An algorithm for automatic calibration

AN ALGORITHM FOR AUTOMATIC CALIBRATION

wpa 1144

Michiel van der Korst Jacko Koster

July 1, 1991

commissioned by : ing. K.G. Struik[1]

Eindhoven University of Technology,
Department of Mathematics and Computing Science,
P.O. Box 513, 5600 MB EINDHOVEN

[1]Department of Mechanical Engineering

Contents

1 THE CALIBRATION

This chapter contains a general description of the problem and introduces the algorithm developed in order to solve it.

1.1 Problem Definition

A *surface roughness tester* is used to determine the profile of a surface. The roughness tester measures this profile by dragging a pin along the surface and sampling the vertical position of this pin at fixed intervals . An example of the image of a surface profile determined in this manner is shown on page 30. This measurement information is used to determine surface properties such as the slope and height distribution and the Abbot curve.
But before the data can be interpreted the scaling factor of the roughness tester must be available. An added difficulty is the fact that the tester offers no less than eight possible accuracy positions, which all have their own scaling factor.
To keep the certainty factors up to date the surface roughness tester must be calibrated with a certain frequency. Until recent the scaling factors were computed from the calibration measurement data by hand. This method, however, was very slow and inaccurate. For this reason an algorithm has been developed, that will compute the scaling factors automatically from the measurement data. This algorithm which provides a quick and easy way to compute the scaling factors is described in this report.
In this section the principle of determining the scaling factor from measurement data will be described. Then the old manual method will be discussed. Finally the method for automating this task will be introduced briefly.

1.2 Scaling Factor

The scaling factor for a given accuracy position is determined by using a calibration object. Such a calibration object has a flat surface and a number of smooth grooves whose depth are certified in a corresponding calibration certificate. In figure 1 the surface of the object used in this project is presented.
Once the depths of the grooves of measured object profile are determined, the scaling factor for the current accuracy can be computed :

$$\mathbf{f}_{scale} = \frac{\mathbf{d}_{real}}{\mathbf{d}_{meas}}$$

- $\mathbf{f}_{scale}$ the scaling factor
- $\mathbf{d}_{real}$ depth of the groove as stated in calibration certificate
- $\mathbf{d}_{meas}$ depth of the groove as determined from measurement data

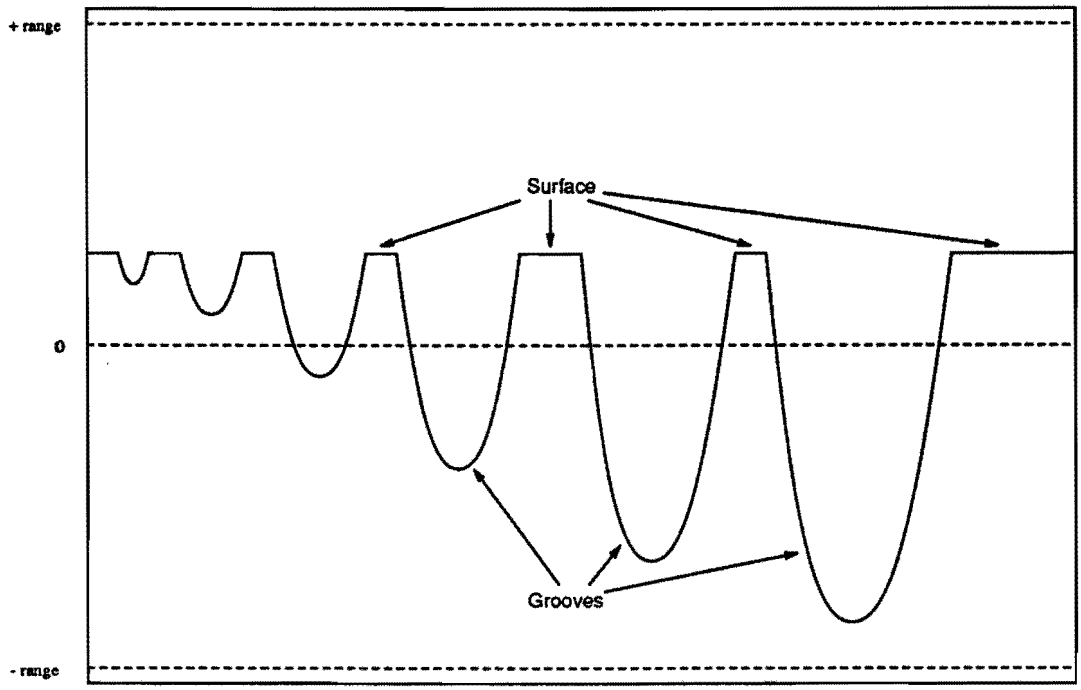

Figure 1: surface of calibration object EN 629

By using this formula the calibration boils down to the problem of determining the depth of the grooves in the measurement data.

