpolate the points because the least squares matrix is overconditioned. When I weighted the point being examined heavily and gave light weights to all others it seemed to me that my results improved considerably. I tried to optimize the weighting, but found that it was very tricky to do. By tweaking a bivariate Gaussian I was able to make things look better for certain values in certain situations. An earlier fake Gaussian gave me pretty good results, I think but I've since lost it.

Evaluations of weighting are found throughout the results section. Based on the results it seems weighting does not help much.

Stability and Conditioning with Alternate Polynomial Bases

I have already shown that the conditioning of a monomial least squares matrix for a regularly spaced rectangular grid is hopeless without some degree of overconstraining. (There's a paper I referred to for proof that the conditioning in this case is poor. Find it Alex!!! I can't!) Excess overconstraining didn't help past a certain point.

To resolve the problem I considered polynomial bases that promised to have better conditioning than the power basis.

Chebyshev Polynomials At first the Chebyshev basis(same paper as above, also a book... can't remember it) looked very promising, but it turned out to have more questions than answers: what is a bivariate Chebyshev basis, how does one differentiate the Chebyshev basis in a stable manner, etc. I shelved the research into the topic quickly.

Bernstein Basis I then turned to the Bernstein basis which is usually used to construct Bézier splines. I found a paper that agreed recommended it for its stability [12] and I implemented it. There was some improvement, but only about three fold. I needed something that provided several orders of magnitude improvement. But the implementation was simple and, for psychological reasons, I favoured the Bernstein basis in all subsequent tests. My results suggest that there are other factors that make the Bernstein basis, as I have implemented it, a worse choice than the power basis.

I implemented the triangular Bernstein basis for the fitter using the explicit basis functions:

$$B^d_{ijk}(s,t) = \binom{d}{ijk} s^i t^j (1-s-t)^k \qquad \text{where} \qquad i+j+k=d$$

The Bernstein basis is stablest in the range [0..1], and is considerably stabler than the power basis. To take advantage of this I made a simple mapping of the points being considered to fit in the triangle $x + y - 1 = 0$. I found it was easiest to put a minimum bounding square around the points and make it fit into the triangle, making the points lie in the range $([0 - 0.5], [0 - 0.5])$.

The difference was not big enough to make a visual difference at all, and it didn't improve conditioning enough for me to constrain the matrix any less.

Implementation notes: B-B basis calculations are not very stable. Taking partial derivatives failed for D7-D9 fits when I changed my compiler to use strictly 64-bit registers instead of 80-bit temporaries. Monomial basis did not show this problem.

Tangent/Normal Meshes

The estimation problem is the same whether it is applied to a surface or the surface described by its derivative, so it was a simple thing to modify my existing code to take advantage of meshes that provided either tangents or normals along with the points. A quick look at the results in tables 4.9 to 4.16 will show that there is an order of magnitude improvement for the fits that use tangents.

3.4.3 Derivation

Monomial Derivation

I use a simple derivation scheme. For a term T given $e_x =$ the exponent on the x variable, $e_y =$ the exponent on the y variable, $\delta_x =$ the derivations on x, $\delta_y =$ the derivations on y and C_i is the coefficient for the term its contribution to the $\delta_x \delta_y$ derivative is

$$c_i \left(\prod_{j=e_x}^{e_x - \delta_x} j \right) \left(\prod_{k=e_y}^{e_y - \delta_y} k \right) x^{(e_x - \delta_x)} y^{(e_y - \delta_y)}$$

which becomes the code in figure 3.2.

Because I am using integers to hold the factorials I could run into trouble. However, as I am only using the first six derivatives in my implementation the highest value that will result from

```c
/*
  getDerdxdy
  Returns the nxnyth derivative monomial x^Dx*y^Dy

  Dx = exponent of x (Degree of x)
  Dy = exponent of y (Degree of Y)
  nx = number of x derivations
  ny = number of y derivations
  */
double getDerdxdy(double x, double y, int xn, int yn, int Dx, int Dy)
{
        double  der=0,i;
        int powy, powx, facx=1, facy=1;

        for (i=0; i < xn; i++)
        {
                facx *= Dx-i;
        }

        for (i=0; i < yn; i++)
        {
                facy *= Dy-i;
        }

        powx = max(Dx-xn,0);
        powy = max(Dy-yn,0);

        der = (double)(facx*facy)*pow(x,powx)*pow(y,powy);

        if (isnan(der))
        {
                fprintf(stderr,
                        "NAN: getDernxny x%lf,y%lf,xn%d,yn%d,Dx%d,Dy%d"
                        , x, y, xn,yn,Dx,Dy);
                exit(1);
        }
        return der;
}
```

Figure 3.2: C realization of monomial term derivation equation. Its inefficiencies include the fact that it does not perform simple checks to see if the return value is zero.

the two factorials is $(9*8*7*6*5*4)^2 = 3,657,830,400 < 4,294,967,296 = 2^{32}$ which fits nicely into an integer. After this I convert to double with its mantissa of 52 bits. Things should be okay for many, but not all, cases. I thought there might be trouble with overflowing the double so I tried to catch NANs, which helped me debug bad matrices, but the real problems will come from loss of precision.

I could have invested more time on researching better Monomial derivation schemes and possibly have use a symbolic math package, but this one seems adequate. To test it I will use some really outrageous coefficients on a degree nine surface.

Bernstein Derivation

I implemented the pyramid triangular Bernstein patch evaluator as promoted by Goldman[1]. The evaluator is easily modified so that it provides derivatives. It provides partial derivatives by setting $1-s-t \to -1, s \to 1, t \to 0$ on a number of levels equal to the derivatives in the s direction and for the t derivatives we set $1-s-t \to -1, s \to 0, t \to 1$. This works, though the discusson provided by Goldman [1] only proves it for directional derivatives along s or t not partials on both. We get derivatives on x and y by setting up the control mesh for the Bernstein polynomial such that s and t are parameterized along those axes.

Because the numbers involved in computing the derivatives of the Bernstein basis are moderate there should not be any numerical difficulties associated with them, except that for degree 9 Bernstein's there are difficulties as shown in tables 4.14 and 4.13

3.5 uncategorized junk

I found a post on derivative estimation on a stats package (Splus is the name of the package) news list, and located the author's faculty page[13]. It pointed to the use of spline smoothing to estimate derivatives and suggested using penalties on the magnitude of some derivatives of the fitted curves to achieve better estimates on some lower order derivative. The technique intrigued me and prompted me to investigate splines approximation as a method, which then led me to look at interpolation as a form of finite differencing. In the end it became clear that the technique was only intended to fit a curve to noisy data, not the precise data that is assumed in this technique. Further it seemed fairly clear that the type of estimation done by the localized least squares variant I came up with did a better job of making smooth surfaced when applied to 'Mann -m 1 ¡ *'

Chapter 4

Results

To evaluate my methods I will present some results that validate certain steps using simple data sets and others that evaluate how the method performs on more complex data sets.

All the numerical tests I perform will be evaluating the error between a computable expected result and the estimator's result. For all tests I report at least the maximum absolute error. It has been suggested to me that the maximum error is the only number I need to report. The minimum error is of little value and I include it only so that the reader can judge whether the mean lies in a reasonable place. I report the mean error here because, especially on non polynomial datasets, the fitter often has a problem with some small portion of the data but performs well most of the time. I also include the mean because it has been a feature of these evaluations since Franke proposed his interpolator evaluation scheme in 1979[10].

In general I would use relative error $\frac{x-x^*}{x^*}$ where x is the estimated value and x^* is the true value. In the case where x^* is very small, however, this breaks down. Much of what I will be analyzing will be very small. ([6] p. 14) I would like to use the relative error because it can be used to estimate the number of significant figures an estimate has:

> The number x is said to approximate x^* to t significant digits if t is the largest

nonnegative integer for which

$$\left|\frac{x - x^*}{x^*} < 5 \cdot 10^{-t}\right|$$

Using this definition, we see that 0.998 approximates the true solution, $x^* = 0.9986$, to three significant digits... On the other hand, 0.999 approximates the true solution, $x^* = 0.9986$, to four significant digits...([6] p. 16)

So by expressing the errors in scientific notation I have a quick way to judge the quality of an estimate.

4.1 Validations

To validate a method is to test certain intermediate results and final results on data sets where the results are knowable and the method should perform well. In the case of my polynomial approximation techniques I check two things. First I test coefficient reproduction, then derivative accuracy.

I could validate monomials of any degree interpolating any monomial of lesser or equal degree. For brevity I demonstrate this only for 3^{rd},4^{th}, 5^{th}, and 9^{th} degree fitters as these are used for fit evaluation later. I show that they reproduce a linear, quadratic, cubic, quartic, quintic, sextic, septic, octic, and nonic surface for fits done with an equal or higher degree polynomial than the surface. For reasons discussed below, I only demonstrate coefficient reproduction for the Bernstein basis for same degree fits. For derivative accuracy tests I include the estimated derivatives of surfaces of higher degree than the fitting polynomial for convenience in comaprison, though they represent evaluations and not validations.

4.1.1 Coefficient Reproduction

I am working with the assumption that when I fit a polynomial to a polynomial of the same or lesser order I will get the same polynomial. This should be true even if the fit is overconstrained. I expect this result because of the uniqueness of the polynomial interpolant. For every set of points equal to the number of coefficients in a polynomial there is a unique interpolating polynomial, regardless of the basis used. If the points were generated using a polynomial function of equal or lesser degree then the unique interpolating polynomial should be the same as the generating polynomial. If the interpolation is done through more points than needed to define the generating polynomial the interpolating polynomial should still represent the generating polynomial. All this is true for exact mathematics. If there is a loss of precision, as in digital representation, then there will be some error. This error will get worse as more points are interpolated. This type of error should not be as critical with a fitter that is approximating the interpolant such as a least squares fitter. If all the points represent the same function but suffer from discrete representation errors, then the low degree function approximating them will likely reflect the generating function better than one that truly interpolates the slightly wrong points with too high a degree.

When the generating function is not a polynomial, or is of higher degree than the fitting function then the last paragraph means squat. The approximating polynomial could be completely wrong and will change depending on how many points are being used in the approximation. Generally the fewest points as possible is best.

To test coefficient reproduction I use functions that are generated using the same polynomial basis that I am fitting with. I then compare the coefficients I get against the ones used to generate the data and report their maximum, minimum, and mean errors, but the only artifacts left over from this are the Bernstein Coefficient Reproduction tables below. This is to ensure that my fitting technique is capable of reproducing the function that produced a data set – it has same basis precision.

In the case of my power basis fitter I can easily test all orders of polynomials equal to or lower than my fitting basis. This is because a polynomial in power basis contains all the exact terms used in lower order polynomials. Producing a lower order polynomial is a simple matter of setting coefficients for the high order basis elements to 0.

The Bernstein basis functions do not have this property so it is difficult to be sure that the fit is reproducing a lower order polynomial accurately. One way to do the test would be to do a degree elevation step. Another way would be to perform a conversion or change of basis on a lower order Monomial function. These both add needless complication to my validations. Instead for lower order polynomial reproduction by the Bernstein fits I will rely on the derivative accuracy tests. I should be able to do this without fear because Bernstein polynomials are able to represent lower order polynomials. Conversions between monomial and Bernstein bases and their proofs exist [1].

The large number of tables that would be generated were I to give a full report on all the surfaces tested has caused me to report in one large table the one maximum coefficient error for all estimates made on the surface for each mesh fit. See Table 4.3. Table 4.1 is a sample of a full coefficient reproduction table for a relatively complex bernstein polynomial of high degree.

Control Point	Errors			Mean Value	Actual Value
	Min	Max	Mean		
007	0.00000e+00	1.59872e-14	4.82947e-15	5.00e+00	5.000000
016	1.77809e-16	1.94930e-13	6.38872e-14	5.00e-03	0.005000
025	5.77316e-15	2.02904e-12	5.79967e-13	3.00e+00	3.000000
034	1.57652e-13	1.13469e-11	3.05504e-12	2.00e+00	2.000000
043	2.18048e-13	4.56328e-11	1.17549e-11	3.00e+00	3.000000
052	7.51843e-13	1.46119e-10	3.69189e-11	4.00e+00	4.000000
061	6.36735e-12	3.96085e-10	9.95393e-11	5.00e+00	5.000000
070	2.42273e-12	9.46626e-10	2.37949e-10	1.00e-00	1.000000
106	7.10543e-15	3.91687e-13	1.04444e-13	6.00e+00	6.000000
115	8.88178e-16	4.86722e-13	1.34712e-13	7.00e+00	7.000000
124	2.66454e-14	2.36788e-12	6.20531e-13	8.00e+00	8.000000
133	7.28306e-14	1.21112e-11	2.75654e-12	9.00e+00	9.000000
142	7.01661e-14	4.46212e-11	1.03707e-11	6.00e+00	6.000000
151	5.42677e-13	1.34755e-10	3.28069e-11	7.00e+00	7.000000
160	2.58638e-12	3.55882e-10	8.96068e-11	1.11e+02	111.000000
205	1.24345e-14	2.19735e-12	6.13759e-13	8.00e+00	8.000000
214	1.77636e-15	2.55085e-12	4.80366e-13	9.00e+00	9.000000
223	4.66294e-15	3.59579e-12	7.61844e-13	1.00e-00	1.000000
232	1.19904e-14	1.01770e-11	2.65035e-12	2.00e+00	2.000000
241	3.73035e-14	3.48117e-11	9.69012e-12	3.00e+00	3.000000
250	1.59872e-13	1.10490e-10	3.06328e-11	4.00e+00	4.000000
304	4.61853e-14	1.02638e-11	2.58366e-12	3.00e+01	30.000000
313	2.66454e-15	1.38538e-11	2.51026e-12	5.00e+00	5.000000
322	1.59872e-14	1.95159e-11	2.99395e-12	6.00e+00	6.000000
331	1.69642e-13	3.17772e-11	4.63055e-12	7.00e+00	7.000000
340	1.33227e-13	4.70504e-11	1.11114e-11	1.00e+01	10.000000
403	3.09086e-13	4.42224e-11	9.27451e-12	8.00e+00	8.000000
412	1.84741e-13	6.01226e-11	1.00828e-11	9.00e+00	9.000000
421	4.07737e-13	8.18542e-11	1.15036e-11	-6.47e-12	0.000000
430	1.86073e-13	9.99336e-11	1.37040e-11	1.23e+00	1.234500
502	1.45040e-12	1.57472e-10	2.99923e-11	5.00e+00	5.000000
511	6.56444e-13	2.05262e-10	3.26794e-11	1.48e-11	0.000000
520	1.62270e-12	2.63876e-10	3.55498e-11	4.00e+00	4.000000
601	4.49418e-13	4.66215e-10	8.39634e-11	7.00e+00	7.000000
610	3.21121e-12	5.86634e-10	8.91119e-11	2.00e+00	2.000000
700	4.42801e-12	1.19451e-09	2.05288e-10	3.00e+00	3.000000

Table 4.1: These coefficient accuracy results are for a weighted septic Bernstein fit with no tangents. The function in question is complex. See the Actual Coefficient entries in the table.

