AIMS Mathematics, 6(4): 3142–3159.
DOI: 10.3934/math.2021190
Received: 01 October 2020
Accepted: 29 December 2020
Published: 14 January 2021

http://www.aimspress.com/journal/Math

Research article

Non-linear least squares fitting of Bézier surfaces to unstructured point clouds

Joseph Lifton[1,*], Tong Liu[2] and John McBride[3]

[1] Intelligent Product Verification Group, Advanced Remanufacturing and Technology Centre, 637143, Singapore
[2] Precision Measurements Group, Singapore Institute of Manufacturing Technology, 637662, Singapore
[3] Mechatronics Engineering Group, Mechanical Engineering Department, Faculty of Engineering and Physical Sciences, University of Southampton, SO17 1BJ, UK

*** Correspondence:** Email: Lifton_Joseph_John@artc.a-star.edu.sg.

Abstract: Algorithms for linear and non-linear least squares fitting of Bézier surfaces to unstructured point clouds are derived from first principles. The presented derivation includes the analytical form of the partial derivatives that are required for minimising the objective functions, these have been computed numerically in previous work concerning Bézier curve fitting, not surface fitting. Results of fitting fourth degree Bézier surfaces to complex simulated and measured surfaces are presented, a quantitative comparison is made between fitting Bézier surfaces and fitting polynomial surfaces. The developed fitting algorithm is used to remove the geometric form of a complex engineered surface such that the surface roughness can be evaluated.

Keywords: Bézier surfaces; least squares fitting; surface metrology
Mathematics Subject Classification: 65D10

1. Introduction

Bézier curves and surfaces are widely used in computer graphics and computer aided design for representing complex geometries as a set of smooth analytically driven curves. For example, they have been used for aerofoil geometry optimisation [1] for representing the outline of textual

characters [2] and are used extensively for designing and reconstructing complex free form objects in computer aided geometric design [3]. A Bézier curve is a parametric curve based on Bernstein polynomials. The shape of the curve is modified by a set of control points, consecutive control points are termed a control polygon, note that the resulting curve does not necessarily pass through these control points. The number of control points defines the possible complexity of the curve, for example, with two control points only a straight line can be evaluated, this being a first degree Bézier curve, with three control points (a second degree Bézier curve) quadratic curvature is possible, and so on. Some example fourth degree (5 by 5 control points) Bézier surfaces are shown in Figure 1 alongside the control points.

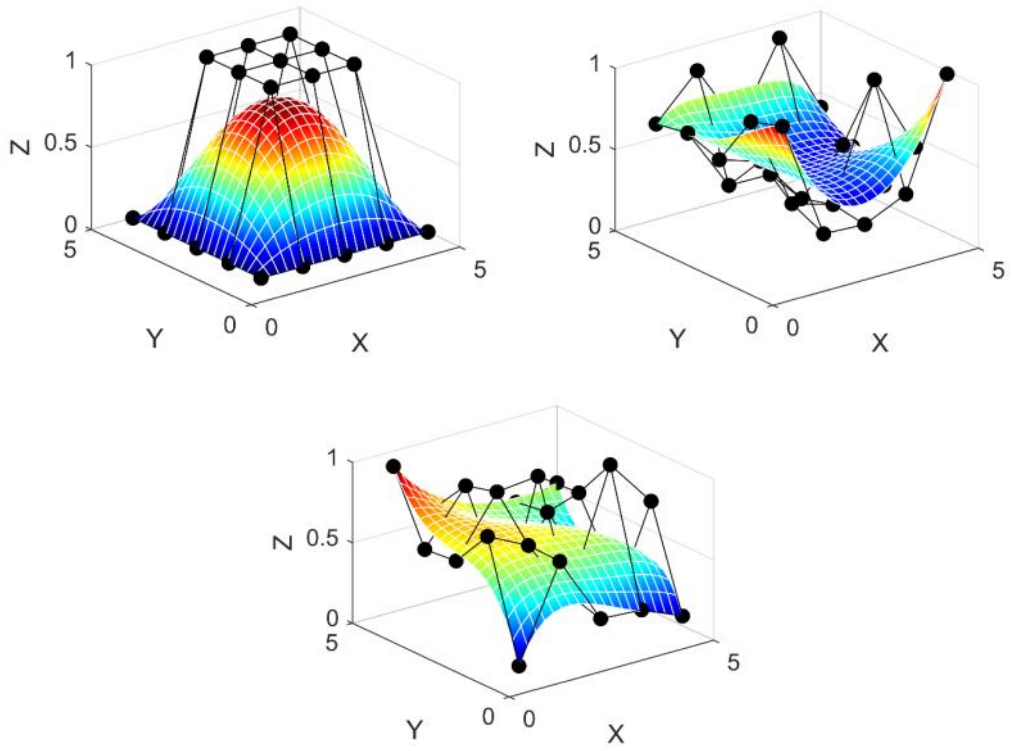


Figure 1. Exemplar Bézier surfaces and the respective control points.

In this work we are interested in fitting Bézier surfaces to a set of noisy, unstructured, scattered points, the particular application we are interested in is the removal of geometric form in order to characterise the surface texture of highly complex engineering surfaces [4]. It is straightforward to fit a polynomial surface to a set of measured points using the least squares method, many commercial software packages are available to do this fitting task, however Bézier surfaces offer the desirable property of passing through the control points at the corners of its control polygon, this means that Bézier patches can be stitched together to form a patchwork which can be used to approximate geometries of even greater complexity, such as the surface geometries that can now be fabricated using additive manufacturing.

Having reviewed the literature, the linear least squares (LLS) solution for fitting Bézier surfaces is well documented, however, we are unable to find the non-linear least squares (NLLS) Bézier surface fitting algorithm that is presented in this work. Fitting Bézier curves (not surfaces) via LLS

and NLLS is considered in references [5] and [6] and a NLLS spline curve fitting algorithm is presented in [7]. In references [8–10] the LLS Bézier surface fitting algorithm is given, but iterative refinement is achieved using a genetic algorithm, the fireworks algorithm and the firefly algorithm, respectively, rather than NLLS, which makes the fitting task more complicated than it need be for many applications such as form removal for surface texture characterisation. The LLS fitting algorithm for Bézier surfaces is given in references [11] and [12] but the authors do not present the NLLS fitting algorithm. In [13] free form surface fitting is discussed from a reverse engineering perspective but neither the LLS or NLLS Bézier surface fitting algorithms are presented. We therefore adopt the general LLS and NLLS approach presented in references [5] and [6] but extend to the case of a fourth degree Bézier surface.

