Surface Topography: Metrology and Properties

PAPER

Mathematical approach to the validation of field surface texture parameter software

Luke Todhunter¹, Richard Leach¹, Simon Lawes¹, Peter Harris¹ and François Blateyron¹

¹ Manufacturing Metrology Team, Faculty of Engineering, University of Nottingham, United Kingdom
² National Physical Laboratory, Teddington, United Kingdom
³ Digital Surf, Besançon, France

E-mail: Luke.Todhunter@nottingham.ac.uk

Keywords: surface texture, areal parameters, reference software

Abstract

A new method for performance validation of surface texture parameter calculation software is introduced, focussing on field surface texture parameters. Surface height functions are defined mathematically, either using Fourier series or polynomials, and are then input into the surface texture parameter definitions to obtain mathematical parameter values. A series of user-adjustable parametric surface functions are defined that correspond to each surface texture parameter, enabling users to create a variety of surfaces to assess their software whilst still retaining mathematical traceability. This method is expanded to include complex surface textures. Chebyshev polynomials are used to perform numerical calculations of surface texture parameters for a selection of polynomial surface functions. Mathematical reference parameter values are calculated for a series of fifteen predefined surfaces and ten parametric surfaces to assess the performance of the software under test for a given dataset resolution. Assessment of the number of significant figures of the software-obtained values that agree with the reference values is used as a performance metric that enables comparison between different third-party software applications for a given dataset resolution. An assessment of the sampling methods used to create discrete datasets of a mathematical surface function for use with numerical third-party software is performed. Two implementations of surface height sampling are used to create datasets that are input into four third-party surface texture parameter calculation software packages, and the results compared, showing a significant variation in the performance metric values for different sampling methods.

1. Introduction

Surface texture parameters are quantitative descriptors of a surface topography measurement that serve as a crucial aspect of precision engineering [1, 2], enabling greater control of functional part properties such as friction, wear and lubricant retention [3–5].

Surface texture parameter values are often calculated using third-party software, and it is important that the results obtained by these software packages agree with the parameter definitions laid out in the ISO specification standards [6, 7]. Validation of surface texture parameter calculation software is performed using software measurement standards, typically developed by national measurement institutes (NMIs) [8–13]. Software measurement standards are given in two forms: type F1/S1 reference datasets with corresponding, well-defined parameter values, and type F2/S2 reference software, which calculates parameter values for any given input dataset with high accuracy [14, 15].

While, theoretically, type F2/S2 reference software is designed to use high accuracy methods to obtain reference parameter values for a given dataset, in practice they are still discrete software implementations utilising finite precision arithmetic and numerical algorithms operating on a discretely sampled dataset, and are still subject to the same sources of approximation as commercial software [16]. Previous work has shown variation in the parameter values obtained by reference software developed by different NMIs [17].

The development of mathematical references for the validation of surface texture parameter calculation software is a valuable step towards providing users...
with a traceable assessment of the parameter values they obtain. Using this new approach, reference surfaces are defined and evaluated analytically, ensuring mathematical traceability and increased accuracy compared to the current state of the art. By moving away from reference software, which is subject to differing interpretations of algorithms, users can have more confidence in the performance of the software under test and deliver more accurate specification and analysis of high-precision parts.

Previous work introduced the use of mathematical references for functional area surface texture parameters [18], showcasing this new mathematical approach. As all functional surface texture parameters are derived from the material ratio curve and not the surface heights directly, it was logical to define the material ratio curve analytically for direct calculation of the parameters, and obtain surface datasets by sampling the curve. For field surface texture parameters, the direct surface height information is needed, and so analytical definitions of the surface height, \( z(x, y) \), are required. Previous work introduced this concept [19], and presented a graphical user interface for the creation of such surfaces.

In section 2, the analytical calculation field surface texture parameters are defined for the general case, and a series of simple test surfaces, created using a variety of analytical functions, are used to showcase the evaluation of field surface texture parameters and assess the results obtained by surface texture parameter software. Section 3 presents an alternative numerical method to overcome the current limitations of the analytical methods and obtain mathematical parameter values for a series of complex test surfaces. Assessment of surface texture parameter software is also performed. Section 4 presents methods for the analytical calculation of field parameters in the instances where calculation for the general case are not possible. Finally, section 5 introduces a method of software performance assessment that accounts for the discretisation error introduced when creating a discrete dataset from an analytical surface.

2. Mathematical parameter evaluation

2.1. Parameter evaluation

To ensure mathematical traceability to analytical surface representations, the surface texture parameter calculations must also be performed analytically. To perform analytical, or symbolic, calculations computationally, a computer algebra system (CAS) is required. A CAS is a mathematical software package or module that can manipulate symbolic mathematical expressions directly, typically using a series of rules or look-up tables dictating mathematical procedures, without relying on numerical approximation. Such an approach delivers seemingly infinite precision results, providing the system is successfully able to obtain a mathematical expression, and ensures mathematical traceability throughout the operation. There are a variety of CASs available, and while some are open-source such as SageMath [20], the systems that are most popular are closed-source and commercial products. For the majority of the work in this paper, Wolfram Mathematica 11.1 was used as the primary CAS, with MAPLE 2017 also used to verify the results of Mathematica for first time calculations (subsequent calculations using a similar mathematical form did not require MAPLE verification) [21, 22]. Both Mathematica and MAPLE are among the most capable CASs available at the time.

The calculation of each parameter requires adhering to the definitions given in ISO 25 178-2 [7, 23]. In an ideal situation, this means substituting the surface equation for \( z(x, y) \) in the parameter definition and evaluating the result. The following sections give the mathematical operations used to calculate each parameter, and, where necessary, any alternative steps taken to circumvent limitations of the CAS used.

Before any parameter calculation can be performed, the surface function must be adjusted so that it has a mean-plane of zero. This is achieved by simply finding the mean height across the entire evaluation area of the surface and subtracting it from the surface function, given by

\[
z(x, y) = z_0(x, y) - \frac{1}{A_{\text{eval}}} \int_{A_{\text{eval}}} z_0(x, y) \, dx \, dy,
\]

where \( z_0(x, y) \) is used to represent the unadjusted surface function, the integration region, \( A_{\text{Bl}} \), is the defined evaluation area of the surface, \( A_{\text{eval}} \) is the total evaluation area, and \( z(x, y) \) will represent the adjusted surface function in all future instances.

2.1.1. Sq, Ssk and Sku

The first three parameters, \( Sq \), \( Ssk \) and \( Sku \), represent the root mean square, skewness and kurtosis of the surface, respectively [7]. The definitions for each of these have similar forms, relying on a integral of some power function of \( z(x, y) \). Such an operation is well supported by most modern CASs, and so these parameters can be evaluated directly using the original definitions,

\[
Sq = \sqrt{\frac{1}{A_{\text{eval}}} \int_{A_{\text{eval}}} z^2(x, y) \, dx \, dy},
\]

\[
Ssk = \frac{1}{Sq} \left( \frac{1}{A_{\text{eval}}} \int_{A_{\text{eval}}} z^3(x, y) \, dx \, dy \right),
\]

and

\[
Sku = \frac{1}{Sq} \left( \frac{1}{A_{\text{eval}}} \int_{A_{\text{eval}}} z^4(x, y) \, dx \, dy \right).
\]

In all surface texture parameter equations throughout this document, parameter symbols are given as a single letter with subscripts as needed, which differs from the parameter form given in text. The reason for this differentiation, in accordance with the ISO
25 178–2 definitions, is to avoid misinterpretation of compound letters as an indication of multiplication between quantities in equations [7]. As both Ssk and Sku rely on Sq in their definitions, Sq is calculated first, and the obtained expression substituted into the expression for Ssk and Sku. In the definitions given above, the double integral is performed over the defined area, A<sub>R</sub>, which is the chosen domain of the continuous analytical expression z(x, y) over which the surface area is defined. A<sub>R</sub> is assumed to be a rectangle defined by the interval [x<sub>0</sub>, x<sub>h</sub>] in the x direction and [y<sub>0</sub>, y<sub>h</sub>] in the y direction.