	Errors				
Control Point	Min	Max	Mean	Mean Value	Actual Value
009	1.46551e-21	6.81364e-19	1.96788e-19	6.56e-20	0.000000
018	1.77591e-19	4.08945e-17	1.04657e-17	-6.66e-18	0.000000
027	1.80298e-18	3.26960e-16	9.14694e-17	7.80e-17	0.000000
036	4.85228e-19	1.82184e-15	6.03416e-16	-5.79e-16	0.000000
045	3.00711e-17	8.00409e-15	3.01354e-15	2.98e-15	0.000000
054	7.32384e-16	2.94291e-14	1.17735e-14	-1.16e-14	0.000000
063	1.93124e-15	9.12922e-14	3.68829e-14	3.62e-14	0.000000
072	1.12605e-15	2.51944e-13	1.00497e-13	-9.77e-14	0.000000
081	7.48738e-16	6.39431e-13	2.48197e-13	2.37e-13	0.000000
090	5.99520e-15	1.51923e-12	5.70625e-13	1.00e-00	1.000000
108	2.68758e-20	2.28869e-17	6.12233e-18	-2.87e-19	0.000000
117	8.63603e-20	3.84652e-17	1.20269e-17	-9.23e-18	0.000000
126	1.27921e-17	2.75177e-16	1.21005e-16	1.21e-16	0.000000
135	1.40477e-16	1.62558e-15	7.33262e-16	-7.28e-16	0.000000
144	1.44292e-16	7.14961e-15	3.11977e-15	3.08e-15	0.000000
153	3.88589e-17	2.56335e-14	1.07160e-14	-1.05e-14	0.000000
162	1.85958e-15	8.12071e-14	3.19761e-14	3.07e-14	0.000000
171	2.01728e-15	2.28775e-13	8.61972e-14	-8.02e-14	0.000000
180	4.14699e-15	5.92719e-13	2.17118e-13	1.95e-13	0.000000
207	5.15609e-19	2.10942e-16	5.50911e-17	4.12e-18	0.000000
216	7.25786e-19	2.01325e-16	5.63347e-17	-2.34e-17	0.000000
225	1.22235e-18	3.26042e-16	1.40576e-16	1.32e-16	0.000000
234	3.10508e-17	1.51810e-15	6.76432e-16	-6.70e-16	0.000000
243	8.60648e-18	6.25465e-15	2.55268e-15	2.46e-15	0.000000
252	6.70908e-16	2.26979e-14	8.71479e-15	-8.04e-15	0.000000
261	5.85298e-16	7.20564e-14	2.61331e-14	2.29e-14	0.000000
270	1.78759e-17	2.10907e-13	7.54921e-14	-6.46e-14	0.000000
306	5.65265e-18	1.18288e-15	3.13697e-16	-2.73e-17	0.000000
315	5.16039e-18	1.00702e-15	3.04221e-16	5.26e-18	0.000000
324	1.05951e-17	1.12017e-15	3.43543e-16	1.42e-16	0.000000
333	7.85488e-18	1.59416e-15	5.69487e-16	-3.40e-16	0.000000
342	3.78907e-17	5.57193e-15	2.13843e-15	1.81e-15	0.000000
351	6.47163e-17	1.90905e-14	6.62571e-15	-4.61e-15	0.000000
360	1.04233e-16	6.97288e-14	2.57439e-14	2.12e-14	0.000000
405	1.35943e-17	5.09233e-15	1.40382e-15	1.15e-16	0.000000
414	9.33345e-19	4.57756e-15	1.38983e-15	-8.09e-17	0.000000
423	1.10406e-17	4.61750e-15	1.48104e-15	-2.25e-16	0.000000
432	6.67998e-17	5.44858e-15	1.70509e-15	-8.29e-16	0.000000
441	6.49219e-17	8.48592e-15	2.80054e-15	-8.51e-16	0.000000
450	1.86845e-16	3.21952e-14	1.19025e-14	-1.06e-14	0.000000
504	1.45653e-17	1.84233e-14	5.34925e-15	-3.32e-16	0.000000
513	8.03626e-17	1.74513e-14	5.36423e-15	3.91e-16	0.000000
522	6.31087e-17	1.84151e-14	5.70323e-15	1.41e-15	0.000000
531	1.81153e-18	2.03172e-14	6.76486e-15	3.46e-15	0.000000
540	2.43017e-16	3.10906e-14	1.28710e-14	1.13e-14	0.000000
603	1.87962e-16	5.71975e-14	1.76776e-14	5.71e-16	0.000000
612	2.09780e-18	5.80555e-14	1.80162e-14	-2.02e-15	0.000000
621	3.19737e-16	6.21482e-14	1.92708e-14	-6.24e-15	0.000000
630	5.87220e-16	7.08900e-14	2.50588e-14	-1.62e-14	0.000000
702	2.06743e-16	1.65976e-13	5.26020e-14	1.61e-16	0.000000
711	1.70681e-16	1.72889e-13	5.41436e-14	8.62e-15	0.000000
720	9.73354e-16	1.87511e-13	5.91750e-14	2.36e-14	0.000000
801	3.57368e-15	4.44387e-13	1.42866e-13	-5.69e-15	0.000000
810	6.63376e-15	4.68091e-13	1.47463e-13	-3.04e-14	0.000000
900	1.00985e-14	1.09893e-12	3.56773e-13	2.68e-14	0.000000

Table 4.2: These coefficient accuracy results are for a weighted nonic Bernstein fit with no tangents. The function in question is $f(x, y) = 1 \cdot B_{090}(x, y)$

Surface and Fit Degree	Bernstein		Monomial	
	Weighting	No Weighting	Weighting	No Weighting
s1, f3	X	X	1.729727e-13	1.727507e-13
s1, f4	X	X	8.783221e-13	8.779101e-13
s1, f5	X	X	5.477327e-12	3.326563e-12
s1, f6	X	X	5.036224e-11	1.214701e-11
s1, f7	X	X	6.870112e-10	2.732676e-10
s1, f8	X	X	8.598440e-10	6.131475e-10
s1, f9	X	X	3.384125e-09	1.142827e-09
s2, f3	X	X	3.734314e-15	3.743661e-15
s2, f4	X	X	7.093970e-14	7.097642e-14
s2, f5	X	X	2.720550e-13	5.478508e-13
s2, f6	X	X	2.812065e-12	1.600094e-12
s2, f7	X	X	2.955039e-10	1.905864e-10
s2, f8	X	X	5.398838e-10	5.007368e-10
s2, f9	X	X	1.513012e-09	6.668481e-10
s3, f3	2.220446e-16	2.220446e-16	2.789578e-13	2.781970e-13
s3, f4	X	X	2.297579e-12	2.293224e-12
s3, f5	X	X	1.317813e-11	1.101230e-11
s3, f6	X	X	1.034484e-10	2.838757e-11
s3, f7	X	X	1.398331e-09	8.376791e-10
s3, f8	X	X	1.469111e-09	1.362308e-09
s3, f9	X	X	7.144527e-09	2.627218e-09
s4, f4	5.551115e-16	5.551115e-16	3.678111e-12	3.674583e-12
s4, f5	X	X	3.019079e-11	1.460651e-11
s4, f6	X	X	1.007205e-10	6.611730e-11
s4, f7	X	X	2.309701e-09	1.379989e-09
s4, f8	X	X	2.822068e-09	2.605346e-09
s4, f9	X	X	1.027299e-08	3.109295e-09
s5, f5	4.440892e-16	4.440892e-16	1.632914e-11	1.159513e-11
s5, f6	X	X	1.181672e-10	4.189979e-11
s5, f7	X	X	1.294341e-09	6.743031e-10
s5, f8	X	X	1.626215e-09	1.385765e-09
s5, f9	X	X	6.002925e-09	1.611873e-09
s6, f6	8.837278e-16	8.837278e-16	7.348534e-11	2.059174e-11
s6, f7	X	X	1.224705e-09	4.706192e-10
s6, f8	X	X	1.502941e-09	7.177839e-10
s6, f9	X	X	4.562886e-09	1.959253e-09
s7, f7	1.194511e-09	1.194511e-09	1.314382e-09	4.755068e-10
s7, f8	X	X	1.048037e-09	8.278925e-10
s7, f9	X	X	1.331299e-08	5.188043e-09
s8, f8	1.108102e-11	1.108102e-11	5.970250e-10	5.489429e-10
s8, f9	X	X	2.986733e-09	9.668437e-10
s9, f9	1.519229e-12	1.519229e-12	8.978815e-09	4.556168e-09

Table 4.3: These coefficient reproduction results are for all point only estimation schemes. The values in this table reflect the maximum absolute error for any coefficient in the test. Bernstein fits were performed only for Bernstein polynomials of the same degree using a grid with degree+1 points on a side over $(0,0)-(.5,.5)$. Monomial fits were done using a 21×21 point grid over $(-1,-1)-(1,1)$. The test performed is written sn-fm where n is the degree of the *s*urface being estimated and m is the degree of the polynomial used for *f*itting in the estimator.

4.1.2 Derivative Accuracy

Having shown that the fitter correctly determines the unique polynomial interpolant of the points in question I check to see if the derivatives I get from them are correct enough to use. Testing the scheme involves analyzing the maximum error for 3^{rd},4^{th}, 5^{th}, and 9^{th} degree fits on equal or lower degree polynomials.

Because of the sheer number of tables that would be required to report all the results pertinent to this test I have constructed tables that report only the maximum absolute error for all estimates or highest order partial derivatives needed degree 5 surfaces: 3rd (see Tables 4.4 and 4.6), and degree 9 surfaces: 5th (see Tables 4.5 and 4.7).

Surface &Fit Degree	Bernstein		Monomial	
	Weighted	Plain	Weighted	Plain
s1-f3	2.76e-10(0.00e+00)	1.14e-12(0.00e+00)	6.67e-11(0.00e+00)	3.73e-13(0.00e+00)
s1-f4	3.71e-11(0.00e+00)	2.12e-12(0.00e+00)	2.69e-11(0.00e+00)	6.92e-13(0.00e+00)
s1-f5	8.60e-11(0.00e+00)	6.64e-12(0.00e+00)	5.06e-11(0.00e+00)	1.28e-12(0.00e+00)
s1-f9	5.48e-01(0.00e+00)	5.48e-01(0.00e+00)	4.96e-09(0.00e+00)	1.50e-10(0.00e+00)
s2-f3	3.22e-11(0.00e+00)	5.42e-13(0.00e+00)	1.21e-12(0.00e+00)	1.00e-14(0.00e+00)
s2-f4	6.61e-11(0.00e+00)	2.01e-12(0.00e+00)	2.52e-12(0.00e+00)	7.44e-14(0.00e+00)
s2-f5	1.44e-09(0.00e+00)	2.68e-12(0.00e+00)	6.76e-12(0.00e+00)	3.45e-13(0.00e+00)
s2-f9	2.07e-01(0.00e+00)	2.07e-01(0.00e+00)	2.29e-09(0.00e+00)	9.75e-11(0.00e+00)
s3-f3	1.90e-10(0.00e+00)	1.66e-12(6.00e+00)	6.47e-11(0.00e+00)	7.66e-13(6.00e+00)
s3-f4	2.46e-10(0.00e+00)	8.08e-12(6.00e+00)	3.71e-10(6.00e+00)	2.57e-12(6.00e+00)
s3-f5	1.89e-10(6.00e+00)	7.88e-12(0.00e+00)	1.25e-10(6.00e+00)	6.65e-12(6.00e+00)
s3-f9	7.38e-01(0.00e+00)	7.38e-01(0.00e+00)	6.60e-09(6.00e+00)	2.88e-10(0.00e+00)
s4-f3	2.89e+02(0.00e+00)	6.00e-01(-6.00e+00)	7.14e+01(0.00e+00)	6.00e-01(-6.00e+00)
s4-f4	3.55e-10(0.00e+00)	7.69e-12(0.00e+00)	1.73e-10(0.00e+00)	3.49e-12(0.00e+00)
s4-f5	2.83e-10(0.00e+00)	1.65e-11(0.00e+00)	4.13e-10(0.00e+00)	9.81e-12(0.00e+00)
s4-f9	1.31e+00(0.00e+00)	1.31e+00(0.00e+00)	4.46e-09(0.00e+00)	4.30e-10(0.00e+00)
s5-f3	1.02e+03(2.65e+01)	1.81e+01(-6.60e+01)	2.92e+02(1.13e+01)	1.81e+01(-6.60e+01)
s5-f4	3.42e+01(2.44e+01)	2.22e+00(-6.60e+01)	2.80e+01(-3.23e+01)	2.22e+00(-6.60e+01)
s5-f5	7.44e-10(3.05e+01)	1.43e-11(-6.38e+01)	1.39e-10(-6.02e+01)	7.89e-12(3.29e+01)
s5-f9	6.86e-01(1.80e+01)	6.86e-01(1.80e+01)	7.29e-09(3.02e+01)	1.75e-10(-6.10e+01)
s9-f3	2.14e+03(4.70e+01)	3.15e+02(5.28e+02)	6.18e+02(-2.67e+02)	3.15e+02(5.28e+02)
s9-f4	1.24e+03(2.60e+02)	1.51e+02(5.28e+02)	3.56e+02(4.87e+02)	1.51e+02(5.28e+02)
s9-f5	1.09e+02(-5.00e+02)	6.06e+01(5.28e+02)	1.72e+02(2.30e+01)	6.06e+01(5.28e+02)
s9-f9	1.09e+00(-4.98e+02)	1.09e+00(-4.98e+02)	1.10e-08(5.03e+02)	5.28e-10(-5.04e+02)