Fourth degree Bézier surfaces are considered here to allow for the fitting of highly complex surfaces such as the engineered surface considered at the end of the paper. Linear, quadratic and cubic surfaces are deemed too simple to represent the complex surfaces that are of interest in this work. It is worth noting that the presented algorithm can be modified to consider any degree of Bézier surface.

In this work the partial derivatives that are required for both the LLS and NLLS fitting are evaluated explicitly (see Appendix 2 and 3) as opposed to numerically as per the work in [5], this leads to a more computationally efficient implementation.

The structure or order of the data points to which we fit Bézier surfaces is not required apriori, the data points can be randomly scattered. Furthermore, no initial estimate of the control points is required, this is provided by the LLS algorithm. Therefore, the developed method is highly practical and should be of great use for a number of surface fitting tasks.

In order to validate the developed method and benchmark it, we compare the algorithm to the fitting of polynomial surfaces for 3 different surfaces, two simulated surfaces with superimposed noise and one measured surface. The algorithm is benchmarked against polynomials because these are the default fitting functions used by engineers, especially in the field of surface metrology [4]. The convergence of the iterative NLLS algorithm is also plotted for each of the considered surfaces to demonstrate the stability of the algorithm.

2. Method

The general formula for a Bézier surface of degree n, m is:

$$P(u_t, v_t) = \sum_{i=0}^{n} \sum_{j=0}^{m} \binom{n}{i} u_t^i (1-u_t)^{n-i} \binom{m}{j} v_t^j (1-v_t)^{m-j} \, k_{ij} \tag{1}$$

where u and v are parametric coordinates with $0 \leq u \leq 1$ and $0 \leq v \leq 1$. The term t is the index of the parametric coordinates. The terms k_{ij} are the control point coordinates in x, y, z and these are the variables that need to be estimated such that the Bézier surface $P(u_t, v_t)$ fits the measured data. We will consider a fourth degree Bézier surface where $m = n = 4$. The fourth degree Bézier surface is written explicitly as per Appendix 1.

Let the measured data we wish to fit the Bézier surface to be denoted d_t with $t = 1, \dots, N$. The least squares solution requires we minimise the sum of the squared errors. Therefore, define error e_t as:

$$e_t = P(u_t, v_t) - d(u_t, v_t). \tag{2}$$

Summing the squared errors gives:

$$M = \sum_{t=1}^{N} \big(P(u_t, v_t) - d(u_t, v_t)\big)^2 = \sum_{t=1}^{N} e_t^2. \tag{3}$$

We want to minimise M, therefore take partial derivatives with respect to k_{ij} and set these to zero:

$$\frac{\partial M}{\partial k_{i,j}} = \frac{\partial M}{\partial e_t}\frac{\partial e_t}{\partial k_{i,j}}, \qquad \frac{\partial M}{\partial e_t} = 2\sum_{t=1}^{N} e_t\,, \qquad \frac{\partial M}{\partial k_{i,j}} = 2\sum_{t=1}^{N} e_t \frac{\partial e_t}{\partial k_{i,j}}. \tag{4}$$

The partial derivatives $\partial e_t/\partial k_{i,j}$ are given in Appendix 2. Taking partial derivatives leads to 25 equations that must be solved simultaneously for the coefficients k_{ij}. These equations can be written in the form $Ax = b$ and solved in the usual way. First, consider the partial derivatives set equal to zero and then rearranged:

$$2\sum_{t=1}^{N} \big(P(u_t, v_t) - d(u_t, v_t)\big)\frac{\partial e_t}{\partial k_{i,j}} = 0 \tag{5}$$

$$\sum_{t=1}^{N} P(u_t, v_t)\frac{\partial e_t}{\partial k_{i,j}} = \sum_{t=1}^{N} d(u_t, v_t)\frac{\partial e_t}{\partial k_{i,j}}. \tag{6}$$

Let the coefficients of the k_{ij} terms in $P(u_t, v_t)$ be denoted $c1\ to\ c25$:

$$\begin{bmatrix} \sum_{t=1}^{N} c1\frac{\partial e_t}{\partial k_{0,0}} & \dots & \sum_{t=1}^{N} c25\frac{\partial e_t}{\partial k_{0,0}} \\ \vdots & \ddots & \vdots \\ \sum_{t=1}^{N} c1\frac{\partial e_t}{\partial k_{4,4}} & \dots & \sum_{t=1}^{N} c25\frac{\partial e_t}{\partial k_{4,4}} \end{bmatrix} \begin{bmatrix} k_{0,0}^x & k_{0,0}^y & k_{0,0}^z \\ \vdots & \vdots & \vdots \\ k_{4,4}^x & k_{4,4}^y & k_{4,4}^z \end{bmatrix}$$
$$= \begin{bmatrix} \sum_{t=1}^{N} d^x(u_t, v_t)\frac{\partial e_t}{\partial k_{0,0}} & \sum_{t=1}^{N} d^y(u_t, v_t)\frac{\partial e_t}{\partial k_{0,0}} & \sum_{t=1}^{N} d^z(u_t, v_t)\frac{\partial e_t}{\partial k_{0,0}} \\ \vdots & \vdots & \vdots \\ \sum_{t=1}^{N} d^x(u_t, v_t)\frac{\partial e_t}{\partial k_{4,4}} & \sum_{t=1}^{N} d^y(u_t, v_t)\frac{\partial e_t}{\partial k_{4,4}} & \sum_{t=1}^{N} d^z(u_t, v_t)\frac{\partial e_t}{\partial k_{4,4}} \end{bmatrix} \tag{7}$$

which takes the form $Ax = b$ where:

$$A = \begin{bmatrix} \sum_{t=1}^{N} c1 \frac{\partial e_t}{\partial k_{0,0}} & \dots & \sum_{t=1}^{N} c25 \frac{\partial e_t}{\partial k_{0,0}} \\ \vdots & \ddots & \vdots \\ \sum_{t=1}^{N} c1 \frac{\partial e_t}{\partial k_{4,4}} & \dots & \sum_{t=1}^{N} c25 \frac{\partial e_t}{\partial k_{4,4}} \end{bmatrix}$$

$$x = \begin{bmatrix} k_{0,0}^{x} & k_{0,0}^{y} & k_{0,0}^{z} \\ \vdots & \vdots & \vdots \\ k_{4,4}^{x} & k_{4,4}^{y} & k_{4,4}^{z} \end{bmatrix} \tag{8}$$

$$b = \begin{bmatrix} \sum_{t=1}^{N} d^{x}(u_t, v_t) \frac{\partial e_t}{\partial k_{0,0}} & \sum_{t=1}^{N} d^{y}(u_t, v_t) \frac{\partial e_t}{\partial k_{0,0}} & \sum_{t=1}^{N} d^{z}(u_t, v_t) \frac{\partial e_t}{\partial k_{0,0}} \\ \vdots & \vdots & \vdots \\ \sum_{t=1}^{N} d^{x}(u_t, v_t) \frac{\partial e_t}{\partial k_{4,4}} & \sum_{t=1}^{N} d^{y}(u_t, v_t) \frac{\partial e_t}{\partial k_{4,4}} & \sum_{t=1}^{N} d^{z}(u_t, v_t) \frac{\partial e_t}{\partial k_{4,4}} \end{bmatrix}.$$