2.1.2. Sp, Sv and Sz

The next three parameters, Sp, Sv and Sz, are conceptually simple in that they relate to the largest peak/valley height values on the surface [7]. Sp corresponds to the largest peak height within the defined area, Sv corresponds to the magnitude of largest valley depth, and Sz is the total distance between the largest peak height and largest valley depth. For a discrete dataset, maximum peak/valley heights are obtained trivially by sorting the discrete heights and obtaining the highest and lowest values.

For a continuous analytical surface, obtaining maximum peak/valley heights symbolically becomes more complex. The parameters require calculating the global maxima and minima of the function within the specified region. Several optimisation techniques can potentially obtain values corresponding to only local minima, not global minima, and so care must be taken to avoid this. Modern CASs have many global optimisation functions available, each with strengths and weaknesses, which can be adopted to find these global maxima and minima [24].

A more manual approach to obtaining the largest peak and valley can be performed by finding all local minima and maxima points within the defined area using first and second derivatives, and sorting the results. Local minima/maxima are found at values of z (x, y) with zero gradient values in both x and y,

\[
\frac{\partial z(x, y)}{\partial x} = 0, \quad \frac{\partial z(x, y)}{\partial y} = 0.
\]

Furthermore, local maxima are found at locations where the second derivative of the function z(x, y) is less than zero, and local minima are found where the second derivative is greater than zero, given by

\[
\frac{\partial^2 z(x, y)}{\partial x^2} < 0, \quad \frac{\partial^2 z(x, y)}{\partial y^2} < 0
\]

and

\[
\frac{\partial^2 z(x, y)}{\partial x^2} > 0, \quad \frac{\partial^2 z(x, y)}{\partial y^2} > 0,
\]

respectively. By solving these equations simultaneously, values for x and y can be found that correspond to each local maxima and minima. By evaluating each of these points in z(x, y), expressions for surface height can be found, which can then be sorted to obtain the largest peak and valley.

Using the method of minimising the derivatives of a surface, any values that are not located at a zero-gradient in both x and y will not be found. This can miss the true Sp or Sv values if they happen to be located on the boundary of the defined area, where the zero-gradient for both x and y is not necessarily satisfied. Visual inspection of the surface can help to identify if this is the case for any individual scenario. If so, simple one-dimensional gradient analysis along the boundary line, at z(x, y) for example, can be performed for each of the four boundary edges to find any zero-gradient locations, which can then be evaluated to find the expressions for the heights. Finally, direct evaluation of z(x, y) at each for the four boundary corners can be performed to ensure any potential maxima/minima are identified there, as these points may not satisfy the boundary gradient analysis conditions, for similar reasons as before.

2.1.3. Sal and Str

Sal and Str are part of a subset of field parameters called spatial parameters [7]. Spatial parameters address the similarities of a surface with itself if spatially translated by a certain amount in the x and y directions. Such an analysis is valuable in identifying the uniformity of the surface. Sal assesses how abruptly a surface’s height changes, and Str gives the ratio of fast-changing directions to slow-changing directions, indicating the presence of any directionality of the surface texture, such as grinding marks.

These parameters are calculated on the autocorrelation function of the surface, f<sub>ACF</sub>(t<sub>x</sub>, t<sub>y</sub>). The autocorrelation function describes the degree of agreement between the surface function, z(x, y), and the same surface translated by t<sub>x</sub> and t<sub>y</sub> in the x and y directions, respectively. The definition for the autocorrelation function is given in ISO 25 178–2 as

\[
f_{ACF}(t_x, t_y) = \frac{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} z(x, y) \cdot z(x - t_x, y - t_y) \, dx \, dy}{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} z(x, y) \cdot z(x, y) \, dx \, dy}.
\]

The denominator of equation (8) is used to normalise the autocorrelation function to give a value of 1 at t<sub>x</sub> = 0 and t<sub>y</sub> = 0. Calculation of the autocorrelation function requires evaluation of integrals similar to SQ, Ssk and Sku, and thus can be performed using a CAS.

Sal and Str are obtained by finding the distance it takes for f<sub>ACF</sub> to decay from 1 to a specified value, s, defaulting at s = 1/5 in ISO 25 178–3 [25]. This is achieved by finding the expression for the contour line of

\[
f_{ACF}(t_x, t_y) = 1/5.
\]

Sal is defined by the shortest distance from the origin, (0,0), to the contour line. The distance from the origin
to a point on the contour is given by
\[
\text{Distance} = \sqrt{t_x^2 + t_y^2},
\] (10)

By rearranging equation (9) in terms of \( t_x \) and substituting into equation (10), it is possible to obtain an expression in one dimension that gives the distance from the contour to the origin. This distance can then be minimised within the defined surface area to find the shortest distance to the origin, corresponding to \( \text{Sal} \). Maximum points can be found in a similar fashion, and the ratio of shortest distance to longest distance gives the value for \( \text{Str} \).

### 2.1.4. \( \text{Sdq} \) and \( \text{Sdr} \)

\( \text{Sdq} \) and \( \text{Sdr} \) are hybrid parameters, and related to the slope of the surface [7]. These parameters assess the steepness of features on the surface and can give valuable information for surfaces that could otherwise present similar \( \text{Sa} \) or \( \text{Sq} \) values. Hybrid parameters can be useful for the assessment of sealing performance and wettability, or any application that requires understanding of average surface gradient values.

These parameters require calculation of the surface gradients in the \( x \) and \( y \) directions, which can be obtained using partial derivatives as shown in section 2.1.2 for calculating \( \text{Sp} \) and \( \text{Sy} \). Once the partial derivatives are calculated, \( \text{Sdq} \) can be found by evaluating the definition

\[
\text{Sdq} = \frac{1}{A_{\text{eval}}^2} \int_{A_{\text{eval}}} \left[ \left( \frac{\partial z(x, y)}{\partial x} \right)^2 + \left( \frac{\partial z(x, y)}{\partial y} \right)^2 \right] \, dx \, dy,
\] (11)

which can be performed similarly to that discussed in section 2.1.1 for \( \text{Sq} \), \( \text{Ssk} \) and \( \text{Sku} \). \( \text{Sdr} \) has the definition

\[
\text{Sdr} = \frac{1}{A_{\text{eval}}^2} \int_{A_{\text{eval}}} \left[ \sqrt{1 + \left( \frac{\partial z(x, y)}{\partial x} \right)^2 + \left( \frac{\partial z(x, y)}{\partial y} \right)^2} - 1 \right] \, dx \, dy,
\] (12)

which, while appearing similar in composition to \( \text{Sdq} \) at first glance, poses a problem to CASs due to the square root of disparate terms within the integral being inseparable. Calculating \( \text{Sdr} \) using this definition, for the general case of an analytical surface expression of any form, requires the object within the integral to be evaluated as a single object, as it is inseparable. This particular inseparable evaluation leads to incalculable symbolic expressions and hanging run-times, due to seemingly endless cycles of techniques such as integration by parts, wherein an integral can be simplified by performing an integral on the derivative of some part of the original function. In this instance, the derived function would still be as complex as the original function, requiring further integration by parts and causing the recursive cycles. Equation (12) takes the form of a nonelementary integral, which is an integral of an elementary function whose solution is not elementary (an elementary function is a function comprised of a finite number of arithmetic operations on traditional functions such as powers, roots, trigonometric functions, exponential functions and logarithmic functions). The concept of a nonelementary integral is proven in Liouville’s theorem [26, 27], and evaluations of nonelementary integrals require the use of either Taylor series expansion (to a required order), numerical integration or nonelementary ‘special’ functions (such as incomplete elliptical integrals or error functions). Both Taylor expansion and numerical integration are approximations, giving the result to a finite precision. Nonelementary special functions are advanced mathematical concepts that cannot be expressed using regular elementary functions, making it challenging to promote widespread use and adoption, and so will not be used here.