Table 4.4: These derivative accuracy results are for all point only estimation schemes. The values in this table reflect the maximum absolute error for all 3rd partial derivatives. The correct value is in brackets to show magnitude of the error.. The fits were done on a 21×21 point grid over $(-1,-1)-(1,1)$. The test performed is written sn-fm where n is the degree of the *s*urface being estimated and m is the degree of the polynomial used for *f*itting in the estimator.

Surface &Fit Degree	Bernstein		Monomial	
	Weighted	Plain	Weighted	Plain
s1-f3	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s1-f4	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s1-f5	2.12e-08(0.00e+00)	2.98e-10(0.00e+00)	1.08e-08(0.00e+00)	8.72e-11(0.00e+00)
s1-f9	3.64e+01(0.00e+00)	3.64e+01(0.00e+00)	1.41e-06(0.00e+00)	4.37e-08(0.00e+00)
s2-f3	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s2-f4	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s2-f5	4.58e-08(0.00e+00)	1.66e-10(0.00e+00)	2.27e-09(0.00e+00)	1.22e-11(0.00e+00)
s2-f9	1.23e+01(0.00e+00)	1.23e+01(0.00e+00)	7.57e-07(0.00e+00)	2.53e-08(0.00e+00)
s3-f3	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s3-f4	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s3-f5	4.03e-08(0.00e+00)	5.32e-10(0.00e+00)	2.27e-08(0.00e+00)	2.38e-10(0.00e+00)
s3-f9	3.38e+01(0.00e+00)	3.38e+01(0.00e+00)	1.99e-06(0.00e+00)	8.49e-08(0.00e+00)
s4-f3	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s4-f4	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s4-f5	4.74e-08(0.00e+00)	7.97e-10(0.00e+00)	4.21e-08(0.00e+00)	3.54e-10(0.00e+00)
s4-f9	2.87e+01(0.00e+00)	2.87e+01(0.00e+00)	1.26e-06(0.00e+00)	1.22e-07(0.00e+00)
s5-f3	1.20e+02(-1.20e+02)	1.20e+02(-1.20e+02)	1.20e+02(-1.20e+02)	1.20e+02(-1.20e+02)
s5-f4	1.20e+02(-1.20e+02)	1.20e+02(-1.20e+02)	1.20e+02(-1.20e+02)	1.20e+02(-1.20e+02)
s5-f5	9.88e-08(-1.20e+02)	8.76e-10(-1.20e+02)	2.81e-08(6.00e+01)	2.64e-10(6.00e+01)
s5-f9	1.98e+01(6.00e+01)	1.98e+01(6.00e+01)	2.09e-06(6.00e+01)	5.68e-08(-1.20e+02)
s9-f3	1.51e+04(1.51e+04)	1.51e+04(1.51e+04)	1.51e+04(1.51e+04)	1.51e+04(1.51e+04)
s9-f4	1.51e+04(1.51e+04)	1.51e+04(1.51e+04)	1.51e+04(1.51e+04)	1.51e+04(1.51e+04)
s9-f5	1.17e+04(-1.51e+04)	1.01e+04(1.51e+04)	1.22e+04(1.96e+03)	1.01e+04(-1.51e+04)
s9-f9	3.08e+01(0.00e+00)	3.08e+01(0.00e+00)	3.18e-06(1.51e+04)	1.62e-07(-1.51e+04)

Table 4.5: These derivative accuracy results are for all point only estimation schemes. The values in this table reflect the maximum absolute error for all 5th partial derivatives. The correct value is in brackets to show magnitude of the error.. The fits were done on a 21×21 point grid over $(-1,-1)-(1,1)$. The test performed is written sn-fm where n is the degree of the *s*urface being estimated and m is the degree of the polynomial used for *f*itting in the estimator.

Surface &Fit Degree	Bernstein		Monomial	
	Weighted	Plain	Weighted	Plain
s1-f3	5.55e-14(0.00e+00)	5.41e-14(0.00e+00)	1.51e-14(0.00e+00)	5.55e-15(0.00e+00)
s1-f4	1.62e-13(0.00e+00)	1.34e-13(0.00e+00)	3.60e-14(0.00e+00)	1.61e-14(0.00e+00)
s1-f5	2.54e-13(0.00e+00)	2.23e-13(0.00e+00)	1.11e-13(0.00e+00)	5.46e-14(0.00e+00)
s1-f9	6.49e-12(0.00e+00)	7.64e-12(0.00e+00)	9.96e-12(0.00e+00)	4.83e-12(0.00e+00)
s2-f3	1.28e-13(0.00e+00)	6.11e-14(0.00e+00)	1.51e-14(0.00e+00)	1.36e-14(0.00e+00)
s2-f4	2.79e-13(0.00e+00)	1.69e-13(0.00e+00)	2.80e-14(0.00e+00)	2.03e-14(0.00e+00)
s2-f5	5.23e-13(0.00e+00)	2.62e-13(0.00e+00)	1.65e-14(0.00e+00)	7.75e-14(0.00e+00)
s2-f9	9.59e-12(0.00e+00)	6.41e-12(0.00e+00)	1.27e-11(0.00e+00)	7.77e-12(0.00e+00)
s3-f3	3.34e-13(-2.00e+00)	1.63e-13(6.00e+00)	1.58e-13(-2.00e+00)	9.86e-14(6.00e+00)
s3-f4	7.96e-13(6.00e+00)	5.56e-13(6.00e+00)	2.54e-13(-2.00e+00)	1.65e-13(-2.00e+00)
s3-f5	1.24e-12(-2.00e+00)	6.41e-13(-2.00e+00)	4.26e-13(6.00e+00)	4.28e-13(6.00e+00)
s3-f9	3.37e-11(-2.00e+00)	1.30e-11(-2.00e+00)	2.44e-11(6.00e+00)	1.32e-11(6.00e+00)
s4-f3	2.47e-01(2.00e+00)	4.00e-01(2.00e+00)	2.47e-01(-6.00e+00)	4.00e-01(2.00e+00)
s4-f4	6.81e-13(0.00e+00)	5.03e-13(0.00e+00)	1.59e-13(4.00e-01)	1.35e-13(0.00e+00)
s4-f5	8.16e-13(0.00e+00)	4.71e-13(0.00e+00)	5.32e-13(0.00e+00)	3.06e-13(0.00e+00)
s4-f9	2.13e-11(2.00e+00)	1.08e-11(0.00e+00)	2.04e-11(0.00e+00)	1.39e-11(0.00e+00)
s5-f3	1.19e+01(-6.49e+01)	1.24e+01(-6.60e+01)	1.19e+01(-6.49e+01)	1.24e+01(-6.60e+01)
s5-f4	1.13e+00(-6.49e+01)	1.16e+00(-6.60e+01)	1.13e+00(-6.22e+01)	1.16e+00(-6.60e+01)
s5-f5	1.79e-12(1.80e+01)	1.19e-12(-6.60e+01)	1.05e-12(-6.60e+01)	6.82e-13(-6.60e+01)
s5-f9	2.70e-11(7.50e+00)	2.11e-11(-6.05e+01)	1.90e-11(-6.38e+01)	9.99e-12(-6.38e+01)
s9-f3	2.33e+02(5.28e+02)	2.36e+02(5.28e+02)	2.33e+02(5.28e+02)	2.36e+02(5.28e+02)
s9-f4	8.96e+01(5.28e+02)	9.08e+01(5.28e+02)	8.96e+01(5.28e+02)	9.08e+01(5.28e+02)
s9-f5	3.05e+01(5.28e+02)	3.07e+01(5.28e+02)	3.05e+01(5.28e+02)	3.07e+01(5.28e+02)
s9-f9	3.08e-11(5.03e+02)	2.43e-11(-5.04e+02)	1.06e-10(2.40e+01)	2.05e-11(8.23e+00)

Table 4.6: These derivative accuracy results are for all point only estimation schemes. The values in this table reflect the maximum absolute error for all 3rd partial derivatives. The correct value is in brackets to show magnitude of the error.. The fits were done on a 21×21 point grid over $(-1,-1)-(1,1)$. The test performed is written sn-fm where n is the degree of the *s*urface being estimated and m is the degree of the polynomial used for *f*itting in the estimator.

Surface &Fit Degree	Bernstein		Monomial	
	Weighted	Plain	Weighted	Plain
s1-f3	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s1-f4	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s1-f5	2.44e-11(0.00e+00)	1.60e-11(0.00e+00)	4.17e-12(0.00e+00)	9.25e-13(0.00e+00)
s1-f9	2.92e-09(0.00e+00)	3.45e-09(0.00e+00)	6.38e-09(0.00e+00)	1.79e-09(0.00e+00)
s2-f3	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s2-f4	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s2-f5	5.31e-11(0.00e+00)	1.81e-11(0.00e+00)	2.62e-12(0.00e+00)	3.65e-12(0.00e+00)
s2-f9	4.70e-09(0.00e+00)	3.17e-09(0.00e+00)	5.81e-09(0.00e+00)	3.06e-09(0.00e+00)
s3-f3	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s3-f4	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s3-f5	1.16e-10(0.00e+00)	4.34e-11(0.00e+00)	3.47e-11(0.00e+00)	2.33e-11(0.00e+00)
s3-f9	1.44e-08(0.00e+00)	6.29e-09(0.00e+00)	1.14e-08(0.00e+00)	4.88e-09(0.00e+00)
s4-f3	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s4-f4	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)
s4-f5	1.14e-10(0.00e+00)	3.74e-11(0.00e+00)	3.46e-11(0.00e+00)	1.55e-11(0.00e+00)
s4-f9	1.00e-08(0.00e+00)	4.80e-09(0.00e+00)	9.19e-09(0.00e+00)	6.69e-09(0.00e+00)
s5-f3	1.20e+02(-1.20e+02)	1.20e+02(-1.20e+02)	1.20e+02(-1.20e+02)	1.20e+02(-1.20e+02)
s5-f4	1.20e+02(-1.20e+02)	1.20e+02(-1.20e+02)	1.20e+02(-1.20e+02)	1.20e+02(-1.20e+02)
s5-f5	1.13e-10(0.00e+00)	7.37e-11(0.00e+00)	5.83e-11(-1.20e+02)	3.50e-11(-1.20e+02)
s5-f9	1.24e-08(0.00e+00)	9.68e-09(-1.20e+02)	7.42e-09(-1.20e+02)	3.97e-09(-1.20e+02)
s9-f3	1.51e+04(1.51e+04)	1.51e+04(1.51e+04)	1.51e+04(1.51e+04)	1.51e+04(1.51e+04)
s9-f4	1.51e+04(1.51e+04)	1.51e+04(1.51e+04)	1.51e+04(1.51e+04)	1.51e+04(1.51e+04)
s9-f5	8.74e+03(1.51e+04)	8.73e+03(-1.51e+04)	8.74e+03(1.51e+04)	8.73e+03(-1.51e+04)
s9-f9	1.48e-08(0.00e+00)	1.13e-08(-1.51e+04)	4.93e-08(0.00e+00)	1.11e-08(0.00e+00)

Table 4.7: These derivative accuracy results are for all point only estimation schemes. The values in this table reflect the maximum absolute error for all 5th partial derivatives. The correct value is in brackets to show magnitude of the error.. The fits were done on a 21×21 point grid over $(-1,-1)-(1,1)$. The test performed is written sn-fm where n is the degree of the *s*urface being estimated and m is the degree of the polynomial used for *f*itting in the estimator.