We therefore solve for x, where $x = A^{-1}b$. This gives best fit control points $k_{i,j}$ for the initial nodal points u_t, v_t. Now we need to minimise e_t by finding the best fit nodal points u_t, v_t. Given that e_t is non-linear with respect to u_t, v_t we must use non-linear least squares, for this we will use the Newton Raphson method, which states the following:

$$0 = f^{'}(x_n)(x_{n+1} - x_n) + f(x_n) \tag{9}$$

where x_n is an initial guess of the root of $f(x)$ and x_{n+1} is an improved guess of the root. For our purpose we want to find the root of e_t with respect to u and v, we therefore write:

$$\begin{aligned} 0 &= \frac{de(u_{1,t}, v_{1,t})}{du_t}\left(u_{2,t} - u_{1,t}\right) + e(u_{1,t}, v_{1,t}) \\ 0 &= \frac{de(u_{1,t}, v_{1,t})}{dv_t}\left(v_{2,t} - v_{1,t}\right) + e(u_{1,t}, v_{1,t}) \end{aligned} \tag{10}$$

for u_t and v_t respectively. These equations require that the partial derivatives of e_t are evaluated with respect to u_t and v_t, these are given in Appendix 3. The partial derivatives are known as the Jacobian matrices, J, and are of size N by N and are filled with zeros apart from the diagonal. Rewriting $\partial e_t/\partial u_t$ and $\partial e_t/\partial v_t$ as J_u and J_v respectively and rearranging to solve for $u_{2,t}$ and $v_{2,t}$ gives:

$$\begin{aligned} u_{2,t} &= u_{1,t} - \alpha(J_u^T J_u)^{-1} J_u^T e(u_{1,t}, v_{1,t}) \\ v_{2,t} &= v_{1,t} - \alpha(J_v^T J_v)^{-1} J_v^T e(u_{1,t}, v_{1,t}) \end{aligned} \tag{11}$$

where α is a relaxation parameter, $\alpha = 0.5$ is used throughout this work and is chosen empirically. These new estimates of u_t and v_t are then used to recalculate the control points $k_{i,j}$ using the linear least squares solution previously derived. The above can be implement as an iterative algorithm as follows:

1. Evaluate the linear least squares solution (Eq 7) to calculate the control points $k_{i,j}^{x}, k_{i,j}^{y}, k_{i,j}^{z}$, use an initial estimate of u_t, v_t which can be uniformly spaced values from 0 to 1.
2. Evaluate the non-linear least squares solution (Eq 11) to calculate a better estimate of u_t, v_t.
3. Use the linear least squares solution to calculate new control points $k_{i,j}^{x}, k_{i,j}^{y}, k_{i,j}^{z}$ using the updated estimate of u_t, v_t.
4. Repeat 2 and 3 until a convergence criterion is met. In this work convergence is deemed to have been met when the percentage difference between the sum of the squared residual (Eq 3) of two successive iterations is less than or equal to 0.5%.

3. Results

To test the developed algorithm, Bézier surfaces are fitted to unstructured scattered data sets that have been generated by randomly sampling two analytical surfaces, these surfaces are then superimposed with uniformly distributed noise. The analytical surfaces are:

$$z = \frac{y \sin x}{5} - \frac{x \cos y}{5} + \varepsilon(x, y) \tag{12}$$

$$z = \sin x + \cos y + \varepsilon(x, y) \tag{13}$$

where the x, y coordinates are generated by randomly generating uniformly distributed values between -5 and 5. The term $\varepsilon(x, y)$ is a noise signal with uniformly distributed values that lie between -0.1 and 0.1. Both data sets are generated to have 5000 x, y, z coordinates.

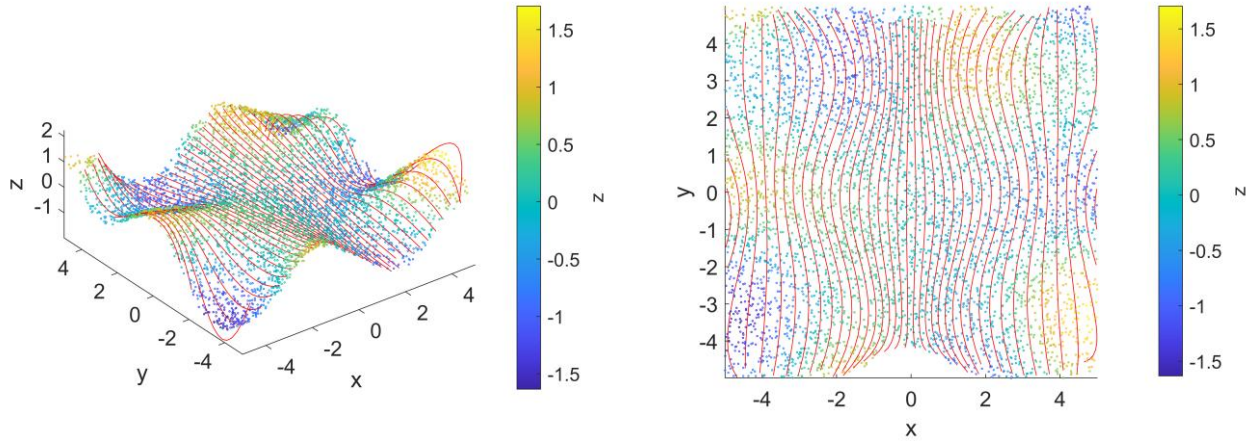
To benchmark the Bézier surface fitting algorithm, polynomial surfaces of 5th to 10th degree are fitted to the same scattered data points. The goodness of fit for both the Bézier and polynomial surfaces is calculated as the sum of the squared difference (error) between the z coordinates of the scattered data and the fitted surfaces for all x, y coordinates.