Practically, \( \text{Sdr} \) is the ratio between the planar area of evaluation of the surface, \( A_{\text{eval}} \), and the total topographical area of the surface itself. The topographical area can be approximated by splitting the surface into a series of triangles, \( BCD \), and summing the area of each of the triangles using the formula

\[
\text{Area} = \frac{1}{2} \left[ \hat{u} \times \hat{v} \right],
\] (13)

where \( \hat{u} \) is the vector that joins point \( B \) to point \( C \), and \( \hat{v} \) is the vector that joins point \( B \) to point \( D \). By sampling a grid of height values from the surface expression, splitting each square defined by four height values into two triangles where the height values describe the vertices, and evaluating the cross product defined in equation (13), an expression can be derived for the approximation of the total topographical area of the surface,

\[
\text{Area} = \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{\delta x \delta y}{2} \left( \sqrt{1 + \frac{(Z_{m-1,n+1} - Z_{m,n+1})^2}{\delta x^2} + \frac{(Z_{m,n+1} - Z_{m,n-1})^2}{\delta y^2}} + \sqrt{1 + \frac{(Z_{m+1,n+1} - Z_{m+1,n})^2}{\delta x^2} + \frac{(Z_{m+1,n} - Z_{m+1,n-1})^2}{\delta y^2}} \right),
\] (14)

where \( N \) and \( M \) are the total number of vertices in the \( x \) and \( y \) directions, respectively, and \( \delta x \) and \( \delta y \) are the widths between adjacent vertices in the \( x \) and \( y \) directions, respectively. \( Z_{n,m} \) describes the surface height values at each vertex,
\[ Z_{n,m} = Z(x_n, y_m), \]  
where
\[ x_n = x_0 + (n - 1)\delta x \]  
\[ y_m = y_0 + (m - 1)\delta y. \]

Here, \( x_0 \) and \( y_0 \) are the coordinates of the starting corner of the surface area of interest. By using this method with an analytical surface expression, it is possible to define \( \delta x \) and \( \delta y \) to be any desired value. \( \delta x \) and \( \delta y \) can be reduced to the infinitesimal case, allowing a value for the total topographical area to be found that is equivalent to a symbolic evaluation. In practice, this would require seemingly infinite computational resources, and so compromises must be made, instead calculating to a precision that is appropriate for the required application. This approach is an alternative workaround to the limitations of calculating \( S\delta r \) in the general case for any surface, however, it sacrifices the mathematical traceability by resorting to discrete, numerical methods. An alternative method, that calculates \( S\delta r \) analytically for a specific case, is presented in section 4.4.

2.1.5. \( S_a \)

\( S_a \) is one of the most popular areal surface texture parameters in industry [28]. \( S_a \) is simply the mean deviation of height values on the surface from the mean plane and is broadly used in industry to define the ‘roughness’ of a surface, as a high \( S_a \) value would correspond to a larger spread between high and low points on a surface. However, due to the simplicity of the definition, the practical application of this metric, for example to understand lubricant retention, is limited, as surfaces with deterministic elements can skew the \( S_a \) value.

Mathematically, \( S_a \) is defined as the arithmetic mean of the absolute of the surface heights within the evaluation area,
\[ S_a = \frac{1}{A_{eval}} \int_{A_e} |z(x, y)| \, dx \, dy. \]  

Unfortunately, due to the absolute function, direct integration of the analytical surface expression is not possible.

All areas of the surface that lie below the mean plane are effectively mirrored to the positive \( z \)-axis, causing discontinuous edges at the points where the surface meets the mean plane. These discontinuities cannot be integrated as a single function. For simple cases, it is possible to separate the full discontinuous expression into a series of region-bound expressions and evaluate them all individually, however, a new expression is required for each region, along with information about the boundaries for each region (which are not necessarily linear, thus requiring line integrals). For more complex surfaces with larger numbers of mean plane crossing points, the formulation of each discrete term becomes prohibitively complex, particularly as the process would be bespoke for each individual surface expression, and not something easily implementable within a CAS. An alternative solution for simple parametric surfaces has been implemented in section 4.

2.2. Assessment of third-party software

A selection of ten simple analytical surfaces were defined for use in showcasing the calculation of field surface texture parameters. In order to highlight the variety of surface functions that can be used, three different types of function were selected:

- Three cosine functions, using the technique introduced in [19].
- Five polynomial functions, comprising of a cubic polynomial and four Bernoulli polynomials (two fourth order and two fifth order).
- Two Gaussian functions, using the natural exponential function, \( e^x \).

To showcase the validity of this new approach, comparisons were performed with a selection of third-party parameter calculation software packages. A total of four different software packages were used, labelled A to D, which included a combination of commercial software and software developed by NMIs. Each software package was given a 700 \( \times \) 700 high resolution SF surface. SDF dataset sampled from the surface expressions defined above, where resolution is defined as the number of individual height values included in the surface dataset. This resolution was chosen because software D suffered from an upper limit on the file size of datasets of 10 MB, and a resolution of 700 \( \times \) 700 was the highest resolution that reliably came under this limit, ensuring comparisons to be possible across all four software packages. No additional form removal or filtration operations were performed to enable sole assessment of the parameter value calculation methods.

The results of each of the third-party software tests are normalised relative to the mathematically-obtained parameter values to allow for more direct comparisons. Figure 1 shows the normalised values obtained for the \( S_q \) parameter, relative to the mathematical value, for a selection of eight analytical surfaces. Upon initial investigation, all software packages perform well in calculating \( S_q \), however, closer inspection shows deviation from the mathematical value of the order of tenths of a percent, particularly for software B and D. This level of deviation could prove important when dealing with high precision applications, and helps metrology software developers to identify the aspects of their software that perform worse than competitors, highlighting key areas where their software requires improvement.

Figures 2 and 3 show the results for the \( S_p \) and \( S_{dq} \) parameters, respectively. All software packages under test
show good agreement with the mathematically-obtained values for Sp, which is expected due its simple definition of maximum peak height within the evaluation area. Discrete-based software can obtain the largest height value in a surface dataset easily with simple sorting algorithms for the height values (although this may not correspond to the true peak, depending on where the continuous surface was sampled). In contrast, the Sdq parameter values show more significant differences from the mathematically-defined values, although these variations are still within a tenth of a percent of the mathematically-obtained value. Despite these deviations, three of the four software packages perform similarly. The definition of the Sdq parameter is based on the average gradient of the surface, which can be highly dependent on the scale at which the surface is measured [29]. This effect can also be affected by the resolution of the surface, and so it should be considered that the 700 × 700 dataset sampling resolution of the analytical surfaces is a primary reason for the deviations.

The results for autocorrelation parameter Sal are given in figure 4, for a selection of six analytical surfaces for which the autocorrelation parameters were calculable. In comparison with the previous results, the scale of deviations from the mathematically-obtained value is immediately apparent, with differences of approximately
10% common. This suggests the increased complexity and higher number of algorithmic steps required to calculate the parameter, relative to the simpler parameters, is contributing to the deviations in the values obtained by the software under test. It should be noted here that only software packages A and B provided the option to calculate autocorrelation parameters, and so results for only two of the four software packages under test are presented here.

This novel method of software assessment by using mathematically-obtained reference values has demonstrated an ability to identify areas of the software under test that would benefit the most from improvement. By identifying which parameters deviate the most from a mathematical reference, in comparison to other software packages, resources can be allocated more efficiently, ensuring the weakest areas of the software are addressed first. The amount of information given to the software via a discrete dataset input can put a limit on the software’s ability to obtain a parameter value close to the mathematical reference value. However, by performing an assessment at several resolutions of input surface dataset, software developers can identify to what extent the deviation from the mathematical parameter value is due to discretisation, and what is due to shortcomings in the software. This idea is addressed further in section 5.