4.1.3 Polynomial Convergence

If an interpolation scheme is precise to a certain number of derivatives, the error for that scheme should converges as predicted by the Taylor error term. For d precise degrees at a sample distance of h the error term should be $O(h^{d+1})$. Doubling the sampling density halves the sample spacing and should produce a predictable ratio. For a quadratic this is

$$\frac{2^{2+1}}{1^{2+1}} = 2^3 = 8$$

for a cubic: 16, a quartic: 32, and so on.

Some BS about why I want to show polynomial precision goes here. Hell of a lot of good it does on non-polynomial surfaces. Talk about Taylor series in more depth? Something about how all continuous functions are differentiable, eg. sine functions can be differentiated and give numerical values at the derivatives. Integrating those derivatives as numbers, rather than as the functions they came from, causes a polynomial to pop out... Assuming that derivatives mean anything special, and I'm told they do, the polynomial that pops out is gonna be some kind of good approximation of the non-polynomial function that produced the derivatives.

My routines only provide point positions and partial derivatives for a mesh. However my scheme as a whole is intended to provide smooth surfaces with polynomial precision. As indicted in Stephen Mann's foundation work [3] the smoothness of a surface is influenced by continuity between patch boundaries. Based on his results that show his first method achieves better polynomial precision with consistently lower errors than any other method available to me[4], I will use Mann's first patch setting scheme with full available continuity adjustments. For a quick guide to the relative results provided by each of Mann's schemes refer to figure 4.8.

method	.2	.1	.05
1 d5	0.0499279	0.0031802(15.70)	3.88865e-05(81.78)
1 d9	0.0645471	0.00418636(15.42)	5.09379e-05(82.19)
2 d5	0.0476294	0.00304394(15.65)	0.00241605(1.26)
2 d9	0.0576738	0.00366265(15.75)	0.00186034(1.97)
3 d5	0.0502117	0.00318274(15.78)	3.92538e-05(81.08)
3 d9	0.0629325	0.00407998(15.42)	5.03143e-05(81.09)
1 d5 uncorrected	0.0507564	0.00328421(15.45)	3.9625e-05(82.88)
1 d9 uncorrected	0.0648539	0.00423641(15.31)	5.11935e-05(82.75)

Table 4.8: Comparison of continuity adjustors. Stephen Mann indicated that method 1 performed best. Method 3 seems to perform nearly as well. I threw a hard function at all three continuity adjustment methods. I chose the best one and did not permit continuity adjustments.

	Conv. Theory		Sample Spacing			
Method	Mann	Clarke	.1	.05	.025	.0125
Bern d5,d3	64	16	2.36287e-05	1.49613e-06(15.8)	9.38116e-08(15.95)	5.86798e-09(15.99)
Bern d5,d4		32	3.38371e-06	1.19371e-07(28.3)	3.84046e-09(31.1)	1.20882e-10(31.8)
Bern d5,d5		64	1.25028e-06	1.99334e-08(62.7)	3.13029e-10(63.7)	4.89719e-12(63.9)
Bern d5,d9		1024	1.14111e-08	1.61278e-10(70.8)	2.52565e-12(63.9)	4.03011e-14(62.7)
Bern d9,d3	1024	16	4.6304e-05	2.92444e-06(15.8)	1.83255e-07(15.96)	1.14609e-08(15.99)
Bern d9,d4		32	7.60793e-06	2.69352e-07(28.2)	8.67482e-09(31.0)	2.73717e-10(31.7)
Bern d9,d5		64	2.38743e-06	3.79773e-08(62.9)	5.96054e-10(63.7)	9.32376e-12(63.9)
Bern d9,d9		1024	5.42704e-09	5.11147e-12(1061.7)	5.88418e-15(868.68)	1.77636e-15(3.3)

Table 4.9: Maximum error convergence rates on surface S1 over [0,1] using point and tangent information. The estimator used was Bernstein basis with weighting.

	Conv. Theory		Sample Spacing			
Method	Mann	Clarke	.1	.05	.025	.0125
Bern d5,d3	64	16	1.79469e-05	1.13618e-06(15.8)	7.12388e-08(15.95)	4.45599e-09(15.99)
Bern d5,d4		32	1.72714e-06	6.06271e-08(28.5)	1.94834e-09(31.1)	6.1309e-11(31.77)
Bern d5,d5		64	6.07531e-07	9.67932e-09(62.8)	1.51976e-10(63.7)	2.37743e-12(63.9)
Bern d5,d9		1024	1.03161e-08	1.61576e-10(63.8)	2.52609e-12(63.96)	4.03011e-14(62.7)
Bern d9,d3	1024	16	3.49058e-05	2.20386e-06(15.8)	1.3809e-07(15.96)	8.6361e-09(15.99)
Bern d9,d4		32	3.51633e-06	1.24111e-07(28.3)	3.99456e-09(31.1)	1.25767e-10(31.8)
Bern d9,d5		64	1.15969e-06	1.84303e-08(62.9)	2.89197e-10(63.7)	4.52371e-12(63.93)
Bern d9,d9		1024	1.42045e-09	1.43352e-12(990.9)	2.77556e-15(516.5)	1.77636e-15(1.6)

Table 4.10: Maximum error convergence rates on surface S1 over [0,1] using point and tangent information. The estimator used was Bernstein basis with no weighting.

	Conv. Theory		Sample Spacing			
Method	Mann	Clarke	.1	.05	.025	.0125
Mono d5,d3	64	16	2.36287e-05	1.49613e-06(15.8)	9.38116e-08(15.95)	5.86798e-09(15.99)
Mono d5,d4	64	32	3.38371e-06	1.19371e-07(28.3)	3.84046e-09(31.1)	1.20882e-10(31.8)
Mono d5,d5	64	64	1.25028e-06	1.99334e-08(62.7)	3.13029e-10(63.7)	4.89719e-12(63.9)
Mono d5,d9	64	1024	1.14111e-08	1.61278e-10(70.6)	2.52565e-12(63.9)	4.03011e-14(62.7)
Mono d9,d3	1024	16	4.6304e-05	2.92444e-06(15.8)	1.83255e-07(15.96)	1.14609e-08(15.99)
Mono d9,d4	1024	32	7.60793e-06	2.69352e-07(28.2)	8.67482e-09(31.0)	2.73717e-10(31.7)
Mono d9,d5	1024	64	2.38743e-06	3.79773e-08(62.9)	5.96054e-10(63.7)	9.32376e-12(63.9)
Mono d9,d9	1024	1024	5.42704e-09	5.11147e-12(1061.7)	5.88418e-15(868.7)	1.77636e-15(3.3)

Table 4.11: Maximum error convergence rates on surface S1 over [0,1] using point and tangent information. The estimator used was monomial basis with weighting.

	Conv. Theory		Sample Spacing			
Method	Mann	Clarke	.1	.05	.025	.0125
Mono d5,d3	64	16	1.79469e-05	1.13618e-06(15.8)	7.12388e-08(15.95)	4.45599e-09(15.99)
Mono d5,d4		32	1.72714e-06	6.06271e-08(28.5)	1.94834e-09(31.1)	6.1309e-11(31.8)
Mono d5,d5		64	6.07531e-07	9.67932e-09(62.8)	1.51976e-10(63.7)	2.37743e-12(63.92)
Mono d5,d9		1024	1.03161e-08	1.61576e-10(63.8)	2.52609e-12(63.96)	4.03011e-14(62.7)
Mono d9,d3	1024	16	3.49058e-05	2.20386e-06(15.8)	1.3809e-07(15.96)	8.6361e-09(15.99)
Mono d9,d4		32	3.51633e-06	1.24111e-07(28.3)	3.99456e-09(31.1)	1.25767e-10(31.8)
Mono d9,d5		64	1.15969e-06	1.84303e-08(62.92)	2.89197e-10(63.7)	4.52371e-12(63.9)
Mono d9,d9		1024	1.42045e-09	1.43352e-12(990.9)	2.77556e-15(516.5)	1.77636e-15(1.6)

Table 4.12: Maximum error convergence rates on surface S1 over [0,1] using point and tangent information. The estimator used was monomial basis with no weighting.

	Conv. Theory		Sample Spacing			
Method	Mann	Clarke	.1	.05	.025	.0125
Bern d5,d3			0.000104692	6.42587e-06(16.3)	3.97045e-07(16.2)	2.46607e-08(16.1)
Bern d5,d4			1.44673e-05	5.39269e-07(26.8)	1.75728e-08(30.7)	5.55013e-10(31.7)
Bern d5,d5			3.64209e-06	4.42408e-08(82.3)	6.11082e-10(72.4)	8.73623e-12(70.0)
Bern d5,d9			5.12957e-05	4.3973e-05(1.2)	4.55502e-05(0.97)	4.66068e-05(0.98)
Bern d9,d3			0.000105462	6.42715e-06(16.4)	3.97091e-07(16.2)	2.46624e-08(16.1)
Bern d9,d4			1.48727e-05	5.52515e-07(26.9)	1.79893e-08(30.7)	5.68019e-10(31.7)
Bern d9,d5			4.11277e-06	5.28212e-08(77.9)	8.04821e-10(65.6)	1.11341e-11(72.3)
Bern d9,d9			7.09052e-05	6.08017e-05(1.2)	6.30212e-05(0.96)	6.45157e-05(0.98)

Table 4.13: Maximum error convergence rates on surface S1 over [0,1] using point position information only. The estimator used was Bernstein basis with weighting.

	Conv. Theory		Sample Spacing			
Method	Mann	Clarke	.1	.05	.025	.0125
Bern d5,d3	64	16	0.000104607	6.42424e-06(16.3)	3.97066e-07(16.2)	2.46659e-08(16.1)
Bern d5,d4		32	1.44766e-05	5.39606e-07(26.8)	1.75835e-08(30.7)	5.55348e-10(31.7)
Bern d5,d5		64	3.47381e-06	5.3333e-08(82.3)	8.21002e-10(72.4)	1.27168e-11(70.0)
Bern d5,d9		1024	5.12675e-05	4.3973e-05(1.2)	4.55502e-05(0.97)	4.66068e-05(0.98)
Bern d9,d3	1024	16	0.000105326	6.42424e-06(16.4)	3.97066e-07(16.2)	2.46659e-08(16.1)
Bern d9,d4		32	1.48837e-05	5.52897e-07(26.9)	1.80013e-08(30.7)	5.68393e-10(31.7)
Bern d9,d5		64	3.49611e-06	5.35896e-08(77.9)	8.2343e-10(65.6)	1.27405e-11(72.3)
Bern d9,d9		1024	7.0871e-05	6.08017e-05(1.2)	6.30212e-05(0.96)	6.45157e-05(0.98)

Table 4.14: Maximum error convergence rates on surface S1 over [0,1] using point position information only. The estimator used was Bernstein basis with no weighting.

	Conv. Theory		Sample Spacing			
Method	Mann	Clarke	.1	.05	.025	.0125
Mono d5,d3			0.000104692	6.42587e-06(16.3)	3.97045e-07(16.2)	2.46607e-08(16.1)
Mono d5,d4			1.44673e-05	5.39269e-07(26.8)	1.75728e-08(30.7)	5.55013e-10(31.7)
Mono d5,d5			4.38879e-06	6.75571e-08(65.0)	1.00747e-09(67.1)	1.57635e-11(63.91)
Mono d5,d9			1.96729e-08	1.66155e-10(118.4)	2.54008e-12(65.4)	4.03011e-14(63.0)
Mono d9,d3			0.000105462	6.42715e-06(16.4)	3.97091e-07(16.2)	2.46624e-08(16.1)
Mono d9,d4			1.48727e-05	5.52515e-07(26.9)	1.79893e-08(30.7)	5.68019e-10(31.7)
Mono d9,d5			4.92451e-06	7.17008e-08(68.7)	1.06923e-09(67.1)	1.64412e-11(65.0)
Mono d9,d9			1.64732e-08	1.42563e-11(1155.5)	1.62093e-14(879.5)	2.44249e-15(6.6)

Table 4.15: Maximum error convergence rates on surface S1 over [0,1] using point position information only. The estimator used was monomial basis with weighting.

	Conv. Theory		Sample Spacing			
Method	Mann	Clarke	.1	.05	.025	.0125
Mono d5,d3	64	16	0.000104607	6.42424E-06(16.3)	3.97066E-07(16.2)	2.46659E-08(16.1)
Mono d5,d4		32	1.44766E-05	5.39606E-07(26.8)	1.75835E-08(30.7)	5.55348E-10(31.7)
Mono d5,d5		64	3.47381E-06	5.3333E-08(65.1)	8.21002E-10(65.0)	1.27168E-11(64.6)
Mono d5,d9		1024	1.44815E-08	1.65667E-10(87.4)	2.52942E-12(65.5)	4.03011E-14(62.8)
Mono d9,d3	1024	16	0.000105326	6.42424E-06(16.4)	3.97066E-07(16.2)	2.46659E-08(16.1)
Mono d9,d4		32	1.48837E-05	5.52897E-07(26.9)	1.80013E-08(30.7)	5.68393E-10(31.7)
Mono d9,d5		64	3.49611E-06	5.35896E-08(65.3)	8.2343E-10(65.1)	1.27405E-11(64.6)
Mono d9,d9		1024	6.50887E-09	4.3161E-12(1508.0)	4.44089E-15(971.9)	1.88738E-15(2.4)

Table 4.16: Maximum error convergence rates on surface S1 over [0,1] using point position information only. The estimator used was monomial basis without weighting.