Figure 2A shows the unstructured scattered data points generated using Eq 12, these are represented as coloured dots, and the fitted Bézier surface is represented as a set of continuous red lines. The lines of the Bézier surface have been generated using uniformly spaced values of u_t, v_t, these are downsampled to show the curvature of the fitted surface more clearly. Figure 2B shows the same data but viewed from the top (z) direction, the curvature of the lines illustrates how the values of u_t, v_t have been modified by the NLLS algorithm in order for the surface to fit the scattered data, this modification would not occur with the LLS algorithm alone.

Figure 3A shows the convergence of the NLLS fitting algorithm, the convergence is smooth and without oscillation. Also plotted in Figure 3A is a constant value that corresponds to the sum of the squared noise; this is the squared sum of $\varepsilon(x, y)$, the noise residual that would remain after subtracting the perfect analytical form in Eq 12. We would not expect the sum of the squared error of the surface fit to fall below this value, if it did, it would indicate overfitting. Figure 3B shows the sum of the squared error for fitting polynomials of varying degree, the plot shows that the 4th order (25 control points) Bézier surface achieves a fit similar in error to that of a 7th or 8th degree polynomial (36 and 45 fitting coefficients respectively). Also plotted in Figure 3B is the squared sum of $\varepsilon(x, y)$, notice that for polynomials of degree 9 and 10 the fit residual falls below this value, which suggests the higher order polynomials are overfitting.

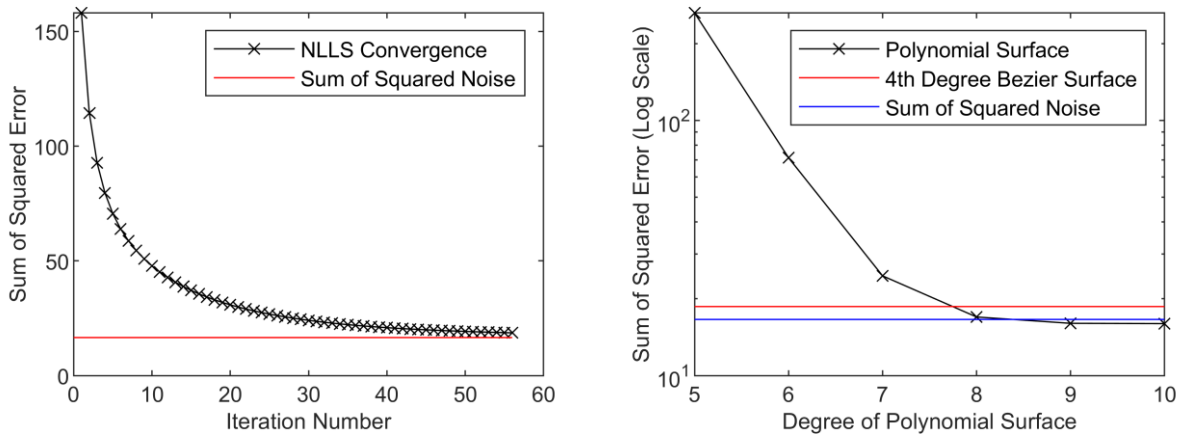
The results for fitting a Bézier surface to Eq 13 are shown in Figure 4A. As before the unstructured scattered data points are represented as coloured dots, and the fitted Bézier surface is represented as a set of continuous red lines. Figure 4B shows the same data but viewed from the top (z) direction, the curvature of the lines illustrates how the values of u_t, v_t have been modified by the NLLS algorithm in order for the surface to fit the scattered data.

Figure 5A shows the convergence of the NLLS fitting algorithm when fitting to data sampled from Eq 13, as before the convergence is smooth and without oscillation and the sum of the squared error does not fall below the sum of the squared noise. Fewer iterations are required for the algorithm to convergence when fitting to Eq 13 compared to fitting to Eq 12 (see Figure 3A). Figure 5B shows that the 4th order (25 control points) Bézier surface achieves a fit similar in error to that of a 6th or 7th degree polynomial (28 and 36 fitting coefficients respectively). The sum of the squared error for polynomials of 8th degree and higher converge to the sum of the squared noise, so little to no overfitting is observed.



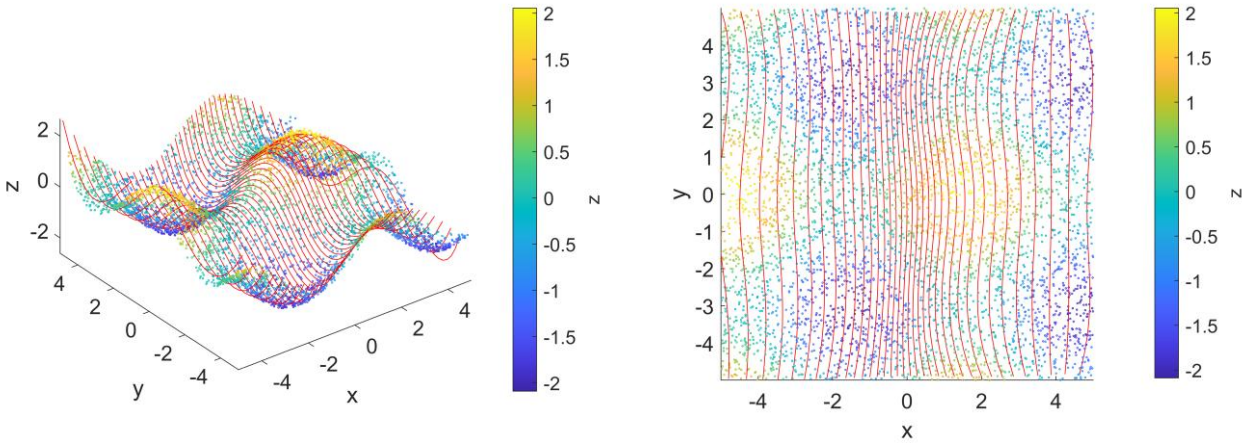
(A) (B)

Figure 2. (A) Unstructured point cloud sampled from Eq 12 (coloured dots) and fitted Bézier surface (red lines). (B) Top view (z) of (A).