1.3 Manual Calibration

The current method for determining the depth of the grooves uses a printout of the profile, a pencil and a ruler. First the surface is interpolated over the groove. Then the perpendicular distance between this surface and the point farthest away from this line is determined with a ruler. This distance is exactly the depth of the groove. The roughness tester does, however, introduce noise in the samples. To smooth the effect of the noise, tangents must be constructed along the groove before determining the distance.
The manual method described above requires some time because of the graphical construction of the tangents and the determination of the longest distance to the surface. The effort this requires results in a low frequency of calibration. An automatic computation of the scaling factors from the measured samples will stimulate the users to increase this frequency. The automatic algorithm will probably also be able to give more accurate information.

1.4 Automatic Calibration

The scope of this report is restricted to interpreting the measurement data. In the course of the project it became clear that the method for acquiring the data also requires improvements. This topic will be discussed shortly in the conclusion.
The algorithm we want to develop will have to compute a scaling factor from a sequence of sampled points. The problem at hand is two-fold :

- determine which sample data belong to a groove
- determine the depth of the detected grooves

To solve the problem we make two assumptions regarding the profile of the calibration object.

1. the surface of the calibration object outside the grooves can be approximated by a line model of the form :

$$y = A + B \times x$$

2. the shape of each groove can be approximated by a parabole model of the form :

$$y = A + B \times x + C \times x^2$$

These simple models proved to work well with respect to this problem. The advantage of these models is that fast and simple algorithms can be used.
In the next chapter the formulas for the line and parabole approximations are derived. In chapter 3 the complete algorithm for determining the scaling factor will be discussed.

2 LEAST SQUARES FITTING

This section describes two methods used by the algorithm that is developed in the section hereafter. Each method determines the best curve (respectively a line and a parabole) that fits an expected relation between a number of measurements. Instead of treating this problem graphically or manually we will do it here mathematically, by means of the principle of maximum likelihood.

First we consider the case where the expected relation between the measurements is *linear*. This means that we have to find the best estimates for the constants A and B based on the measured data, in order to find a linear equation of the form $y = A + Bx$. The analytical method of finding the best straight line to fit a series of experimental points is well known as *linear regression*, or the *least squares fit for a line (line fit)* and it is one the main subjects of this chapter. After having determined the line that bests fits the data, we investigate the uncertainties in our estimates for A and B due to possible uncertainties in the measurements.
Next we consider the case when we expect the behavior between the measurements to be *quadratic*. Here we have to find the best estimates for the three constants A, B and C in order to calculate the best fitting quadratic polynomial $y = A + Bx + Cx^2$. This is called a *parabole fit*. Again we look at the influence of possible uncertainties in the measurements on the calculated results.

2.1 Line Fit

Throughout this section the following general equation for a straight line will be used:

$$y = A + Bx,$$

where A and B are the constants that we are looking for. It represents the line with slope B which intersects the y-axis at $y = A$.
The measurements are represented by N pairs $(x_1,y_1),\ldots,(x_N,y_N)$, where N is the number of measurements and we assume $N \geq 2$.

2.1.1 the computation of A and B

In order to find the line that bests fits the measurements we compute for each measurement the difference between the y-coordinate of that measurement and the corresponding value of the line. For each measurement i, $1 \leq i \leq N$, this is $y_i - A - Bx_i$. For the line that bests fits the measurements holds that the sum of all these differences is minimal. To avoid difficulties with possible negative values we square all differences. (This is why the method is known as the least squares fitting.)

So, if we define the sum S as

$$S(A,B) = \sum_{i=1}^{N}(y_i - A - Bx_i)^2,$$

then after differentiating S to A and B we have to set the derivatives to zero, giving

$$\frac{\delta S}{\delta A} = -2\sum_{i=1}^{N}(y_i - A - Bx_i) = 0$$

and

$$\frac{\delta S}{\delta B} = -2\sum_{i=1}^{N} x_i(y_i - A - Bx_i) = 0.$$

These two equations can be rewritten as (from now on we omit the limits $i = 1$ to N from the summation symbols $\sum$)

$$\begin{aligned} AN &+B\sum x_i &&= \sum y_i \\ A\sum x_i &+B\sum x_i^2 &&= \sum x_i y_i \end{aligned}$$