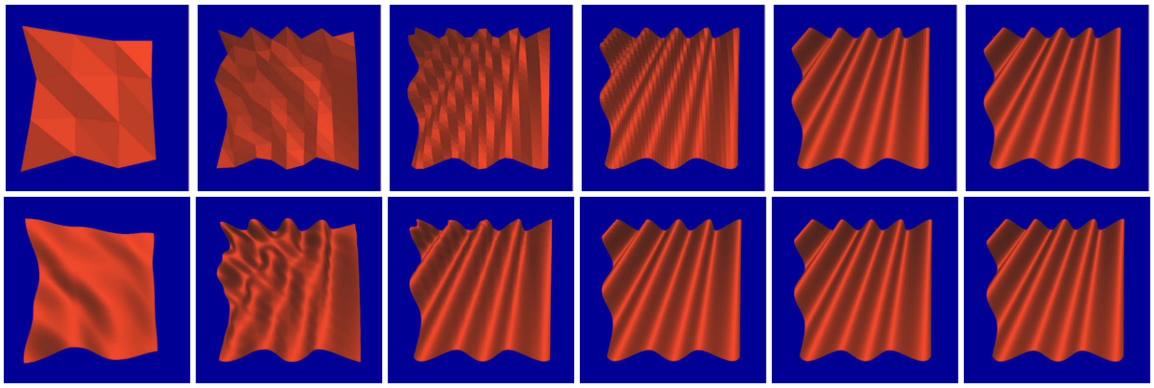

Figure 4.1: This figure should help visualize when polynomial convergence begins to kick in. Errors and convergence rates for these surfaces are, from left to right, $h_0 = 6 : 0.201997; h_1 = 11 : 0.179627, \frac{h_0}{h_1} = 1.1; h_2 = 21 : 0.045317, \frac{h_1}{h_2} = 4; h_3 = 41 : 0.00429185, \frac{h_0}{h_1} = 10.6; h_4 = 81 : 0.000377487, \frac{h_0}{h_1} = 11.4; h_5 = 161 : 2.5479e - 05, \frac{h_0}{h_1} = 14.8$

Note: to get the Polynomial Convergence behaviour from a polynomial derivative estimator we need all significant surface features to be present at the sampling densities we're using. Maybe I'll produce an example of a rapidly varying sine function at densities that are too sparse say $c * sin(\frac{2*x*\pi*5}{y+1})$ sampled 6, 11, 21, 41, and 81 times on $(0,0)-(1,1)$. That sounds like fun. Here goes! In figure 4.1 see at $y = 0$ that there are 5 full periods on x and at $y = 1$ there are only 2.5. There are two features per period, a high and a low. So we should begin to see convergence once we pass 1 sample per feature (that's recognizability on the big end at between 6 and 11 samples), and once we pass 2 samples per feature we should be converging well (small end at 21 to 41 samples). I suspect that full convergence won't begin until all the fitting points are working on the same feature. I'm doing a degree 3 fit, so look for convergence around 4 samples per feature (samples > 81, until we're out of machine precision, or time, or memory.

	Sample Spacing			
Method	.1	.05	.025	.0125
mono d5-d3	0.00432878	0.000620344(6.98)	6.20037e-05(10.00)	4.41299e-06(14.05)
mono d5-d4	0.00314496	0.000336009(9.36)	1.57898e-05(21.28)	5.21906e-07(30.25)
mono d5-d5	0.00368548	0.000178879(20.60)	5.76456e-06(31.03)	1.11661e-07(51.63)
mono d5-d9	0.00887377	5.95236e-05(149.08)	3.82512e-07(155.61)	8.07125e-09(47.39)
mono d9-d3	0.00791327	0.00137463(5.76)	0.000121984(11.27)	8.51518e-06(14.33)
mono d9-d4	0.00610741	0.000809075(7.55)	3.39616e-05(23.82)	1.06651e-06(31.84)
mono d9-d5	0.00647368	0.000440846(14.68)	1.17647e-05(37.47)	2.19638e-07(53.56)
mono d9-d9	0.0122225	9.70491e-05(125.94)	2.91801e-07(332.59)	2.94602e-09(99.05)
MFarin	0.00996975	0.000880518(11.32)	6.01092e-05(14.65)	3.92236e-06(15.32)

Table 4.17: *Franke 1:* Comparison of point and 1$^{\text{st}}$ derivative interpolant errors for variants of the monomial based derivative estimation scheme and Clough-Tocher.

4.2 Evaluations

4.2.1 Clarke-Mann vs Clough-Tocher

The most widely recognized spline based interpolator is the Clough-Tocher method. While others exist, this one works well and other methods in the category compare themselves to it. The best Clough-Tocher type interpolator is a modified Farin implementation. I will compare some maximum surface errors and visuals between Clarke-Mann and Clough-Tocher.

4.2.2 Franke

My driver programs are not currently up to the task of reading an arbitrary set of points, triangulating it and estimating it yet. The reader would be easy, the triangulator probably already exists, and I'd need to implement k-Nearest-Neighbours in some form or another. So I will not run my method on Franke's datasets and compare my results to what is reported in [9]. Instead checked the error of some randomized triangulations of F1–F6,rendered some Gouraud shaded pics, and rendered some Gaussian curvature pictures to show how the estimator performs. See table **??** and figures 4.2 to 4.13.

Method	Sample Spacing			
	.1	.05	.025	.0125
mono d5-d3	0.00170322	0.000256699(6.64)	2.53832e-05(10.11)	2.12314e-06(11.96)
mono d5-d4	0.00175063	0.000202246(8.66)	6.5928e-06(30.68)	3.95031e-07(16.69)
mono d5-d5	0.00123908	0.000135379(9.15)	2.7135e-06(49.89)	7.76718e-08(34.94)
mono d5-d9	0.00218724	8.48791e-05(25.77)	1.09502e-06(77.51)	1.50975e-07(7.25)
mono d9-d3	0.00264042	0.00049046(5.38)	5.06427e-05(9.68)	3.85502e-06(13.14)
mono d9-d4	0.00309595	0.000290028(10.67)	1.67764e-05(17.29)	7.65487e-07(21.92)
mono d9-d5	0.00157835	0.000170945(9.23)	6.82677e-06(25.04)	1.68101e-07(40.61)
mono d9-d9	0.00372608	0.000115018(32.4)	1.05864e-06(108.65)	2.33066e-07(4.54)
MFarin	0.00473717	0.000665899(7.11)	4.62304e-05(14.40)	2.96625e-06(15.58)

Table 4.18: *Franke 2:* Comparison of point and 1st derivative interpolant errors for variants of the monomial based derivative estimation scheme and Clough-Tocher.

Method	Sample Spacing			
	.1	.05	.025	.0125
mono d5-d3	0.000267408	2.60948e-05(10.25)	1.83419e-06(14.23)	1.17885e-07(15.56)
mono d5-d4	0.000146329	6.46967e-06(22.62)	2.39649e-07(27)	7.9054e-09(30.31)
mono d5-d5	7.54483e-05	2.48541e-06(30.36)	4.73975e-08(52.44)	8.00403e-10(59.22)
mono d5-d9	0.000119095	3.12423e-07(381.2)	1.68873e-09(185)	3.15267e-11(53.57)
mono d9-d3	0.000562004	5.2576e-05(10.69)	3.54928e-06(14.81)	2.28211e-07(15.55)
mono d9-d4	0.000276995	1.35193e-05(20.49)	4.92449e-07(27.45)	1.61921e-08(30.41)
mono d9-d5	0.00015102	5.29267e-06(28.53)	9.9474e-08(53.21)	1.6723e-09(59.48)
mono d9-d9	0.000177508	4.69597e-07(378)	6.0791e-10(772.48)	1.83019e-11(33.22)
MFarin	0.00543834	0.00246574(2.21)	0.00120089(2.05)	0.000598596(2.01)

Table 4.19: *Franke 3:* Comparison of point and 1st derivative interpolant errors for variants of the monomial based derivative estimation scheme and Clough-Tocher.

Method	Sample Spacing			
	.1	.05	.025	.0125
mono d5-d3	0.000104485	7.3783e-06(14.16)	4.75466e-07(15.52)	2.99449e-08(15.88)
mono d5-d4	2.56497e-05	8.77194e-07(29.24)	2.70853e-08(32.39)	8.36648e-10(32.37)
mono d5-d5	9.64182e-06	1.85394e-07(52.01)	3.0519e-09(60.75)	4.83025e-11(63.18)
mono d5-d9	7.42968e-07	1.34346e-08(55.3)	2.17189e-10(61.86)	3.4191e-12(63.52)
mono d9-d3	0.000205303	1.42129e-05(14.44)	9.1146e-07(15.59)	5.73344e-08(15.9)
mono d9-d4	5.5852e-05	1.80047e-06(31.02)	5.44702e-08(33.05)	1.66035e-09(32.81)
mono d9-d5	1.95357e-05	3.63733e-07(53.71)	5.95089e-09(61.12)	9.40517e-11(63.27)
mono d9-d9	5.27489e-07	1.33398e-09(395.42)	8.18401e-13(1629.98)	2.49911e-13(3.27)
MFarin	0.000101986	6.57157e-06(15.52)	4.13864e-07(15.88)	2.59158e-08(15.96)

Table 4.20: *Franke 4:* Comparison of point and 1st derivative interpolant errors for variants of the monomial based derivative estimation scheme and Clough-Tocher.

Method	Sample Spacing			
	.1	.05	.025	.0125
mono d5-d3	0.00102744	0.000104485(9.83)	7.3783e-06(14.16)	4.75466e-07(15.52)
mono d5-d4	0.00055346	2.56497e-05(21.58)	8.77194e-07(29.24)	2.70853e-08(32.39)
mono d5-d5	0.000287575	9.64182e-06(29.83)	1.85394e-07(52.01)	3.0519e-09(60.75)
mono d5-d9	0.000187556	6.46111e-07(290.28)	1.34346e-08(48.09)	2.17186e-10(61.86)
mono d9-d3	0.0022028	0.000205303(10.73)	1.42129e-05(14.44)	9.1146e-07(15.59)
mono d9-d4	0.00137759	5.5852e-05(24.67)	1.80047e-06(31.02)	5.44702e-08(33.05)
mono d9-d5	0.00066024	1.95357e-05(33.8)	3.63733e-07(53.71)	5.95089e-09(61.12)
mono d9-d9	0.000251441	4.87334e-07(515.95)	7.00285e-10(695.91)	4.44719e-12(157.47)
MFarin	0.00144386	0.000101986(14.16)	6.57157e-06(15.52)	4.13864e-07(15.87)

Table 4.21: *Franke 5:* Comparison of point and 1^{st} derivative interpolant errors for variants of the monomial based derivative estimation scheme and Clough-Tocher.

Method	Sample Spacing			
	.1	.05	.025	.0125
mono d5-d3	5.18267e-05	7.42746e-06(6.98)	8.14952e-07(9.11)	7.21177e-08(11.3)
mono d5-d4	1.37625e-05	1.38569e-06(9.93)	9.48247e-08(14.61)	4.80545e-09(19.73)
mono d5-d5	3.72578e-06	2.71837e-07(13.71)	1.17137e-08(23.21)	3.80598e-10(30.78)
mono d5-d9	8.23377e-07	5.93807e-08(13.87)	2.55885e-09(23.21)	9.24133e-11(27.69)
mono d9-d3	8.64031e-05	1.26899e-05(6.81)	1.40828e-06(9.01)	1.27124e-07(11.08)
mono d9-d4	2.02062e-05	2.13135e-06(9.48)	1.51828e-07(14.04)	8.03468e-09(18.9)
mono d9-d5	5.03416e-06	4.07762e-07(12.35)	2.01967e-08(20.19)	7.32635e-10(27.57)
mono d9-d9	2.18172e-07	7.47774e-09(29.18)	1.02085e-10(73.25)	8.00395e-11(1.28)
MFarin	3.75679e-05	6.73025e-06(5.58)	8.41358e-07(8.00)	7.93016e-08(10.61)

Table 4.22: *Franke 6:* Comparison of point and 1^{st} derivative interpolant errors for variants of the monomial based derivative estimation scheme and Clough-Tocher.