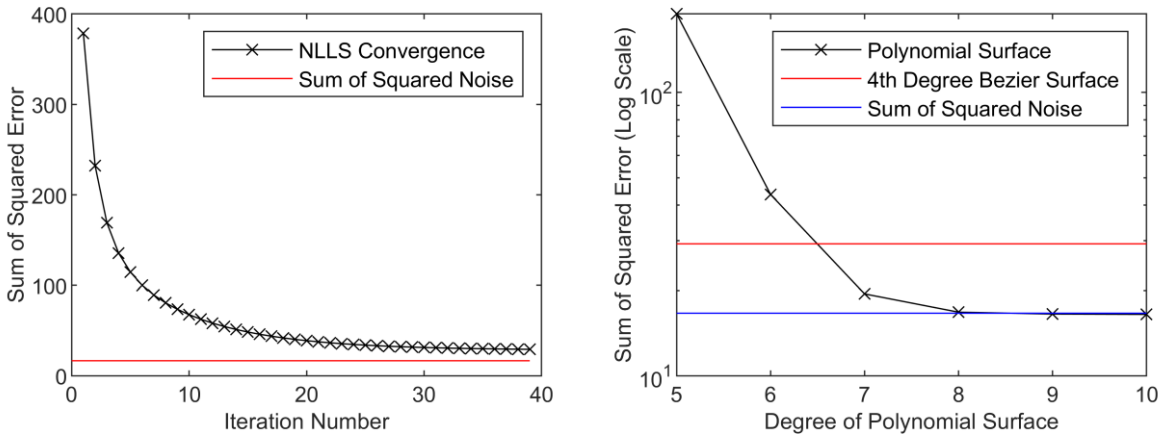
(A) (B)

Figure 3. (A) convergence graph of the NLLS Bézier fitting algorithm. (B) Comparison of fitting error for polynomial surfaces and the Bézier surface.



(A) (B)

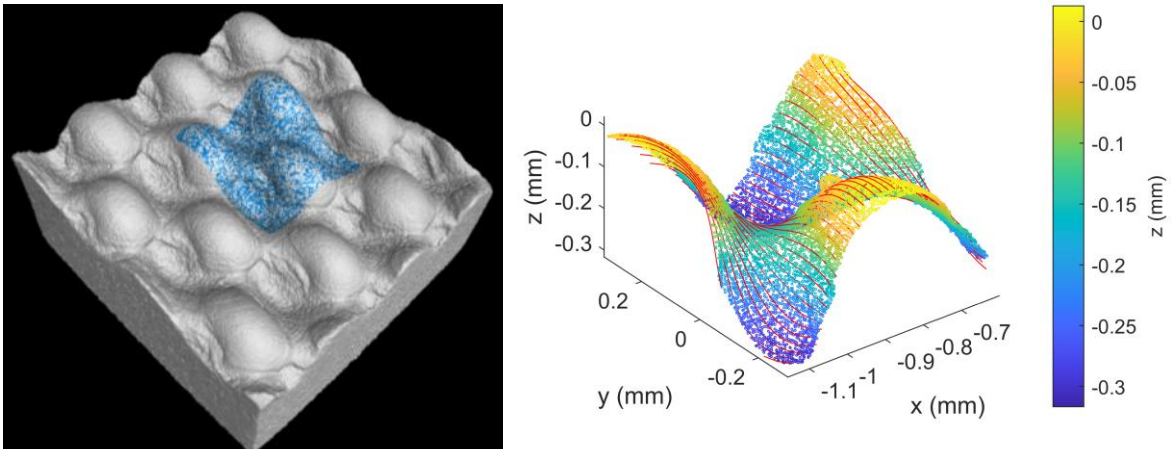
Figure 4. (A) Unstructured point cloud sampled from Eq 13 (coloured dots) and fitted Bézier surface (red lines). (B) Top view (z) of (A).



(A) (B)

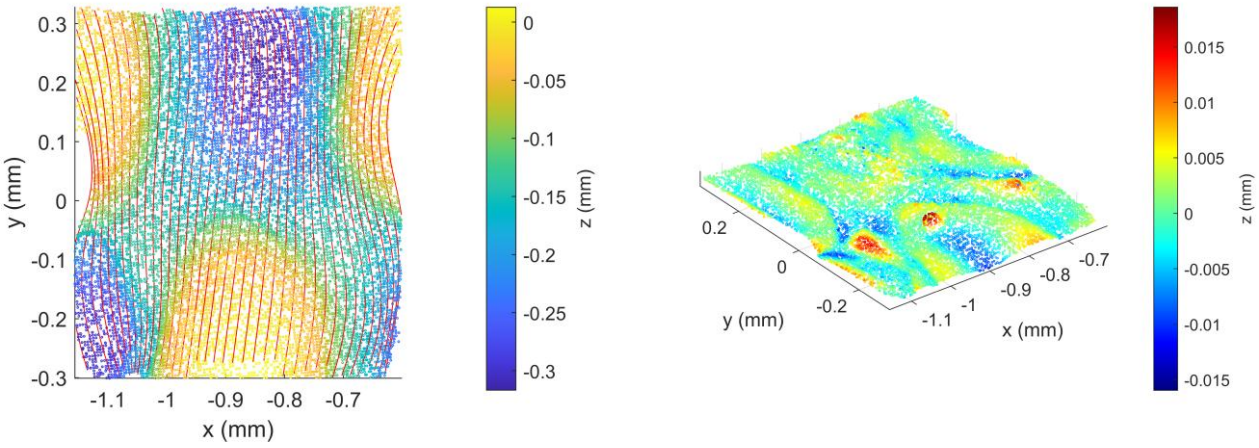
Figure 5. (A) convergence graph of the NLLS Bézier fitting algorithm. (B) Comparison of fitting error for polynomial surfaces and the Bézier surface.

To test the proposed method with real measured data we consider the problem of fitting and subtracting the geometric form of an engineering surface in order to isolate the surface texture for subsequent surface texture analysis. A complex metallic surface is measured via X-ray computed tomography (XCT), Figure 6A shows a volume rendering of the surface. The surface is extracted from the XCT data as an unstructured point cloud with 14478 $x, y\, z$ coordinates (Figure 6B) and we fit a Bézier surface using the developed algorithm. The algorithm converges after 75 iterations. The fitted surface is shown in Figure 7A as a set of continuous red lines that are down sampled for clarity. Subtracting the fitted surface from the measured surface yields the waviness and roughness of the surface that can be analysed in the usual way [4], see Figure 7B.



(A) (B)

Figure 6. (A) Volume rendering of an XCT scan of a complex metal surface. (B) Extracted point cloud (coloured dots) and Bézier surface (red lines) that has been fitted to the point cloud.



(A) (B)

Figure 7. (A) Top view (z) of Figure 6B. (B) Surface residual, the fitted surface has been subtracted from the measured points.