3. Numerical parameter evaluation

As discussed in section 2, there are certain mathematical operations, such as those required for $S_d$ and $S_d q$, 

---

**Figure 3.** Software obtained values for the $S_d q$ parameter, normalised to the mathematically-obtained value, for a series of analytically-defined surfaces. *Top:* Full view of the results *Bottom:* Zoomed-in view of the results for software A, B and C.
that are too complex to be reliably calculated in current CASs (results may be calculable for individual cases with some manipulation, but not in a general sense). In addition, very complex analytical surfaces with a high number of mathematical terms (> 1000) can prove to be too computationally intensive for modern high-performance computers. As a workaround to both of these options, numerical methods can be implemented that perform discrete-based calculations using numbers stored to a finite precision far less than those used in CASs. This approach can lead to approximations and inaccuracies in the results on a potentially significant scale, at the very least due to rounding errors [16].

For the purpose of assessing the performance of software, these issues can be minimised by performing reference calculations using methods with an operating precision better than that for the software under test. In this way, numerically-obtained reference values can retain their value and be reliably compared against to assess software performance, up to a certain precision. By unlocking the use of extended precision numerical methods, complex calculations can be performed, and the scope of the mathematical references can be broadened.

3.1. Chebfun

To perform the extended precision calculations, the MATLAB plug-in Chebfun was used [30, 31]. Chebfun is an open source software package that utilises the idea that smooth functions can be effectively represented by expansions of Chebyshev polynomials (which are a series of recursively defined polynomials whose roots are used in polynomial interpolation techniques). Analytical functions are stored using the roots of Chebyshev polynomials, and are able to be represented accurately to machine precision, that is, fifteen to sixteen significant decimal digits. The use of Chebyshev polynomials to represent smooth non-periodic functions has been seen as an equivalent to Fourier series and its ability to represent smooth periodic functions and is a major component of the branch of mathematics known as approximation theory [32, 33]. The nature of the process is stable for even very large numbers of Chebyshev points, corresponding to complex analytical surfaces.

The aim of the Chebfun software is to achieve for functions what floating-point arithmetic achieves for numbers: rapid computation in which each successive operation is carried out exactly apart from a rounding error that is very small in relative terms [34]. This precise, numerical approach to computation with functions is an effective solution for overcoming the current computational limitations of purely symbolic calculations using CASs.

3.2. Comparison with mathematical evaluation

Before applying the Chebfun methods to new applications, such as complex surfaces and analytically challenging parameters, it is first worthwhile to perform a test to gain confidence in Chebfun’s performance. This can be achieved by using the Chebfun method to calculate the same parameters as can be obtained symbolically using a CAS.

A selection of seven parameters were calculated for eight analytical surfaces mathematically, using the methods described in section 2.1. Each of the analytical results were then evaluated to fifteen significant decimal figures and stored numerically. Each analytical surface expression was then input into MATLAB

Figure 4. Software obtained values for the Sal parameter, normalised to the mathematically-obtained value, for a series of analytically-defined surfaces.
Table 1. Obtained parameter values for analytical surface expressions from both symbolic and Chebfun methods.

| Surface name          | Method     | S<sub>j</sub>                      | S<sub>k</sub>                      | S<sub>ku</sub>                      | S<sub>p</sub>                      |
|-----------------------|------------|-----------------------------------|-----------------------------------|-----------------------------------|-----------------------------------|
| 2 term cosine         | Symbolic   | 1.00000000000000E-06              | 0.00000000000000E+00              | 2.25000000000000E+00              | 2.00000000000000E-06              |
|                       | Chebfun    | 1.00000000000000E-06              | -8.81620763116716E-17             | 2.25000000000000E+00              | 2.00000000000000E-06              |
| 3 term cosine         | Symbolic   | 7.0896096303098E-07               | 1.69670632288577E-01              | 2.55374822035268E+00              | 1.69038664043398E-06              |
|                       | Chebfun    | 7.0896096303098E-07               | 1.69670632288577E-01              | 2.55374822035268E+00              | 1.69038664043398E-06              |
| 4 term cosine         | Symbolic   | 8.43910029893118E-07              | 1.21384118435050E-01              | 2.10108580181232E+00              | 1.91356806070197E-06              |
|                       | Chebfun    | 8.43910029893118E-07              | 1.21384118435050E-01              | 2.10108580181232E+00              | 1.91356806070197E-06              |
| Cubic polynomial      | Symbolic   | 5.348787134343747E-05             | -8.69318622977992E-01             | 3.75227221178735E+00              | 1.11211299937499E-04              |
|                       | Chebfun    | 5.348787134343747E-05             | -8.69318622977992E-01             | 3.75227221178735E+00              | 1.11211299937499E-04              |
| Separable 4th Bernoulli| Symbolic | 3.08606699924183E-08             | -9.56752706058658E-02             | 2.62929932812285E+00              | 5.83333333333333E-08              |
|                       | Chebfun    | 3.08606699924183E-08             | -9.56752706058658E-02             | 2.62929932812285E+00              | 5.83333333333333E-08              |
| 4th Bernoulli         | Symbolic   | 4.76190476190476E-10              | 1.83075148101313E-02              | 2.36728582500112E+00              | 1.11111111111111E-09              |
|                       | Chebfun    | 4.76190476190476E-10              | 1.83075148101313E-02              | 2.36728582500112E+00              | 1.11111111111111E-09              |
| Separable 5th Bernoulli| Symbolic | 2.452044119306874E-08             | 0.00000000000000E+00              | 2.24729792807811E+00              | 4.89163817393952E-08              |
|                       | Chebfun    | 2.452044119306874E-08             | 0.00000000000000E+00              | 2.24729792807811E+00              | 4.89163817393952E-08              |
| 5th Bernoulli         | Symbolic   | 3.00625300625300E-10              | 0.00000000000000E+00              | 2.23381677323936E+00              | 5.98203100618559E-10              |
|                       | Chebfun    | 3.00625300625300E-10              | 0.00000000000000E+00              | 2.23381677323936E+00              | 5.98203100618559E-10              |
and converted into a two-dimensional Chebfun object over the required evaluation area. Algorithms for the calculation of each surface texture parameter were performed, utilising the relevant Chebyshev-specific functions. The results for each are given in tables 1 and 2 and show good agreement between the two methods. The Chebfun method is consistently able to match the mathematical result to within fourteen significant decimal figures, with the final fifteenth significant figure often only deviating by at most one digit.

3.3. Complex surfaces

With the Chebfun approach successfully able to obtain parameter values that agree with the mathematically obtained values to machine precision (minus rounding error in some cases), the next step is to extend this approach to more complex surfaces that better simulate realistic surfaces.

A selection of five surfaces were created with increased complexity, three pseudo-random surfaces with increasing high spatial frequency components, and two modified versions of simple surfaces from section 2.2 that have had pseudo-random elements added. Each of these surfaces were created utilising the Chebfun framework. For each, an analytical expression is converted into a Chebfun object and then used to obtain surface texture parameter values. In addition, alternative numerical calculation methods can be used alongside the Chebfun methods, and the number of significant figures that agree between both methods can be used as the reference value, increasing the confidence in the presented values.