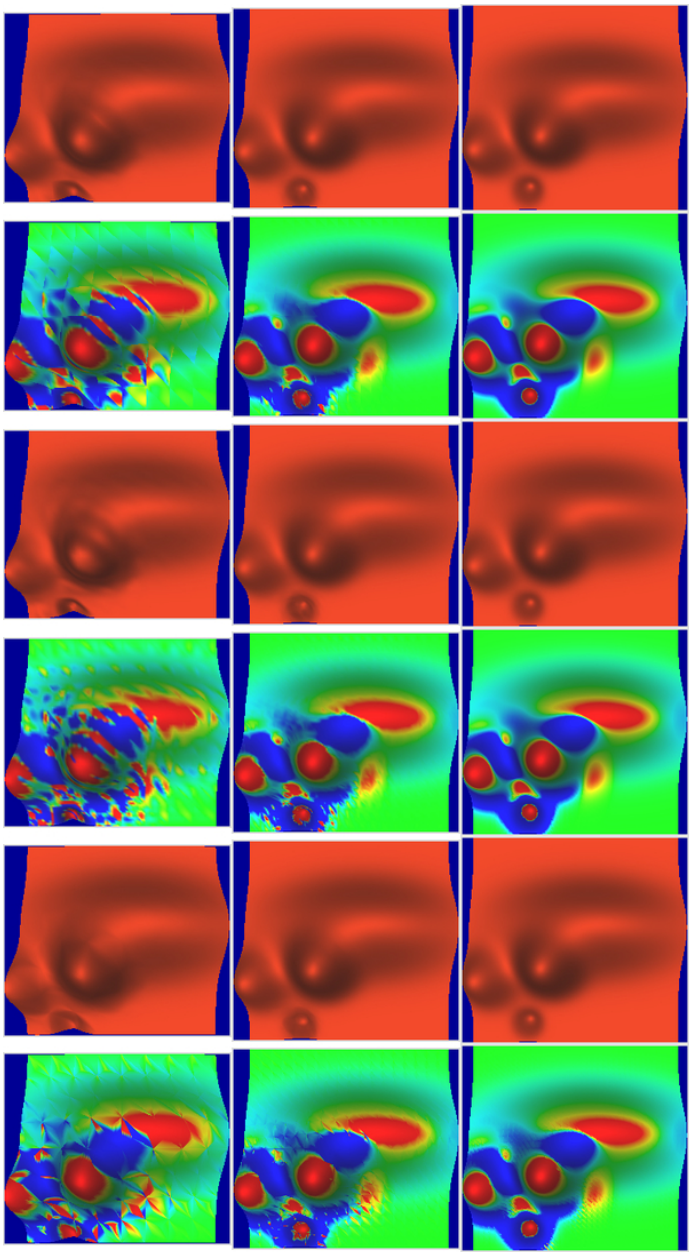

Figure 4.2: *Franke 1:* Comparison of the tangent fitting scheme using the Monomial basis at degree 5 to the Clough-Tocher scheme. Each pair of rows represents a scheme, first shaded, then coloured with Gaussian curvature to highlight discontinuities. The first rows are a Monomial degree 5 fitter feeding Mann's degree 5 method 1 continuity adjuster. The next are the same fitter with the degree 9 adjuster. The last two rows are the Clough-Tocher interpolation. All columns use the same sample density over $(-1,-1)-(1,1)$: 11×11, 21×21 and 41×41

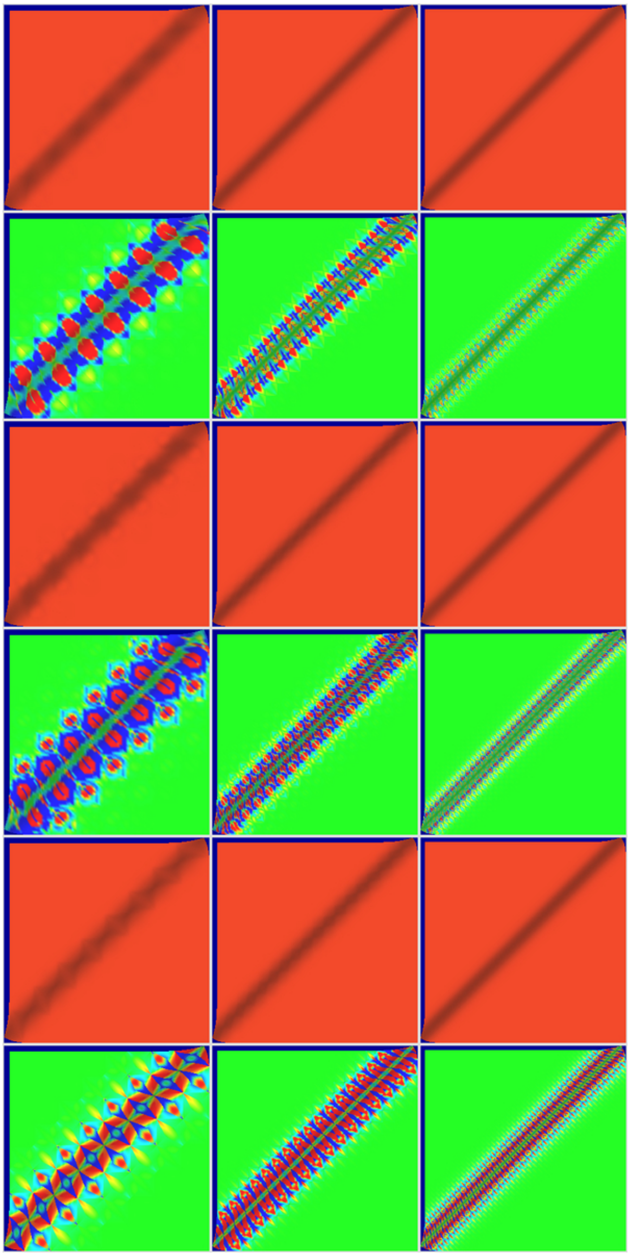

Figure 4.3: *Franke 2:* Comparison of the tangent fitting scheme using the Monomial basis at degree 5 to the Clough-Tocher scheme. Each pair of rows represents a scheme, first shaded, then coloured with Gaussian curvature to highlight discontinuities. The first rows are a Monomial degree 5 fitter feeding Mann's degree 5 method 1 continuity adjuster. The next are the same fitter with the degree 9 adjuster. The last two rows are the Clough-Tocher interpolation. All columns use the same sample density over $(-1,-1)-(1,1)$: 11×11, 21×21 and 41×41

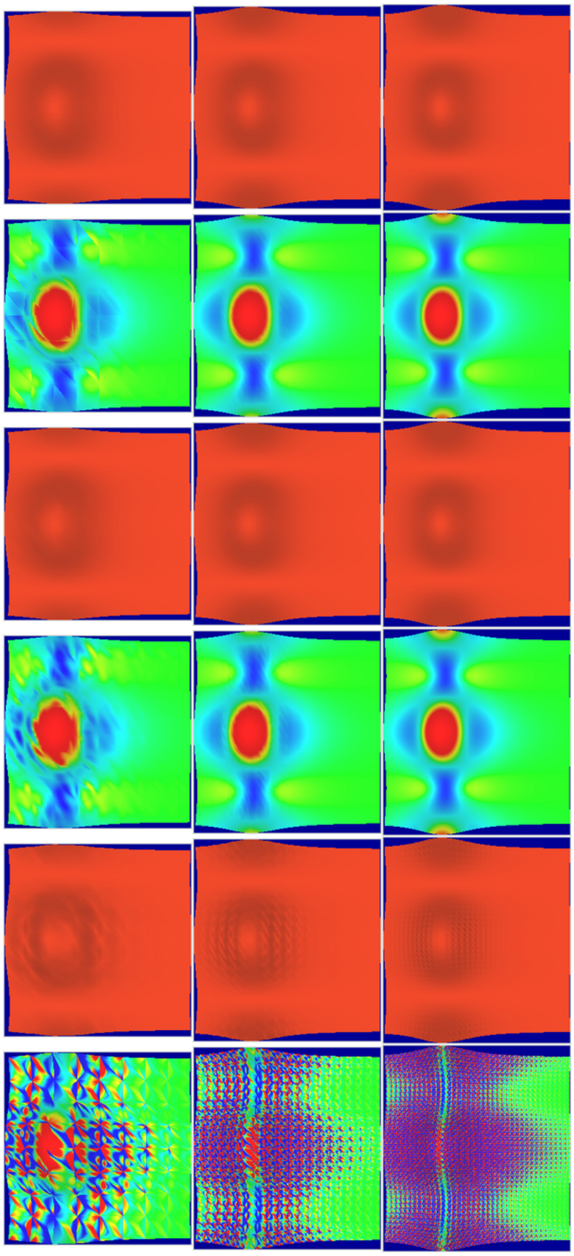

Figure 4.4: *Franke 3:* Comparison of the tangent fitting scheme using the Monomial basis at degree 5 to the Clough-Tocher scheme. Each pair of rows represents a scheme, first shaded, then coloured with Gaussian curvature to highlight discontinuities. The first rows are a Monomial degree 5 fitter feeding Mann's degree 5 method 1 continuity adjuster. The next are the same fitter with the degree 9 adjuster. The last two rows are the Clough-Tocher interpolation. All columns use the same sample density over $(-1,-1)-(1,1)$: 11×11, 21×21 and 41×41

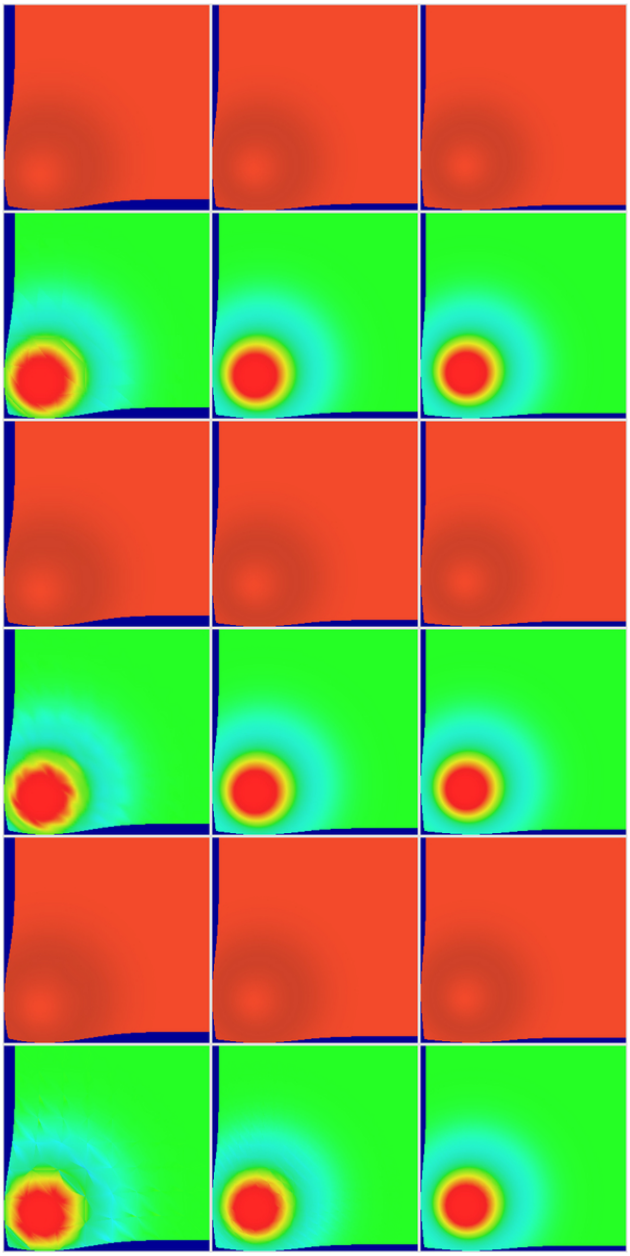

Figure 4.5: *Franke 4:* Comparison of the tangent fitting scheme using the Monomial basis at degree 5 to the Clough-Tocher scheme. Each pair of rows represents a scheme, first shaded, then coloured with Gaussian curvature to highlight discontinuities. The first rows are a Monomial degree 5 fitter feeding Mann's degree 5 method 1 continuity adjuster. The next are the same fitter with the degree 9 adjuster. The last two rows are the Clough-Tocher interpolation. All columns use the same sample density over $(-1,-1)-(1,1)$: 11×11, 21×21 and 41×41

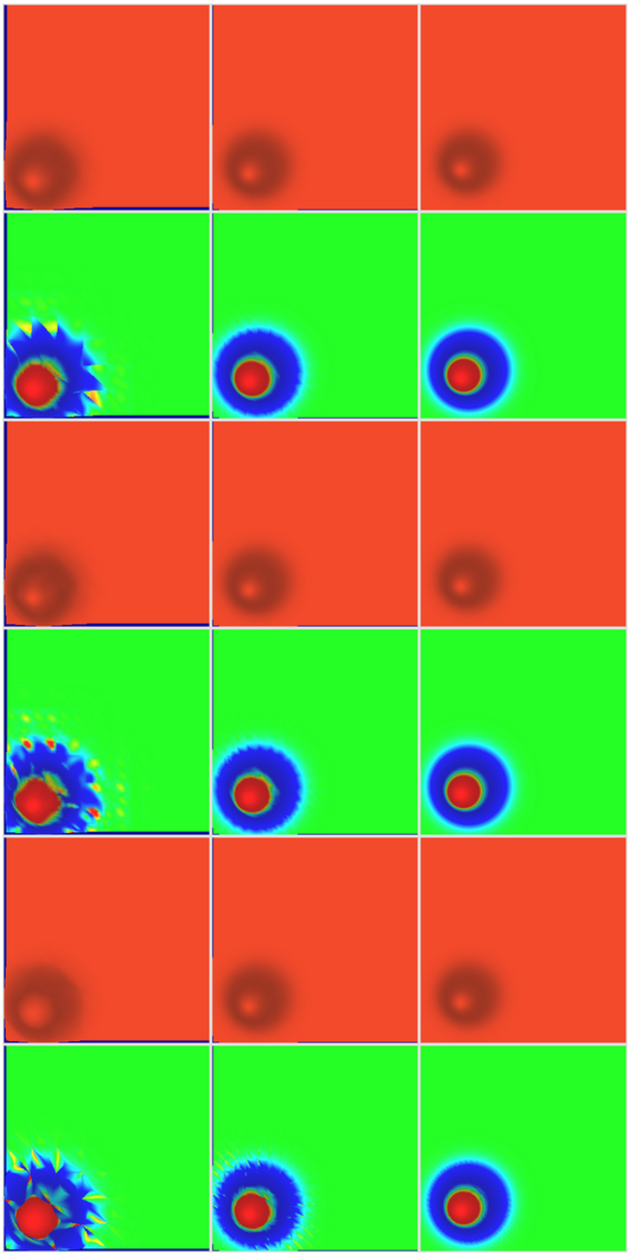

Figure 4.6: *Franke 5:* Comparison of the tangent fitting scheme using the Monomial basis at degree 5 to the Clough-Tocher scheme. Each pair of rows represents a scheme, first shaded, then coloured with Gaussian curvature to highlight discontinuities. The first rows are a Monomial degree 5 fitter feeding Mann's degree 5 method 1 continuity adjuster. The next are the same fitter with the degree 9 adjuster. The last two rows are the Clough-Tocher interpolation. All columns use the same sample density over $(-1,-1)-(1,1)$: 11×11, 21×21 and 41×41