4. Discussion and conclusions

From the examples considered, it can be seen that the NLLS Bézier surface fitting algorithm is able to approximate surfaces of a high degree of complexity and is comparable to a 6th to 8th degree polynomial. The main advantage of using a Bézier surface rather than a polynomial is the ability to form piecewise Bézier patchworks to approximate surfaces of even higher complexity.

Bézier surfaces have the highly desirable property of passing through the control points at the corners of the control polygon, see Figure 1. Adjacent Bézier patches can be joined by equating the control points at the joining edges, however, to ensure a smooth transition from one patch to another requires equating the tangents of the joining control points, this is no trivial task. Fitting a patchwork of Bézier surfaces is complicated by the fact that control points at the edge of a patch cannot be changed without influencing the rest of the fitted surface, hence it is desirable to consider the stitching and fitting steps simultaneously. To address this problem Cao et al. [15] proposed a method to automatically subdivide a surface into a patchwork, whilst Lin et al. [16] proposed an algorithm for adaptively fitting a Bézier patchwork and ensuring first order continuity. Addressing this challenge is beyond the scope of the present work, but will be considered in future work. A comprehensive review of constrained fitting methods can be found in reference [17].

In the field of dimensional and surface metrology LLS and NLLS fitting algorithms are well known and accepted for their robustness [14]. Although more exotic iterative Bézier surface fitting algorithms have been developed, their complexity is not required for the type of surface fitting considered here, the robustness of the fitting algorithm is of more importance, this being the motivation for developing the algorithm presented here. Furthermore, many of the methods presented in the literature require the coordinate points to be ordered in order to fit a Bézier surface. Our algorithm requires no such apriori information, only the x, y, z coordinates of an unstructured point cloud are required as inputs, such as those generated by optical scanners and X-ray computed tomography.

To avoid overfitting or underfitting, the degree of the Bézier surface should be selected to match the complexity of the measured surface. This work has focused on an engineering surface with two inflection points (a peak and a trough), so at least 4 by 4 control points are required to approximate the surface. It was decided that the number of control points of the Bézier surface should be increased to 5 by 5 in order to better approximate the complexity of the measured surface, thus judgement was exercised in the choice of the degree of the Bézier surface. A more formal approach to selecting the degree of the Bézier surface is to conduct a convergence study, whereby the fit residual is plotted as a function of the degree of the surface, the point at which the fit residual converges is then selected as the appropriate degree of the Bézier surface. Alternatively, a Bézier surface of a fixed degree could be used and the patch size adjusted: if the fit residual is too large then the Bézier patch should be fitted to a smaller region of the measured surface and vice versa.

To conclude, linear and non-linear least squares Bézier surface fitting algorithms have been derived from first principles, the latter has not previously been published for Bézier surfaces. The analytical form of the partial derivatives required for the fitting process have been presented, these were previously evaluated numerically for the case of fitting Bézier curves, not surfaces [5]. The performance of the fitting algorithm has been evaluated for simulated and measured data and shown to be suitable for approximating complex surfaces. The developed algorithm has been shown to be stable and to smoothly converge to a solution.

Conflict of interest

All authors declare that there is no conflict of interest in this paper.

References

1. A. Sóbester, A. I. J. Forrester, *Aircraft aerodynamic design: geometry and optimization*, John Wiley & Sons, West Sussex, 2015.
2. L. Shao, H. Zhou, Curve fitting with Bézier cubics, *Graphical Models and Image Processing*, **58** (1996), 223–232.
3. T. Várady, P. Salvi, M. Vaitkus, Á. Sipos, Multi-sided Bézier surfaces over curved, multi-connected domains, *Computer Aided Geometric Design*, **78** (2020), 101828.
4. J. N. Petzing, J. M. Coupland, R. K. Leach, *The measurement of rough surface topography using coherence scanning interferometry*, National Physical Laboratory, UK, 2010
5. T. A. Pastva, *Bézier curve fitting*, Thesis, Naval Postgraduate School, Monterey, California, 1998.
6. C. F. Borges, T. Pastva, Total least squares fitting of Bézier and B-spline curves to ordered data, *Computer Aided Geometric Design*, **19** (2002), 275–289.
7. P. Kovacs, A. M. Fekete, Nonlinear least-squares spline fitting with variable knots, *Appl. Math. Comput.*, **354** (2019), 490–501.
8. A. Gálvez, A. Iglesias, A. Cobo, J. Puig-Pey, J. Espinola, Bézier curve and surface fitting of 3D point clouds through genetic algorithms, functional networks and least-squares approximation, *Computational Science and Its Applications*, **4706** (2007), 680–693.
9. K. S. Reddy, A. Mandal, K. K. Verma, G. Rajamohan, Fitting of Bézier surfaces using the fireworks algorithm, *International Journal of Advances in Engineering & Technology*, **9** (2016), 396–403.
10. A. Gálvez, A. Iglesias, Firefly Algorithm for Polynomial Bézier Surface Parameterization, *J. Appl. Math.*, **2013** (2013).
11. L. Piegl, W. Tiller, *The NURBS book*, 2 Eds., Springer-Verlag, Berlin, Heidelberg, New York, 1995.
12. J. Arvo, *Graphics Gems II*, Academic Press Professional, San Diego, CA, United States, 1991.
13. T. Varady and R. Martin, *Handbook of Computer Aided Geometric Design*, North-Holland, Amsterdam, The Netherlands, 2002.
14. A. B. Forbes, *Least-squares best-fit geometric elements*, NPL Report DITC 140/89, 1991.
15. Y. Cao, D. M. Yan, P. Wonka, Patch layout generation by detecting feature networks, *Computers & Graphics*, **46** (2015), 275–282.
16. H. Lin, W. Chen, H. Bao, Adaptive patch-based mesh fitting for reverse engineering, *Computer-Aided Design*, **39** (2007), 1134–1142.
17. I. Kovács, T. Várady, Constrained fitting with free-form curves and surfaces, *Computer-Aided Design*, **122** (2020), 102816.