This set of equations can easily be solved. This leads to

$$A = \frac{(\sum x_i^2)(\sum y_i) - (\sum x_i)(\sum x_i y_i)}{\Delta}, \tag{1}$$

$$B = \frac{N(\sum x_i y_i) - (\sum x_i)(\sum y_i)}{\Delta}, \tag{2}$$

where Δ is defined as

$$\Delta = N(\sum x_i^2) - (\sum x_i)^2. \tag{3}$$

Finally we show how the equations 1, 2 and 3 are computed in the algorithm in section 3.2. The algorithm computes the model of the best fitting line iteratively. This means that new sample points are added to the model one after another and every time a new sample point is added, the model is adjusted.
Looking at the definitions of A, B and Δ, we see that they contain summation signs $\sum$. At first sight this would mean that every time a new measurement is added, these sums have to be recomputed. However, this can be done more efficiently, by using results from the previous computation of the model. To be able to use these previous results we introduce a number of variables which contain the values of the summations.
Suppose we have computed a model for n sample points $(x_1,y_1),\ldots,(x_n,y_n)$ where $n \geq 2$. For determining this model the following sums have been computed: X, Y, XX, XY which are defined as

$$\begin{aligned} X &= \textstyle\sum_{i=1}^{n} x_i & Y &= \textstyle\sum_{i=1}^{n} y_i \\ XX &= \textstyle\sum_{i=1}^{n} x_i x_i & XY &= \textstyle\sum_{i=1}^{n} x_i y_i \end{aligned}$$

For adding a new sample point (x_{n+1},y_{n+1}) the following changes must be made to adjust the model

$X = X + x_{n+1}$;
$Y = Y + y_{n+1}$;
$XX = XX + x_{n+1}^2$;
$XY = XY + x_{n+1}y_{n+1}$;
compute Δ (*using* 3);
compute A (*using* 1);
compute B (*using* 2);

Now holds

$$X = \sum_{i=1}^{n+1} x_i \qquad Y = \sum_{i=1}^{n+1} y_i$$
$$XX = \sum_{i=1}^{n+1} x_i x_i \quad XY = \sum_{i=1}^{n+1} x_i y_i$$

and these variables can be used later for a possible further adjustment of the model.

2.1.2 the uncertainties in A and B

In this section we will assume that the uncertainties in y all have the same magnitude. Moreover we will suppose that, although the measurements of y suffer some uncertainty, the uncertainty in the measurements of x is negligible.

A commonly used measure to indicate whether the analytical result satisfies the experiments, is the mean (squared) difference between the y-coordinates of the measurements and the corresponding value of the calculated line. This is called the *variance of* y, σ_y^2. Formally:

$$\sigma_y^2 = \frac{1}{N}\sum(y_i - A - Bx_i)^2$$

In practice however, a slightly different definition of σ_y^2 is often used:

$$\sigma_y^2 = \frac{1}{N-2}\sum(y_i - A - Bx_i)^2 \tag{4}$$

We will not attempt to justify the factor $N - 2$ here, but it can be argued that the latter definition is a more realistic estimate for the variance of y [2]. This new definition changes the value of σ_y^2 only if N is small. When the number of measurements (i.e., N) is very large, this modification of σ_y^2 has no significant effect and could be omitted. Notice that definition 4 is only meaningful if $N > 2$.

The algorithm that is presented in the next section determines the surface iteratively. Every time when a new sample point is added to the model computed so far, the variance of y is calculated. (This is done to check whether the sample point is positioned in a groove or not.) However, using equation 4 means computing the sum $\sum$ every time a point is added to the model. This can be avoided. Therefore we rewrite σ_y^2 as follows:

$$\sigma_y^2 = \frac{1}{N-2}\left(\sum y_i^2 - 2A\sum y_i - 2B\sum x_i y_i + NA^2 + 2AB\sum x_i + B^2\sum x_i^2\right) \tag{5}$$

The summations occurring in this formula can all be declared in variables (the same as introduced in section 2.1.1). In every new iteration step of the algorithm all these variables can easily be updated. Now all successive computations of σ_y^2 together require only a linear amount of time while with equation 4 this will be quadratic in the number of measurements (N). Using the last definition speeds up the algorithm considerably.
Using definition 5 and some simple calculus we can compute the variance of A and B. This computation of σ_A^2 and σ_B^2 is given in Appendix C. Here we only present the result of that computation. The variances of A and B are:

$$\sigma_A^2 = \frac{\sum x_i^2}{\Delta}\sigma_y^2 \tag{6}$$

and

$$\sigma_B^2 = \frac{N}{\Delta}\sigma_y^2, \tag{7}$$

where Δ is given by equation 3.