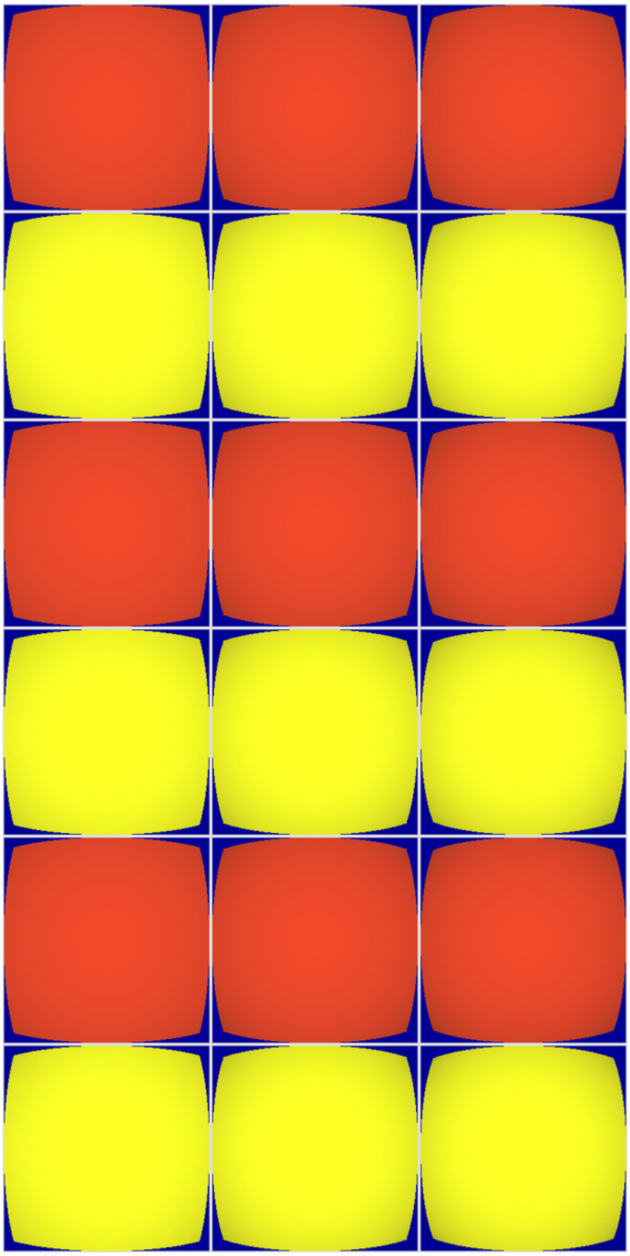

Figure 4.7: *Franke 6:* Comparison of the tangent fitting scheme using the Monomial basis at degree 5 to the Clough-Tocher scheme. Each pair of rows represents a scheme, first shaded, then coloured with Gaussian curvature to highlight discontinuities. The first rows are a Monomial degree 5 fitter feeding Mann's degree 5 method 1 continuity adjuster. The next are the same fitter with the degree 9 adjuster. The last two rows are the Clough-Tocher interpolation. All columns use the same sample density over $(-1,-1)-(1,1)$: 11×11, 21×21 and 41×41

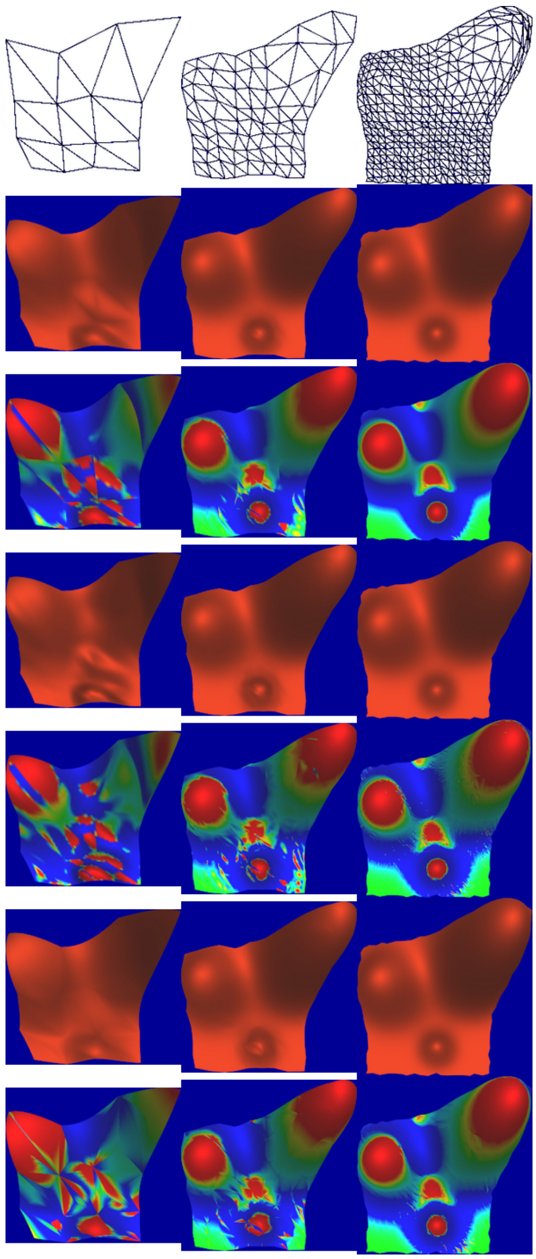

Figure 4.8: *Franke 1:* Randomized mesh comparison of the tangent fitting scheme using the Monomial basis at degree 5 to the Clough-Tocher scheme. The top row shows an overhead view of the randomized mesh. Each subsequent pair of rows represents a scheme, first shaded, then coloured with Gaussian curvature to highlight discontinuities. The first rows are a Monomial degree 5 fitter feeding Mann's degree 5 method 1 continuity adjuster. The next are the same fitter with the degree 9 adjuster. The last two rows are the Clough-Tocher interpolation. All columns use the same sample density over $(0,0)-(1,1)$: 11×11, 21×21 and 41×41

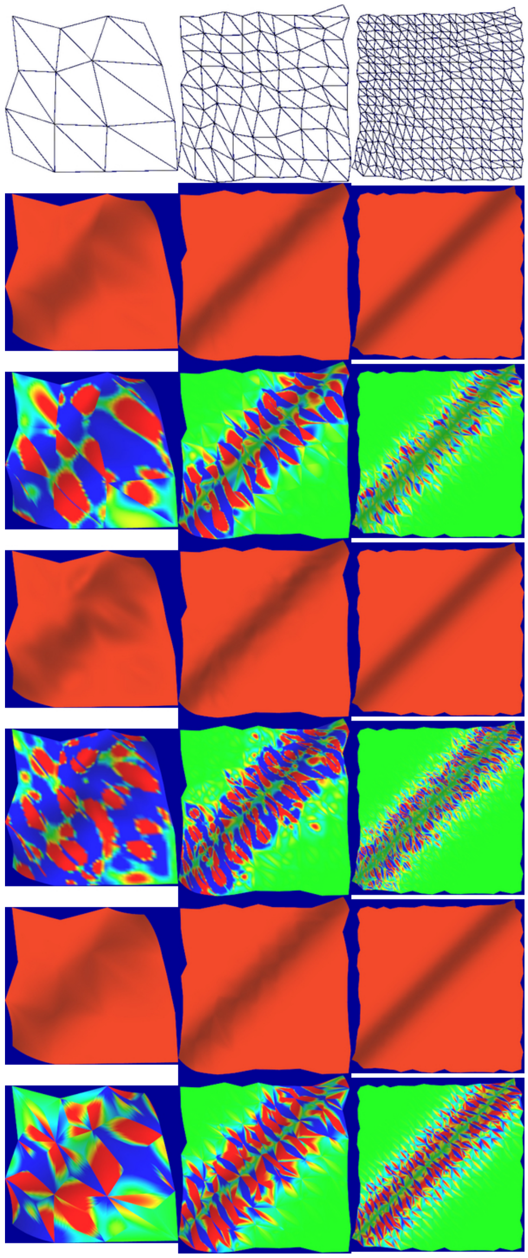

Figure 4.9: *Franke 2:* Randomized mesh comparison of the tangent fitting scheme using the Monomial basis at degree 5 to the Clough-Tocher scheme. The top row shows an overhead view of the randomized mesh. Each subsequent pair of rows represents a scheme, first shaded, then coloured with Gaussian curvature to highlight discontinuities. The first rows are a Monomial degree 5 fitter feeding Mann's degree 5 method 1 continuity adjuster. The next are the same fitter with the degree 9 adjuster. The last two rows are the Clough-Tocher interpolation. All columns use the same sample density over $(0,0)-(1,1)$: 11×11, 21×21 and 41×41

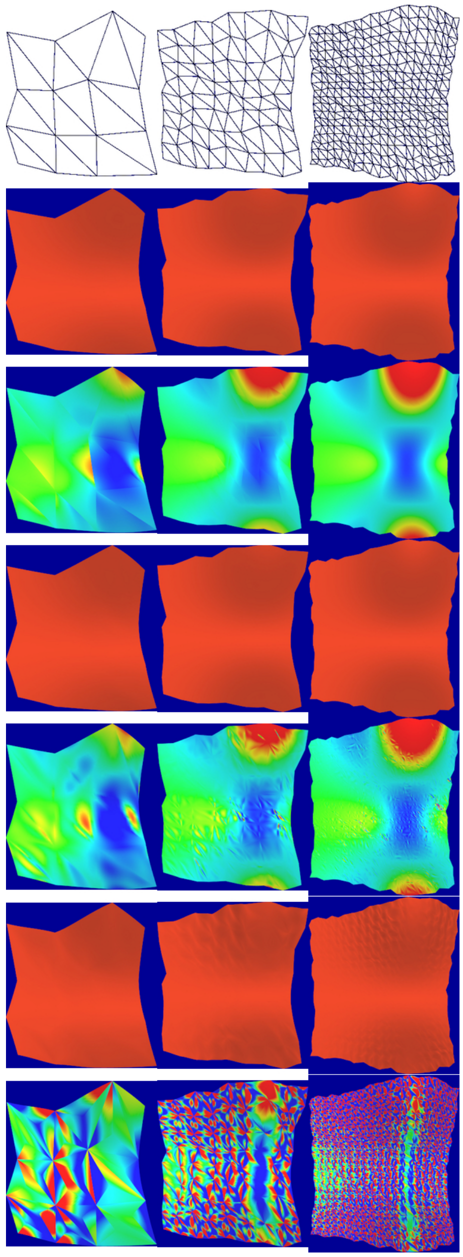

Figure 4.10: *Franke 3:* Randomized mesh comparison of the tangent fitting scheme using the Monomial basis at degree 5 to the Clough-Tocher scheme. The top row shows an overhead view of the randomized mesh. Each subsequent pair of rows represents a scheme, first shaded, then coloured with Gaussian curvature to highlight discontinuities. The first rows are a Monomial degree 5 fitter feeding Mann's degree 5 method 1 continuity adjuster. The next are the same fitter with the degree 9 adjuster. The last two rows are the Clough-Tocher interpolation. All columns use the same sample density over $(0,0)-(1,1)$: 11×11, 21×21 and 41×41