Appendix 1: Explicit form of a fourth degree Bézier surface

$$\binom{4}{0}=1,\binom{4}{1}=4,\binom{4}{2}=6,\binom{4}{3}=4,\binom{4}{4}=1$$

$$[P^x(u_t,v_t)\quad P^y(u_t,v_t)\quad P^z(u_t,v_t)]=$$

$$(1-u_t)^4(1-v_t)^4\left[k_{0,0}^x\quad k_{0,0}^y\quad k_{0,0}^z\right]+$$

$$(1-u_t)^4 4v_t(1-v_t)^3\left[k_{0,1}^x\quad k_{0,1}^y\quad k_{0,1}^z\right]+$$

$$(1-u_t)^4 6v_t^2(1-v_t)^2\left[k_{0,2}^x\quad k_{0,2}^y\quad k_{0,2}^z\right]+$$

$$(1-u_t)^4 4v_t^3(1-v_t)\left[k_{0,3}^x\quad k_{0,3}^y\quad k_{0,3}^z\right]+$$

$$(1-u_t)^4 v_t^4\left[k_{0,4}^x\quad k_{0,4}^y\quad k_{0,4}^z\right]+$$

$$4u_t(1-u_t)^3(1-v_t)^4\left[k_{1,0}^x\quad k_{1,0}^y\quad k_{1,0}^z\right]+$$

$$4u_t(1-u_t)^3 4v_t(1-v_t)^3\left[k_{1,1}^x\quad k_{1,1}^y\quad k_{1,1}^z\right]+$$

$$4u_t(1-u)^3 6v_t^2(1-v_t)^2\left[k_{1,2}^x\quad k_{1,2}^y\quad k_{1,2}^z\right]+$$

$$4u_t(1-u_t)^3 4v_t^3(1-v_t)\left[k_{1,3}^x\quad k_{1,3}^y\quad k_{1,3}^z\right]+$$

$$4u_t(1-u_t)^3 v_t^4\left[k_{1,4}^x\quad k_{1,4}^y\quad k_{1,4}^z\right]+$$

$$6u_t^2(1-u_t)^2(1-v_t)^4\left[k_{2,0}^x\quad k_{2,0}^y\quad k_{2,0}^z\right]+$$

$$6u_t^2(1-u_t)^2 4v_t(1-v_t)^3\left[k_{2,1}^x\quad k_{2,1}^y\quad k_{2,1}^z\right]+$$

$$6u_t^2(1-u_t)^2 6v_t^2(1-v_t)^2\left[k_{2,2}^x\quad k_{2,2}^y\quad k_{2,2}^z\right]+$$

$$6u_t^2(1-u_t)^2 4v_t^3(1-v_t)\left[k_{2,3}^x\quad k_{2,3}^y\quad k_{2,3}^z\right]+$$

$$6u_t^2(1-u_t)^2 v_t^4\left[k_{2,4}^x\quad k_{2,4}^y\quad k_{2,4}^z\right]+$$

$$4u_t^3(1-u_t)(1-v_t)^4\left[k_{3,0}^x\quad k_{3,0}^y\quad k_{3,0}^z\right]+$$

$$4u_t^3(1-u_t)4v_t(1-v_t)^3\left[k_{3,1}^x\quad k_{3,1}^y\quad k_{3,1}^z\right]+$$

$$4u_t^3(1-u_t)6v_t^2(1-v_t)^2\left[k_{3,2}^x\quad k_{3,2}^y\quad k_{3,2}^z\right]+$$

$$4u_t^3(1-u_t)4v_t^3(1-v_t)\left[k_{3,3}^x\quad k_{3,3}^y\quad k_{3,3}^z\right]+$$

$$4u_t^3(1-u_t)v_t^4\begin{bmatrix}k_{3,4}^x & k_{3,4}^y & k_{3,4}^z\end{bmatrix}+$$

$$u_t^4(1-v_t)^4\begin{bmatrix}k_{4,0}^x & k_{4,0}^y & k_{4,0}^z\end{bmatrix}+$$

$$u_t^4 4v_t(1-v_t)^3\begin{bmatrix}k_{4,1}^x & k_{4,1}^y & k^z{}_{4,1}\end{bmatrix}+$$

$$u_t^4 6v_t^2(1-v_t)^2\begin{bmatrix}k_{4,2}^x & k_{4,2}^y & k_{4,2}^z\end{bmatrix}+$$

$$u_t^4 4v_t^3(1-v_t)\begin{bmatrix}k_{4,3}^x & k_{4,3}^y & k_{4,3}^z\end{bmatrix}+$$

$$u_t^4 v_t^4\begin{bmatrix}k_{4,4}^x & k_{4,4}^y & k_{4,4}^z\end{bmatrix}$$

Appendix 2: Partial derivatives of e_t with respect to k_{ij}

$$\frac{\partial e_t}{\partial k_{0,0}} = (1-u_t)^4(1-v_t)^4$$
$$\frac{\partial e_t}{\partial k_{0,1}} = (1-u_t)^4 4v_t(1-v_t)^3$$
$$\frac{\partial e_t}{\partial k_{0,2}} = (1-u_t)^4 6v_t^2(1-v_t)^2$$
$$\frac{\partial e_t}{\partial k_{0,3}} = (1-u_t)^4 4v_t^3(1-v_t)$$
$$\frac{\partial e_t}{\partial k_{0,4}} = (1-u_t)^4 v_t^4$$
$$\frac{\partial e_t}{\partial k_{1,0}} = 4u_t(1-u_t)^3(1-v_t)^4$$
$$\frac{\partial e_t}{\partial k_{1,1}} = 4u_t(1-u_t)^3 4v_t(1-v_t)^3$$
$$\frac{\partial e_t}{\partial k_{1,2}} = 4u_t(1-u_t)^3 6v_t^2(1-v_t)^2$$
$$\frac{\partial e_t}{\partial k_{1,3}} = 4u_t(1-u_t)^3 4v_t^3(1-v_t)$$
$$\frac{\partial e_t}{\partial k_{1,4}} = 4u_t(1-u_t)^3 v_t^4$$
$$\frac{\partial e_t}{\partial k_{2,0}} = 6u_t^2(1-u_t)^2(1-v_t)^4$$
$$\frac{\partial e_t}{\partial k_{2,1}} = 6u_t^2(1-u_t)^2 4v_t(1-v_t)^3$$
$$\frac{\partial e_t}{\partial k_{2,2}} = 6u_t^2(1-u_t)^2 6v_t^2(1-v_t)^2$$
$$\frac{\partial e_t}{\partial k_{2,3}} = 6u_t^2(1-u_t)^2 4v_t^3(1-v_t)$$
$$\frac{\partial e_t}{\partial k_{2,4}} = 6u_t^2(1-u_t)^2 v_t^4$$
$$\frac{\partial e_t}{\partial k_{3,0}} = 4u_t^3(1-u_t)(1-v_t)^4$$
$$\frac{\partial e_t}{\partial k_{3,1}} = 4u_t^3(1-u_t)4v_t(1-v_t)^3$$
$$\frac{\partial e_t}{\partial k_{3,2}} = 4u_t^3(1-u_t)6v_t^2(1-v_t)^2$$
$$\frac{\partial e_t}{\partial k_{3,3}} = 4u_t^3(1-u_t)4v_t^3(1-v_t)$$
$$\frac{\partial e_t}{\partial k_{3,4}} = 4u_t^3(1-u_t)v_t^4$$

$$\frac{\partial e_t}{\partial k_{4,0}} = u_t^4(1-v_t)^4$$
$$\frac{\partial e_t}{\partial k_{4,1}} = u_t^4 4v_t(1-v_t)^3$$
$$\frac{\partial e_t}{\partial k_{4,2}} = u_t^4 6v_t^2(1-v_t)^2$$
$$\frac{\partial e_t}{\partial k_{4,3}} = u_t^4 4v_t^3(1-v_t)$$
$$\frac{\partial e_t}{\partial k_{4,4}} = u_t^4 v_t^4$$