Table 2. Obtained parameter values for analytical surface expressions from both symbolic and Chebfun methods.

| Surface name          | Method      | \(S_v\)          | \(S_z\)          | \(S_{dq}\)         |
|-----------------------|-------------|------------------|------------------|-------------------|
| 2 term cosine         | Symbolic    | 2.00000000000000E-06 | 4.00000000000000E-06 | 3.14159265358979E-03 |
|                       | Chebfun     | -2.00000000000000E-06 | 4.00000000000000E-06 | 3.14159265358979E-03 |
| 3 term cosine         | Symbolic    | -1.52250330348555E-06 | 3.212 889 645 782 53E-06 | 9.078 222 484 715 97E-04 |
|                       | Chebfun     | -1.52250330348555E-06 | 3.212 889 645 782 53E-06 | 9.078 222 484 715 98E-04 |
| 4 term cosine         | Symbolic    | -1.65884305398862E-06 | 3.572 411 114 690 60E-06 | 2.590 703 411 282 46E-03 |
|                       | Chebfun     | -1.65884305398862E-06 | 3.572 411 114 690 60E-06 | 2.590 703 411 282 46E-03 |
| Cubic polynomial      | Symbolic    | 2.200 000 000 000 00E-04 | 3.312 112 999 374 99E-04 | 1.743 559 577 416 26E-04 |
|                       | Chebfun     | -2.20000000000000E-04 | 3.312 112 999 374 99E-04 | 1.743 559 577 416 26E-04 |
| Separable 4th Bernoulli| Symbolic    | 6.666 666 666 666 66E-08 | 1.250 000 000 000 00E-07 | 1.951 800 145 897 06E-04 |
|                       | Chebfun     | -6.66666666666666E-08 | 1.250 000 000 000 00E-07 | 1.951 800 145 897 06E-04 |
| 4th Bernoulli         | Symbolic    | 9.722 222 222 222 22E-10 | 2.083 333 333 333 33E-09 | 4.259 177 099 999 59E-06 |
|                       | Chebfun     | -9.72222222222222E-10 | 2.083 333 333 333 34E-09 | 4.259 177 099 999 60E-06 |
| Separable 5th Bernoulli| Symbolic    | -4.89163817393952E-08 | 9.783 276 347 879 04E-08 | 1.543 033 499 620 91E-04 |
|                       | Chebfun     | -4.89163817393951E-08 | 9.783 276 347 879 04E-08 | 1.543 033 499 620 91E-04 |
| 5th Bernoulli         | Symbolic    | -5.98203100618559E-10 | 1.196 406 201 237 11E-09 | 2.675 396 279 616 04E-06 |
|                       | Chebfun     | -5.98203100618559E-10 | 1.196 406 201 237 12E-09 | 2.675 396 279 616 03E-06 |

\[
z(x, y) = \sum_{m=0}^{N} \sum_{n=0}^{M} a_{n,m} T_n(x) T_m(y) \quad (20)
\]

and substituting them into the following equation.

3.4. Assessment of third-party software

The five complex surfaces were converted into Chebfun objects and then used to obtain surface texture parameters.

As the limitations of some symbolic calculations have been alleviated due to the numerical-based approach, parameters \(S_v\) and \(S_{dr}\) were also calculated in addition to the original set shown in section 2.2. Whilst the Chebfun method’s calculation of the original set of parameters was verified against symbolic mathematical calculation in section 3.2, no such verification could be performed for \(S_u\) and \(S_{dr}\). Therefore, although Chebfun may be able to calculate these parameters as accurately as the original set, caution must be taken when using the Chebfun parameter values as a reference, as errors may be present. For this reason, the calculation methods used should be explicitly stated alongside the obtained value to inform users of the potential sources of error. This is good practice that promotes transparency and should be carried out for any method of calculating reference values. In addition, alternative numerical calculation methods can be used alongside the Chebfun methods, and the number of significant figures that agree between both methods can be used as the reference value, increasing the confidence in the presented value.
With these reference parameter values obtained, it is now possible to use these values to assess the performance of third-party metrology software packages. The process was performed in a similar method to section 2.2; the five complex surfaces were sampled to create high resolution 700×700 SDF datasets which were then input into each third-party software package to calculate parameters. The results were then normalised to the mathematically-obtained values to enable easier comparisons.

Figure 6 shows the results for the $S_{sk}$ parameter obtained by each of the software packages under test for all five complex surfaces. An interesting result to note here is the apparent similarities and differences between the underlying calculation methods employed by each of the software packages. Software A, C and D obtain similar results for each surface; even those for which they deviate from the reference value by as much as 3%. This suggests software A, C and D each calculate $S_{sk}$ using a similar approach. Software B, however, clearly uses a different calculation method, allowing it to remain more consistently close to the reference value, but does not perform as well as the best cases of the other software packages.

Figure 7 shows the results for the $S_t$ parameter, newly calculable using the Chebfun method. All

---

Figure 5. Surface topography images of the five surfaces created using the Chebfun method. Top left: A matrix of 5×5 random Chebyshev coefficients. Top right: A matrix of 10×10 random Chebyshev coefficients. Middle left: A matrix of 20×20 random Chebyshev coefficients. Middle right: The 5th Bernoulli simple surface defined in section 2.2 with added random Chebyshev components. Bottom: The 4 term cosine simple surface defined in section 2.2 with added random Chebyshev components.
software packages under test show good agreement with the reference value, deviating by no more than one tenth of a percent. Here, software B shows the greatest amount of variation, suggesting the methods employed to calculate the $Sa$ parameter are not as accurate as those employed by the other three software packages. Again, there is very close agreement between software A, C and D.

Figure 6. Software obtained values for the $Sk$ parameter, normalised to the mathematically-obtained value, using Chebfun, for a series of analytically-defined surfaces.

Figure 7. Software obtained values for the $Sa$ parameter, normalised to the mathematically-obtained value, using Chebfun, for a series of analytically-defined surfaces.

Figure 8 shows the results for the $Sdr$ parameter. Here, there are significant differences in the values obtained by software A and C, suggesting $Sdr$ is one parameter that the two software packages calculate differently. Similar to the results for the $Sdq$ parameter shown in figure 3, the $Sdr$ parameter is affected by the resolution of the dataset, and the results here show that surfaces with more high spatial frequency components are the more significantly affected, with each of the software packages performing worse for those cases. It should be noted here that software D only gave $Sdr$ results to two decimal places, making meaningful comparisons with the other software package results difficult.
Through the use of the numerical methods made available using Chebfun, high precision reference parameter values can be obtained not only for parameters whose calculations are prohibitive for symbolic calculations using CASs, but also for surfaces with an increased degree of complexity that would otherwise be too computationally intensive. Using these new methods, a greater understanding of the strengths and weaknesses of the software under test can be performed using a wider range of surface texture parameters. In addition, the use of high complexity surfaces with high spatial frequency components delivers more information about the software under test, and how the calculated parameter values can vary.

4. Parametric surfaces

In section 2, methods for calculating surface texture parameters were presented that are applicable to any general surface. That is, for any given analytical surface function describing surface height given $x$, $y$ coordinates, those general symbolic methods could be performed to obtain parameter values. For computationally prohibitive cases, section 3 introduced the use of numerical methods to extend the application to more complex computations, still using the general case.

Alternatively, it is possible to define a limited set of analytical reference surfaces which have been designed
by using a specific surface texture parameter as a starting point. Using this approach, reference pairs [35] of surfaces and parameter values can be achieved symbolically for parameters that are incalculable for the general case. In addition, these surfaces can be defined parametrically, in terms of variables instead of fixed numbers, allowing for any number of surfaces to be created with known parameter values, given they are defined in accordance with specific rules. As a downside to the general case, not all parameters will be calculated for each surface.

In this work, parametric surfaces have been created as an alternative to using numerical methods, such as Chebfun, that allow for the complete symbolic calculation of the full range of field surface texture parameters. Further details of how to obtain symbolic values for surfaces that were not possible in the general case are given in the following sections.

4.1. \textit{Sal} and \textit{Str}

As mentioned previously, \textit{Sal} and \textit{Str} are based on the autocorrelation function, and are calculated using the longest and shortest distances from the origin to a specified autocorrelation value. For the general case, difficulty can occur when applying the autocorrelation definition to the surface function, leading to an autocorrelation function of unknown shape which must be analysed to find the required decay lengths. This can be avoided by using a two-dimensional Gaussian function, whose autocorrelation function is also a Gaussian function. The Gaussian function about the origin is given by

\[ f_{ACP}(t_x, t_y) = e^{-(at_x^2 + bt_x t_y + ct_y^2)}, \]

where

\[ a = \frac{\cos^2(\theta)}{\sigma_x^4} + \frac{\sin^2(\theta)}{\sigma_y^4}, \]

\[ b = -\frac{\sin(2\theta)}{\sigma_x^2} + \frac{\sin(2\theta)}{\sigma_y^2}, \]

\[ c = \frac{\sin^2(\theta)}{\sigma_x^4} + \frac{\cos^2(\theta)}{\sigma_y^4}, \]

where \( \sigma_x \) and \( \sigma_y \) are the standard deviations, or widths, of the two-dimensional bell, and \( \theta \) is the rotation of the Gaussian function about its centre. Using the default specified height of 0.2, the equation for the contour of interest becomes

\[ e^{-(at_x^2 + bt_x t_y + ct_y^2)} = \frac{1}{5}, \]

which can be rearranged to obtain

\[ \frac{at_x^2}{\ln 5} + \frac{bt_x t_y}{\ln 5} + \frac{ct_y^2}{\ln 5} = 1, \]

which is the form of an equation for an ellipse, where the semi-major and semi-minor axes are given by \( \sigma_x \sqrt{\ln 5} \) and \( \sigma_y \sqrt{\ln 5} \) respectively. Therefore, the values of \textit{Sal} and \textit{Str} can be calculated directly using the semi-major and semi-minor axes from a surface defined in terms of \( \sigma_x \), \( \sigma_y \), and \( \theta \).