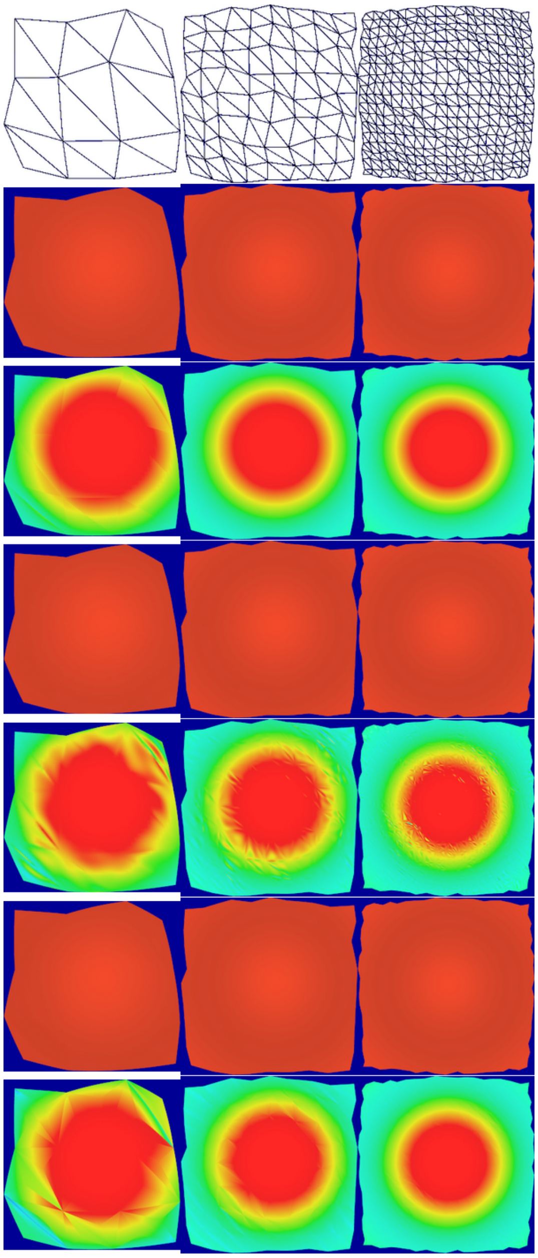

Figure 4.11: *Franke 4:* Randomized mesh comparison of the tangent fitting scheme using the Monomial basis at degree 5 to the Clough-Tocher scheme. The top row shows an overhead view of the randomized mesh. Each subsequent pair of rows represents a scheme, first shaded, then coloured with Gaussian curvature to highlight discontinuities. The first rows are a Monomial degree 5 fitter feeding Mann's degree 5 method 1 continuity adjuster. The next are the same fitter with the degree 9 adjuster. The last two rows are the Clough-Tocher interpolation. All columns use the same sample density over $(0,0)-(1,1)$: 11×11, 21×21 and 41×41

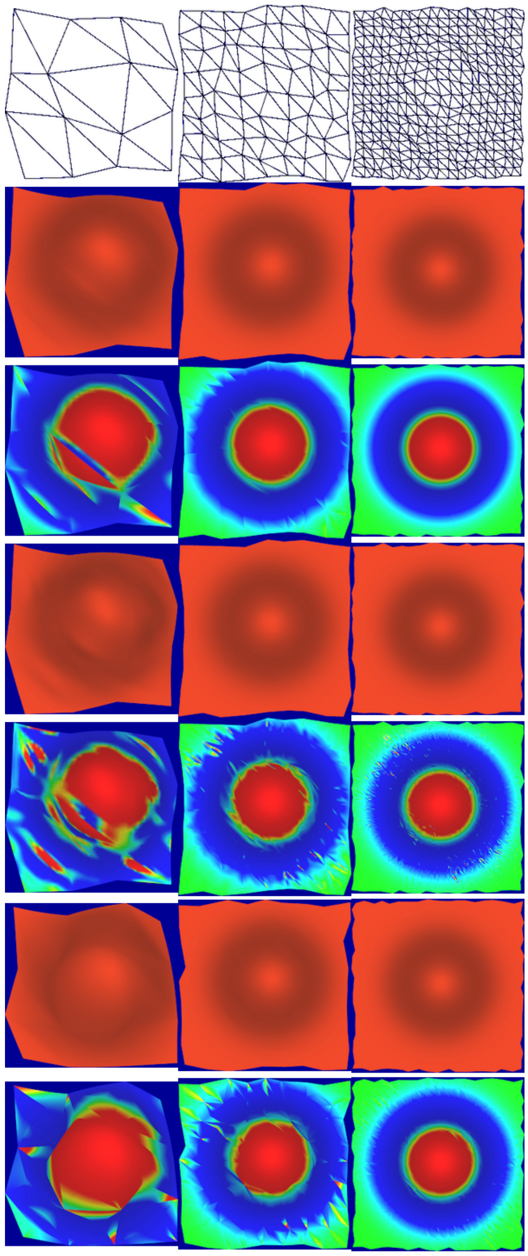

Figure 4.12: *Franke 5:* Randomized mesh comparison of the tangent fitting scheme using the Monomial basis at degree 5 to the Clough-Tocher scheme. The top row shows an overhead view of the randomized mesh. Each subsequent pair of rows represents a scheme, first shaded, then coloured with Gaussian curvature to highlight discontinuities. The first rows are a Monomial degree 5 fitter feeding Mann's degree 5 method 1 continuity adjuster. The next are the same fitter with the degree 9 adjuster. The last two rows are the Clough-Tocher interpolation. All columns use the same sample density over $(0,0)-(1,1)$: 11×11, 21×21 and 41×41

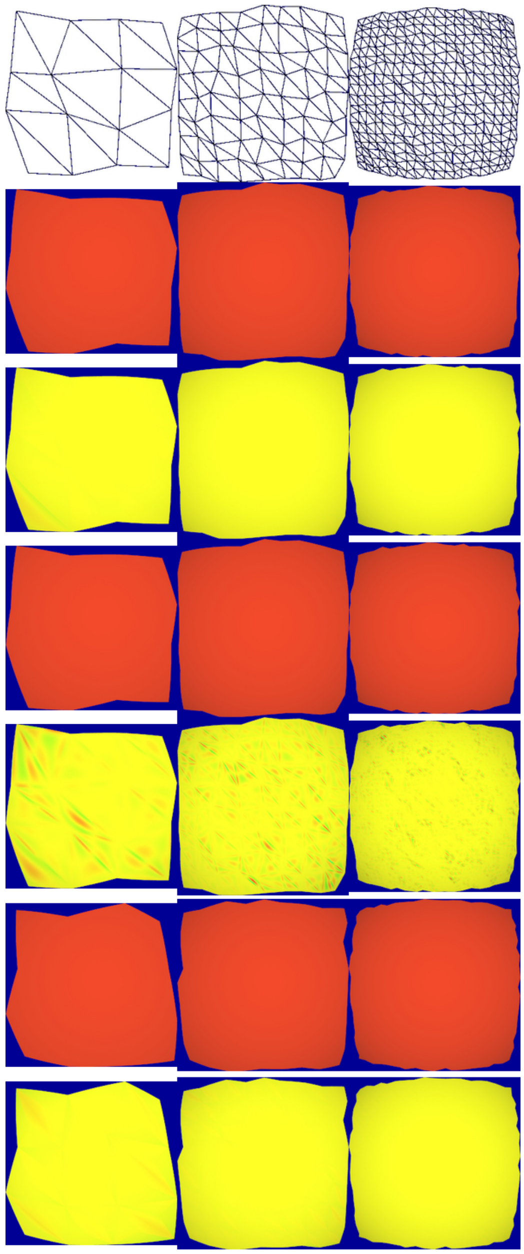

Figure 4.13: *Franke 6:* Randomized mesh comparison of the tangent fitting scheme using the Monomial basis at degree 5 to the Clough-Tocher scheme. The top row shows an overhead view of the randomized mesh. Each subsequent pair of rows represents a scheme, first shaded, then coloured with Gaussian curvature to highlight discontinuities. The first rows are a Monomial degree 5 fitter feeding Mann's degree 5 method 1 continuity adjuster. The next are the same fitter with the degree 9 adjuster. The last two rows are the Clough-Tocher interpolation. All columns use the same sample density over $(0,0)-(1,1)$: 11×11, 21×21 and 41×41

Chapter 5

Conclusions

The derivative estimation technique works.

It reproduces polynomials accurately.

When the derivatives are fed into the patch setter/continuity adjuster the resulting surfaces show polynomial precision for all schemes except Bernstein Point only schemes at degree 9(tables 4.9 to 4.16). The problems seem to occur because of the range remapping step used to get best stability for the bernstein basis. Degree 9 fitting should never be called for as degree 5 fitting provides aduquate results when used in degree 9 patches.

The tangent based scheme shows better precision than Clough-Tocher in all tests except degree 3 surface fitting tests for both degree 5 and 9 interpolating surfaces. For degree 3 surface fitting the scheme shows similar convergence to Clough-Tocher. See tables 4.17 to 4.22.

The same scheme also showed visually similar or better results on shaded and gaussian curvature coloured renderings at three sample densities. See figures 4.2 to 4.13.

The C^2 continuity adjustments show problems when applied to randomly scattered data for certain surfaces. See figures 4.10 and 4.13 middle pair of rows, note that the shaded surface shows no defect. Defects are difficult to see even when an interactive model is examined, but they are faintly visible at some angles. The second continuity adjustment method shows critical problems (fig. 5.1). However the estimator is able to provide accurate enough derivatives to

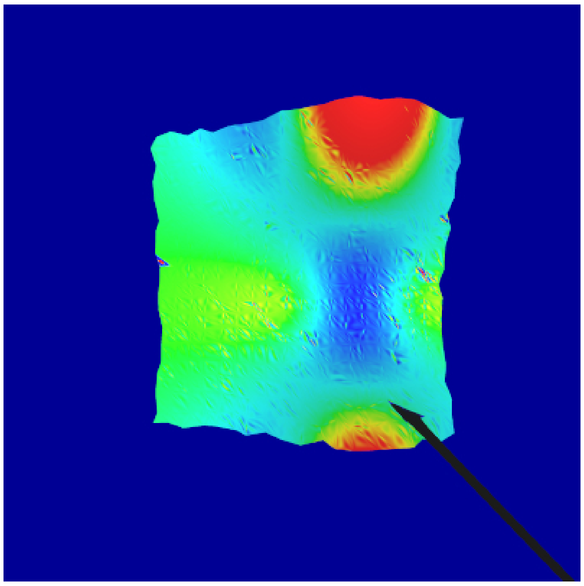

Figure 5.1: *Franke 3:* rendered at 19 samples per patch from a jittered 21×21 mesh over $(0,0)-(1,1)$ with degree 5 monomial based point and tangent fitting and degree 9 C^2 adjustments using the 2^{nd} continuity adjusting method.

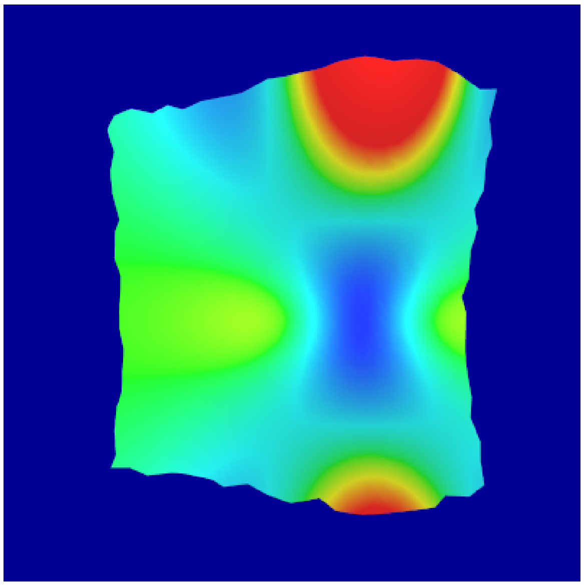

Figure 5.2: *Franke 3:* rendered with the same fitting and point information as 5.1 and degree 9 C^0 adjustments.

produce smooth looking gaussian curvature shaded images when sampling is fine enough and no continuity adjustments are made (fig. 5.2).

Further work is needed to make the scheme practical. A mesh reader should be added to my estimator. The problem shown in 5.1 should be looked into to see if C^2 polynomial precision interpolants can be achieved for more general triangulations.

Bibliography

[1] Ron Goldman. *Pyramid Algorithms.* Morgan Kaufmann Publishers, 2003.

[2] Eric W. Weisstein. Taylor series. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/TaylorSeries.html, March 2006.

[3] Stephen Mann. Continuity adjustments to triangular bezier patches that retain polynomial precision. Technical Report CS-2000-01, Computer Science Departement, University of Waterloo, January 2000.

[4] Stephen Mann. Implementation of some triangular data fitting schemes using averaging to get continuity. Technical Report CS-2000-10, Computer Science Departement, University of Waterloo, April 2000.

[5] Sastry S. Isukapalli. *Uncertainty Analysis of Transport-Transformation Models.* PhD thesis, Rutgers, The State University of New Jersey, 1999. Section 4.1 has a very brief survey of derivative estimation techniques.

[6] *Applied Numerical Analysis Using Matlab*, pages 369–372. Prentice hall, first edition, 1999.

[7] W.F. Ames. *Numerical Methods for Partial Differential Equations.* Academic Press, Boston, third edition, 1992.

[8] Dunno. *Linear Algebra Something.* yep, ? edition, ????

[9] Hiroshi Akima. Algorithm 761; scattered-data surface fitting that has the accuracy of a cubic polynomial. *ACM Trans. Math. Softw.*, 22(3):362–371, 1996.

[10] R. Franke. A critical comparison of some methods for intermpolation of scattered data. Technical Report NPS-53-79-003, Dept. of Mathematics, Naval Postgraduate School, Monterey, Calif., 1979. I've never seen it. Discussed extensively in Alg761.

[11] Robert J. Renka and Ron Brown. Remark on algorithm 761. *ACM Trans. Math. Softw.*, 24(4):383–385, 1998.

[12] T. N. T. Goodman R. T. Farouki. On the optimal stability of the bernstein basis. *Mathematics of Computation*, 65(216):1553–1566, October 1996.

[13] Jim Ramsay. Smoothing and nonparametric regression. Research interest on Dr. Ramsay's homepage:http://www.psych.mcgill.ca/faculty/ramsay/ramsay.html. Waiting for him to get back to me with background or a proper reference.