Appendix 3: Partial derivatives of e_t with respect to u_t and v_t

$$\frac{\partial e_t}{\partial u_t} = -4(1-u_t)^3(1-v_t)^4k_{0,0} +$$

$$-4(1-u_t)^3 4v_t(1-v_t)^3k_{0,1} +$$

$$-4(1-u_t)^3 6v_t^2(1-v_t)^2k_{0,2} +$$

$$-4(1-u_t)^3 4v_t^3(1-v_t)k_{0,3} +$$

$$-4(1-u_t)^3 v_t^4k_{0,4} +$$

$$(4(1-u_t)^3 - 12u_t(1-u_t)^2)(1-v_t)^4k_{1,0} +$$

$$(4(1-u_t)^3 - 12u_t(1-u_t)^2)4v_t(1-v_t)^3k_{1,1} +$$

$$(4(1-u_t)^3 - 12u_t(1-u_t)^2)6v_t^2(1-v_t)^2k_{1,2} +$$

$$(4(1-u_t)^3 - 12u_t(1-u_t)^2)4v_t^3(1-v_t)k_{1,3} +$$

$$(4(1-u_t)^3 - 12u_t(1-u_t)^2)v_t^4k_{1,4} +$$

$$(12u_t(1-u_t)^2 - 12u_t^2(1-u_t))(1-v_t)^4k_{2,0} +$$

$$(12u_t(1-u_t)^2 - 12u_t^2(1-u_t))4v_t(1-v_t)^3k_{2,1} +$$

$$(12u_t(1-u_t)^2 - 12u_t^2(1-u_t))6v_t^2(1-v_t)^2k_{2,2} +$$

$$(12u_t(1-u_t)^2 - 12u_t^2(1-u_t))4v_t^3(1-v_t)k_{2,3} +$$

$$(12u_t(1-u_t)^2 - 12u_t^2(1-u_t))v_t^4k_{2,4} +$$

$$(12u_t^2(1-u_t) - 4u_t^3)(1-v_t)^4k_{3,0} +$$

$$(12u_t^2(1-u_t) - 4u_t^3)4v_t(1-v_t)^3k_{3,1} +$$

$$(12u_t^2(1-u_t) - 4u_t^3)6v_t^2(1-v_t)^2k_{3,2} +$$

$$(12u_t^2(1-u_t) - 4u_t^3)4v_t^3(1-v_t)k_{3,3} +$$

$$(12u_t^2(1-u_t) - 4u_t^3)v_t^4k_{3,4} +$$

$$4u_t^3(1-v_t)^4k_{4,0} +$$

$$4u_t^3 4v_t(1-v_t)^3 k_{4,1} +$$

$$4u_t^3 6v_t^2(1-v_t)^2 k_{4,2} +$$

$$4u_t^3 4v_t^3(1-v_t)k_{4,3} +$$

$$4u_t^3 v_t^4 k_{4,4}$$

$$\frac{\partial e_t}{\partial v_t} = (1-u_t)^4 -4(1-v_t)^3 k_{0,0} +$$

$$(1-u_t)^4(4(1-v_t)^3 - 12v_t(1-v_t)^2)k_{0,1} +$$

$$(1-u_t)^4(12v_t(1-v_t)^2 - 12v_t^2(1-v_t))k_{0,2} +$$

$$(1-u_t)^4(12v_t^2(1-v_t) - 4v_t^3)k_{0,3} +$$

$$(1-u_t)^4 4v_t^3 k_{0,4} +$$

$$4u_t(1-u_t)^3 -4(1-v_t)^3 k_{1,0} +$$

$$4u_t(1-u_t)^3(4(1-v_t)^3 - 12v_t(1-v_t)^2)k_{1,1} +$$

$$4u_t(1-u)^3(12v_t(1-v_t)^2 - 12v_t^2(1-v_t))k_{1,2} +$$

$$4u_t(1-u_t)^3(12v_t^2(1-v_t) - 4v_t^3)k_{1,3} +$$

$$4u_t(1-u_t)^3 4v_t^3 k_{1,4} +$$

$$6u_t^2(1-u_t)^2 -4(1-v_t)^3 k_{2,0} +$$

$$6u_t^2(1-u_t)^2(4(1-v_t)^3 - 12v_t(1-v_t)^2)k_{2,1} +$$

$$6u_t^2(1-u_t)^2(12v_t(1-v_t)^2 - 12v_t^2(1-v_t))k_{2,2} +$$

$$6u_t^2(1-u_t)^2(12v_t^2(1-v_t) - 4v_t^3)k_{2,3} +$$

$$6u_t^2(1-u_t)^2 4v_t^3 k_{2,4} +$$

$$4u_t^3(1-u_t) -4(1-v_t)^3 k_{3,0} +$$

$$4u_t^3(1-u_t)(4(1-v_t)^3 - 12v_t(1-v_t)^2)k_{3,1} +$$

$$4u_t^3(1-u_t)(12v_t(1-v_t)^2-12v_t^2(1-v_t))k_{3,2}+$$

$$4u_t^3(1-u_t)(12v_t^2(1-v_t)-4v_t^3)k_{3,3}+$$

$$4u_t^3(1-u_t)4v_t^3k_{3,4}+$$

$$u_t^4-4(1-v_t)^3k_{4,0}+$$

$$u_t^4(4(1-v_t)^3-12v_t(1-v_t)^2)k_{4,1}+$$

$$u_t^4(12v_t(1-v_t)^2-12v_t^2(1-v_t))k_{4,2}+$$

$$u_t^4(12v_t^2(1-v_t)-4v_t^3)k_{4,3}+$$

$$u_t^4 4v_t^3k_{4,4}$$