4.2. \textit{Sa}

Section 2.1.5 explains how the definition of \textit{Sa} leads to a discontinuous surface function, causing problems when trying to evaluate a surface in the general case. Discontinuities occur after the application of the absolute function at the points where the surface crosses the zero-plane with a non-zero gradient. The absolute function causes the sign of the gradient to switch, leading to an abrupt change in surface slope. This can be avoided by defining a surface in such a way that the gradient at the zero-line is always zero.

One example of this is a modified version of a simple cosine wave, such that the sign of the wave is inverted for every second period. This example is given by an equation of the form

\[
Z(x, y) = \begin{cases} 
A(1 - \cos(f_1 \pi x)), & 0 \leq x \mod \frac{4}{f_1} < \frac{2}{f_1}, \\
-A(1 - \cos(f_1 \pi x)), & \frac{2}{f_1} < x \mod \frac{4}{f_1} < \frac{4}{f_1}.
\end{cases}
\]

(27)

where \( A \) is the amplitude of the cosine wave and \( f_1 \) is the frequency. A profile example is given in figure 9. When the absolute function is applied to this surface, the parts of the surface below the zero-plane, which are the inverted sections of the surface, return above the zero-plane and recreate the unmodified cosine wave, for which \textit{Sa} can easily be found.

4.3. \textit{Std}

The \textit{Std} parameter is defined as the texture direction of the scale-limited surface and gives the angle (with respect to a specified direction) of the most prominent texture on a surface. Mathematically, this is given as the absolute maximum value of the angular power spectrum of the surface for a given direction, \( s \) given by

\[
f_{APs}(s) = \int_{R_1}^{R_2} |F[r \sin (s - \theta), r \cos (s - \theta)]|^2 dr,
\]

(28)

where \( R_1 \) and \( R_2 \) denote the range of integration in the radial direction, and also requires the Fourier transformation

\[
F(u, v) = \int_{-\lambda_u}^{\lambda_u} \int_{-\lambda_v}^{\lambda_v} z(x, y) e^{-i(ux + vy)} dx dy,
\]

(29)

where \( u \) and \( v \) are spatial frequencies in the \( x \) and \( y \) directions, respectively. This calculation, involving two transformations and an absolute function (which has already been explained to be symbolically challenging for the case of \textit{Sa}, see section 2.1.5), poses a complex symbolic challenge for the general case. Instead, it is possible to create reference analytical expressions with a known \textit{Std} value that operate based
on the parameter’s more practical definition: the angle of the most prominent directional texture on a surface.

Creating a surface with a known, specific direction of texture can be achieved with a sinusoid with a known direction of propagation. By defining a surface of the form

$$z(x, y) = \cos([A\pi x] + [B\pi y]),$$

the angle perpendicular to the direction of wave propagation, which gives the texture direction, is given by

$$S_{td} = \tan^{-1}(B/A) + 90^\circ$$

in degrees.

4.4. $S_{dr}$

As mentioned previously, $S_{dr}$ is calculated by finding the ratio of the evaluation area of the surface to the topographical area of the surface and subtracting 1. For the general case, the calculation of topographical area takes the form of a nonelementary integral. However, $S_{dr}$ is readily calculable for more specific surfaces, a simple example of which is a linear slope, defined by

$$z(x, y) = Ax + By.$$  \hspace{1cm} (32)

A schematic of this is shown in figure 10. The topographical area of a linear gradient surface can be found using the Pythagorean theorem to find the lengths of the two perpendicular sides of the rectangle of the form

$$\sqrt{x^2 + A^2x^2} = x\sqrt{1 + A^2},$$

$$\sqrt{y^2 + B^2y^2} = y\sqrt{1 + B^2},$$

and multiplying them together. The ratio of the topographical area to the evaluation area is then easily obtained, giving an expression for $S_{dr}$ of

$$S_{dr} = \sqrt{1 + A^2} \sqrt{1 + B^2} - 1.$$ \hspace{1cm} (35)

4.5. Assessment of third-party software

Similar to the assessments performed in sections 2.2 and 3.4, four surface texture parameter calculation software packages were used to showcase the validity of the parametric reference surfaces as a means to assess the performance of software. A total of five surfaces were created for parameters $S_{dr}$ and $Std$, substituting different variable values into the parametric surface
This approach demonstrates the flexibility of parametrically-defined surface/parameter reference pairs, as it enables a theoretically infinite variety of surfaces to be created that have mathematically traceable associated parameter values by adjusting the variables in the surface definition.

Figure 11 shows the results obtained by the software packages of the Sdr parameter for five surfaces obtained using a parametric surface function. Software C and D both show good agreement with the mathematical reference value for all surfaces, with software C obtaining normalised parameter values of the order of 1.000 000 05 for each surface. Software D shows more deviation than the other software packages for the fourth and fifth surface, which correspond to the surfaces with the lowest gradients. Conversely, software A and B performed poorly for the three surfaces that have the highest gradients and improve for the low gradient surfaces. In addition, software A and B obtain similar results for all five surfaces, suggesting a similar implementation of the Sdr calculation is performed for both software packages.

Figure 12 shows the results obtained for the Std parameter, for five surfaces obtained using a parametric surface function. It should be noted here that only software packages A and C offered the ability to calculate...
the \( \text{Std} \) parameter. Initial results, given in the upper graph, show a significant difference between the performance of software A and software C in their ability to obtain parameter values comparable to the mathematical reference value. Further investigation revealed the values obtained by software C were all less than those obtained by software A by around 90°, suggesting a difference in interpretation of the definition of the \( \text{Std} \) parameter given in ISO 25178-2 \[7\], specifically, relative to what axis should the most prominent angle of texture should be measured. Manually rotating the values obtained by software C gives the results shown in the lower graph, which show significantly better agreement with the mathematical reference values. With the corrections applied, software A is able to obtain a value for \( \text{Std} \) closer to the mathematical reference than software C for four of the five surfaces tested.

5. Performance assessment

The work presented here and in reference \[18\] introduce mathematically-defined reference pairs \[35\] for the performance assessment of surface texture parameter software. An important aspect of this process, however, is the delivery of the analytical surface to the discrete software under test.

All surface texture parameter software operates with a discrete surface height dataset as the primary
input. As the reference surfaces are defined analytically, it is therefore necessary to perform a discretisation operation in order to create a discrete dataset that can be used by the software. This is achieved by some method of sampling: extracting a finite number of height values by evaluating the surface expression at different points in x and y. By virtue of this finite sampling, however, information about the surface is inevitably lost, as the areas in between the sampling locations contain surface information that is not transferred to the discrete representation.

Due to the loss of information when moving from an analytical representation to a discrete representation, surface texture parameter calculation software will not be able to obtain the exact same results as the mathematical reference. The software is working with less information, and so will deviate from the mathematical reference values in scenarios where the information required has not been sampled and incorporated into the dataset. It is important, therefore, to attempt to account for this discretisation error and assess the software under test fairly, based on the information it is given.

This work introduces performance metrics that can be used to numerically assess the performance of surface texture parameter calculation software under test against mathematical reference values, while taking into account the discretisation error introduced from using discrete datasets for testing.

5.1. Agreement of significant figures
In this section, a method for the assessment of surface texture parameter software performance is introduced. This approach quantifies the number of significant figures given by the software-obtained value that agree with the mathematical reference value. This number is then given alongside the resolution of the dataset used to obtain the software value, giving a performance metric describing of the quality the software’s performance relative to the amount of discretisation error present. In addition, this performance metric allows for direct comparisons between third-party software.

Table 3 showcases the assessment of significant figure agreement for two analytically defined surfaces introduced in section 2. Here, the level of numerical precision that the software-obtained values agree with the reference values is presented for each parameter, allowing direct comparisons between software in a meaningful way that is directly applicable to the use of the software under test: to understand the extent to which the software gives an accurate result in comparison to a traceable reference. In addition, the total number of agreed significant figures is also calculated for each parameter set, providing a simple performance metric for the software under test for the reference surface as a whole. The results of the totals show software C gives the overall best agreement with the reference values for a dataset resolution of 700 × 700, narrowly improving upon software A for the ‘4th Bernoulli’ analytical surface.

It should be noted that software D, as discussed in section 2, is limited to 10 MB dataset file sizes, restricting the maximum resolution of the tested datasets to 700 × 700. In addition, software D often gave parameter values to a reduced precision in comparison to the other test software, restricting its ability to perform well in this assessment.

5.2. Variation due to sampling
A stage of the discretisation process that can affect the dataset and, therefore, the resulting parameter values is the sampling method used to obtain discrete height values from the analytical surface. The continuous surface can have different height values at very close x, y locations on the surface, and so small variations in the sampling locations can obtain different height values for use in the dataset. Additionally, averaging methods can be used that obtain single height values that describe areas of the analytical surface instead of specific sampling locations, which could also produce a different dataset for the same evaluation area.

There is no ‘correct’ sampling method, as different choices can be used to represent the height capture methods of different measurement processes. For example, point autofocus and contact stylus measurement instruments sample at discrete lateral intervals, whereas areal techniques, such as focus variation, obtain a full area image wherein each pixel represents a small area of the surface. In any case, sampling method variations can be addressed by detailing the method used when performing the software assessment. By explicitly defining the sampling method used, any further tests can be made using the same sampling methods and inter-software comparisons can still be made.

To highlight this variation, a second sampling implementation was used to compare to that used in all previous sections. For the previous sections, each

---

Table 3. Number of significant figures from the software-obtained parameter value results that agree with reference values obtained using the methods in sections 2 and 3.2 for 700 × 700 resolution datasets.

|                  | 4 term cosine | 4th Bernoulli |
|------------------|---------------|---------------|
|                  | A  | B  | C  | D  | A  | B  | C  | D  |
| Sq               | 5  | 3  | 5  | 5  | 10 | 2  | 10 | 2  |
| Sdk              | 4  | 2  | 4  | 4  | 8  | 2  | 10 | 5  |
| Ska              | 5  | 2  | 5  | 5  | 8  | 3  | 8  | 6  |
| Sp               | 5  | 5  | 5  | 5  | 8  | 7  | 7  | 3  |
| Sv               | 4  | 6  | 4  | 4  | 7  | 6  | 7  | 2  |
| Sz               | 4  | 8  | 4  | 4  | 8  | 7  | 8  | 3  |
| Sa               | 4  | 2  | 4  | 4  | 6  | 2  | 6  | 2  |
| Sdq              | 6  | 5  | 6  | 4  | 3  | 3  | 3  | 0  |
| Sdr              | 6  | 6  | 6  | 0  | 0  | 0  | 0  | 0  |
| Total            | 43 | 39 | 43 | 35 | 58 | 32 | 59 | 23 |

---

To highlight this variation, a second sampling implementation was used to compare to that used in all previous sections. For the previous sections, each parameter value results that agree with reference values obtained using the methods in sections 2 and 3.2 for 700 × 700 resolution datasets. The software under test: to understand the extent to which the software gives an accurate result in comparison to a traceable reference. In addition, the total number of agreed significant figures is also calculated for each parameter set, providing a simple performance metric for the software under test for the reference surface as a whole. The results of the totals show software C gives the overall best agreement with the mathematical reference values in scenarios where the information required has not been sampled and incorporated into the dataset. It is important, therefore, to attempt to account for this discretisation error and assess the software under test fairly, based on the information it is given.

This work introduces performance metrics that can be used to numerically assess the performance of surface texture parameter calculation software under test against mathematical reference values, while taking into account the discretisation error introduced from using discrete datasets for testing.

5.1. Agreement of significant figures
In this section, a method for the assessment of surface texture parameter software performance is introduced. This approach quantifies the number of significant figures given by the software-obtained value that agree with the mathematical reference value. This number is then given alongside the resolution of the dataset used to obtain the software value, giving a performance metric describing of the quality the software’s performance relative to the amount of discretisation error present. In addition, this performance metric allows for direct comparisons between third-party software.

Table 3 showcases the assessment of significant figure agreement for two analytically defined surfaces introduced in section 2. Here, the level of numerical precision that the software-obtained values agree with the reference values is presented for each parameter, allowing direct comparisons between software in a meaningful way that is directly applicable to the use of the software under test: to understand the extent to which the software gives an accurate result in comparison to a traceable reference. In addition, the total number of agreed significant figures is also calculated for each parameter set, providing a simple performance metric for the software under test for the reference surface as a whole. The results of the totals show software C gives the overall best agreement with the reference values for a dataset resolution of 700 × 700, narrowly improving upon software A for the ‘4th Bernoulli’ analytical surface.

It should be noted that software D, as discussed in section 2, is limited to 10 MB dataset file sizes, restricting the maximum resolution of the tested datasets to 700 × 700. In addition, software D often gave parameter values to a reduced precision in comparison to the other test software, restricting its ability to perform well in this assessment.

5.2. Variation due to sampling
A stage of the discretisation process that can affect the dataset and, therefore, the resulting parameter values is the sampling method used to obtain discrete height values from the analytical surface. The continuous surface can have different height values at very close x, y locations on the surface, and so small variations in the sampling locations can obtain different height values for use in the dataset. Additionally, averaging methods can be used that obtain single height values that describe areas of the analytical surface instead of specific sampling locations, which could also produce a different dataset for the same evaluation area.

There is no ‘correct’ sampling method, as different choices can be used to represent the height capture methods of different measurement processes. For example, point autofocus and contact stylus measurement instruments sample at discrete lateral intervals, whereas areal techniques, such as focus variation, obtain a full area image wherein each pixel represents a small area of the surface. In any case, sampling method variations can be addressed by detailing the method used when performing the software assessment. By explicitly defining the sampling method used, any further tests can be made using the same sampling methods and inter-software comparisons can still be made.

To highlight this variation, a second sampling implementation was used to compare to that used in all previous sections. For the previous sections, each parameter value results that agree with reference values obtained using the methods in sections 2 and 3.2 for 700 × 700 resolution datasets.
height value data point was treated as a pixel with a width equal to the spacing between $x$, $y$ sampling points. Therefore, to span the appropriate area, one sampling point was removed to account for the extra half spaces either side of the first and last data points in each direction. For example, for a 1000 × 1000 resolution surface spanning 10 mm by 10 mm, height values were sampled between 0 mm and 9.99 mm in each direction, accounting for the pixel widths which extend the surface size to between −0.005 mm and 9.995 mm. The second sampling implementation uses the same spacing between points, but includes an extra point on the end, giving a 1001 × 1001 resolution dataset with height values sampled between 0 mm and 10 mm. This method treats the dataset as a collection of discretely sampled height values with zero pixel width, and so still spans the same evaluation area as the previous implementation.

Table 4 gives the same significant figure analysis as performed in table 3, but for the second sampling method. Software B shows a significant improvement in its agreement with the reference values, while software A, C and D all show worse results. These performance metrics suggest this style of sampling is well interpreted by software B, whereas the other software packages are more suitable for the previous implementation. Again, neither implementation is ‘correct’, and clear specification of the sampling choices used when creating a dataset are necessary for proper assessment of metrology software.

### 6. Conclusion

The work presented in this paper applied the mathematical reference concept that was introduced in previous work [18] to field surface texture parameters. The previous work focussed on functional surface texture parameters, which are calculated from the material ratio curve, a relatively simple one-dimensional function. For field surface texture parameters, calculations are based, often directly, on the surface expression itself, which can be a complex two-dimensional function.

Methods were presented for calculating mathematical reference values for field surface texture parameters. This novel approach utilising analytical functions and parameter definitions to obtain mathematically traceable reference values for field surface texture parameters, and was showcased for a series of simple surfaces. Parametric methods were also defined to facilitate analytical calculation of individual parameters in the cases where solving a general case was not possible. In addition, machine precision numerical methods were used to show the application of the method to high complexity surfaces.

Comparisons were performed for four software packages, showing how the obtained parameter values can vary from the mathematical reference values. This showcases the value of this new method, and establishes a way to obtain traceability for surface texture parameter software. The combination of the methods developed in this paper delivers a comprehensive toolkit for the calculation of high-accuracy reference parameter values for analytical surface expressions. These reference pairs can be used to assess surface texture parameter calculation software and give both developers and end-users better insight into the performance of the software.

This paper presented metrics to assess the performance of surface texture parameter software whilst accounting for discretisation error introduced when creating a discrete dataset from an analytical surface. A technique was presented that calculated the number of significant figures in the software-obtained parameter values that agree with the reference value. This method gave a succinct performance metric for each parameter that enabled easy inter-software comparisons, and allowed for quick assessment of a software’s ability to agree with the reference value. The performance metrics for each parameter were also combined to assess the performance of the software as a whole, taking each parameter into account. In addition, the implementation of sampling on the created dataset, and hence the resulting parameter values, was investigated.

The importance of explicitly defining the sampling methods used when producing a dataset were emphasised, and an example comparison of two different sampling implementations were compared, showing a significant effect on different software packages.

The combination of the performance metrics and reference surface texture parameter values introduced in this paper provide a novel framework for the validation of surface texture parameter software that improves upon the current state of the art by introducing mathematical traceability and increased accuracy. By incorporating performance metrics, both developers and end-users of metrology software are able to assess and compare software results easily and accurately.
Acknowledgments

We would like to thank the Engineering and Physical Sciences Research Council (EPSRC Grants EP/M008983/1 and EP/1017933/1).

ORCID iDs

Luke Todhunter  https://orcid.org/0000-0003-4767-3088

References

[1] Leach R K 2013 Characterisation of Areal Surface Texture 1st edn (Berlin: Springer Berlin Heidelberg)
[2] Whitehouse D J 1994 Handbook of Surface Metrology (Bristol: Institute of Physics)
[3] Thomas T 2013 Roughness and function Surface Topography: Metrology and Properties 2 014001
[4] Pettersson U and Jacobson S 2003 Influence of surface texture on boundary lubricated sliding contacts Tribol. Int. 36 857–64
[5] Merlo A M 2003 The contribution of surface engineering to the product performance in the automotive industry Surf. Coat. Technol. 174–175 21–6
[6] ISO 4287 1998 Geometrical Product Specifications (GPS) – Surface Texture: Profile Method—Terms, definitions and Surface Texture Parameters (Geneva: International Organisation for Standardisation)
[7] ISO 25 178-2 2012 Geometrical Product Specifications (GPS) – Surface Texture: Areal Part 2: Terms, definitions and Surface Texture Parameters (Geneva: International Organisation for Standardisation)
[8] Blunt L, Jiang X, Leach R K, Harris P and Scott P 2008 The development of user-friendly software measurement standards for surface topography software assessment Wear 264 389–93
[9] Harris P M, Smith I M, Leach R K, Giusca C, Jiang X and Scott P 2012 Software measurement standards for areal surface texture parameters: I. Algorithms Meas. Sci. Technol. 23 105008
[10] Harris P M, Smith I M, Wang C, Giusca C and Leach R K 2012 Software measurement standards for areal surface texture parameters: II. Comparison of software Meas. Sci. Technol. 23 105009
[11] Bui S H, Renegar T B, Vorburger T V, Rajaj and Malburg M C 2004 Internet-based surface metrology algorithm testing system Wear 257 1213–8
[12] Bui S H and Vorburger T V 2007 Surface metrology algorithm testing system Precis. Eng. 31 218–25
[13] Jung L, Krüger-Sehm K, Spranger B and Koenders L 2004 Reference software for roughness analyses—features and results XI International Colloquium on Surfaces (Chemnitz, Germany) p 164
[14] ISO 5436–2 2012 Geometrical Product Specifications (GPS) – Surface Texture: Profile Method; Measurement Standards: II
[15] ISO 25 178–71 2012 Geometrical Product Specifications (GPS) – Surface Texture: Areal—Part 71: Software Measurement Standards (Geneva: International Organisation for Standardisation)
[16] Linares J, Goch G, Forbes A, Sprauer J, Clément A, Haertig F and Gao W 2018 Modelling and traceability for computationally-intensive precision engineering and metrology CIRP Ann. 67 815–38
[17] Todhunter L D, Leach R K, Lawes S D A and Blateyron F 2017 An analysis of type F2 software measurement standards for profile surface texture parameters Meas. Sci. Technol. 28 065017
[18] Todhunter L D, Leach R K, Lawes S, Harris P M and Blateyron F 2019 Mathematical approach to the validation of functional surface texture parameter software Surface Topography: Metrology and Properties 7 015020
[19] Todhunter L D, Senin N, Leach R K, Lawes S, Blateyron F and Harris P 2018 A programmable software framework for the generation of simulated surface topography XVIII International Conf. of the European Society for Precision Engineering and Nanotechnology, Venice (EUSPEN)
[20] Stein W and Jobner D 2005 Sage: system for algebra and geometry experimentation ACM SigSAM Bulletin 39 61–4
[21] Wolfram Research, Inc 2017 Mathematica, Version 11.1 (Champaign, Illinois: Wolfram Research, Inc)
[22] Maplesoft, a division of Waterloo Maple Inc 2017 Maple 2017 (Waterloo, Ontario: Maplesoft)
[23] Blateyron F 2013 The areal field parameters Characterisation of Areal Surface Texture ed R K Leach (Berlin: Springer Berlin Heidelberg) ch 2, pp 15–41 I ed
[24] Loehle C 2006 Global optimization using mathematica: a test of software tools Mathematika in Education and Research 11 39–52
[25] ISO 25 178-3 2012 Geometrical Product Specifications (GPS) – Surface Texture: Areal: III Specification Operators (Geneva: International Organisation for Standardisation) July
[26] Rosenlicht M 1968 Liouville’s theorem on functions with elementary integrals Pacific Journal of Mathematics 24 153–61
[27] Rosenlicht M 1972 Integration in finite terms The American Mathematical Monthly 79 963–72
[28] Todhunter L D, Leach R K, Lawes S D A and Blateyron F 2017 Industrial survey of ISO surface texture parameters CIRP J. Manuf. Sci. Technol. 19 84–92
[29] Brown C A et al 2018 Multiscale analyses and characterizations of surface topographies CIRP Ann. 67 839–62
[30] Driscoll T A, Hale N and Trefethen L N 2014 Chebfun Guide (Oxford: Pafnuty Publications)
[31] Townsend A and Trefethen L N 2013 An extension of Chebfun to two dimensions SIAM J. Sci. Comput. 35 C495–518
[32] Powell M D 1981 Approximation Theory and Methods (Cambridge: Cambridge University Press)
[33] Mason J C and Handscomb D C 2002 Chebyshev Polynomials (Boca Raton: Chapman and Hall/CRC)
[34] Trefethen L N 2007 Computing numerically with functions instead of numbers Mathematics in Computer Science 1 9–19
[35] Kok G J, Harris P M, Smith I M and Forbes A B 2016 Reference data sets for testing metrology software Metrologia 53 1091–100