2.1.3 an example

Consider the following four measured points:

$$(x_1, y_1) = (1,1), \quad (x_3, y_3) = (3,2),$$
$$(x_2, y_2) = (2,2), \quad (x_4, y_4) = (4,2).$$

Applying equations 1, 2 and 3 to the first three points gives (N=3)

$$\Delta = 6, \quad A = \frac{2}{3}, \quad B = \frac{1}{2}.$$

So, the best fitting line becomes

$$y = \frac{2}{3} + \frac{1}{2}x,$$

and equations 5, 6 and 7 give

$$\sigma_y^2 = \frac{1}{6}, \quad \sigma_A^2 = \frac{7}{18}, \quad \sigma_B^2 = \frac{1}{12}.$$

Doing the same for all four points results in (N=4)

$$\Delta = 20, \quad A = 1, \quad B = \frac{3}{10},$$

and the corresponding line is:

$$y = 1 + \frac{3}{10}x.$$

Equations 5, 6 and 7 now give

$$\sigma_y^2 = 1, \quad \sigma_A^2 = \frac{3}{2}, \quad \sigma_B^2 = \frac{1}{5}.$$

2.2 Parabole Fit

A general equation of a polynomial that is quadratic (a parabole) is:

$$y = A + Bx + Cx^2, \tag{8}$$

where A, B and C are the constants we have to determine.
Remember that the measured data is represented by N pairs $(x_1,y_1),\ldots,(x_N,y_N)$, $N \geq 2$.

2.2.1 the computation of A, B and C

The computations of the three constants is in first instance analogous to those of the line fit in section 2.1.1. Again we define a sum S, this time it has three parameters:

$$S(A,B,C) = \sum_{i=1}^{N}(y_i - A - Bx_i - Cx_i^2)^2$$

and we differentiate S to A, B and C successively, giving:

$$\frac{\delta S}{\delta A} = -2\sum_{i=1}^{N}(y_i - A - Bx_i - Cx_i^2) = 0,$$

$$\frac{\delta S}{\delta B} = -2\sum_{i=1}^{N} x_i(y_i - A - Bx_i - Cx_i^2) = 0,$$

$$\frac{\delta S}{\delta C} = -2\sum_{i=1}^{N} x_i^2(y_i - A - Bx_i - Cx_i^2) = 0.$$

From these three equations the following set of equations can be obtained:

$$\begin{array}{rrrr} AN & +B\sum x_i & +C\sum x_i^2 & = \sum y_i \\ A\sum x_i & +B\sum x_i^2 & +C\sum x_i^3 & = \sum x_i y_i \\ A\sum x_i^2 & +B\sum x_i^3 & +C\sum x_i^4 & = \sum x_i^2 y_i \end{array}$$

Solving this set is somewhat more difficult than in the previous section. However, using linear algebra (Cramer 's Rule) leads to the following solution:

$$A = D \times n_1,\ B = D \times n_2,\ C = D \times n_3, \tag{9}$$

where

$$D = \frac{\sum y_i}{n_1 N + n_2 \sum x_i + n_3 \sum x_i^2} \tag{10}$$

and (n_1, n_2, n_3) is the outerproduct of (a_1, a_2, a_3) and (b_1, b_2, b_3):

$$\begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{pmatrix} \tag{11}$$

and

$$\begin{aligned} a_1 &= \sum x_i \sum y_i - N \sum x_i y_i, & b_1 &= \sum x_i^2 \sum y_i - N \sum x_i^2 y_i, \\ a_2 &= \sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i, & b_2 &= \sum x_i^3 \sum y_i - \sum x_i \sum x_i^2 y_i, \\ a_3 &= \sum x_i^3 \sum y_i - \sum x_i^2 \sum x_i y_i, & b_3 &= \sum x_i^4 \sum y_i - \sum x_i^2 \sum x_i^2 y_i \end{aligned}$$

The definitions in equation 9 are used in the algorithm in section 3.2 for determining the model of the best fitting parabole to a number of measurements. Again we introduce a number of variables to calculate the values of the summations occurring in the definitions of A, B and C.
The sample points are $(x_1,y_1),\ldots,(x_N,y_N)$, where $N \geq 2$. Before we can compute the model seven sums have to be calculated: X, Y, XX, XY, XXX, XXY, $XXXX$ which are

$$\begin{aligned} X &= \textstyle\sum_{i=1}^{N} x_i & Y &= \textstyle\sum_{i=1}^{N} y_i \\ XX &= \textstyle\sum_{i=1}^{N} x_i^2 & XY &= \textstyle\sum_{i=1}^{N} x_i y_i \\ XXX &= \textstyle\sum_{i=1}^{N} x_i^3 & XXY &= \textstyle\sum_{i=1}^{N} x_i^2 y_i \\ XXXX &= \textstyle\sum_{i=1}^{N} x_i^4 \end{aligned}$$

These sums and the parabole model are then computed in the following way:

```
X := 0; Y := 0;
XX := 0; XY := 0;
XXX := 0; XXY := 0;
XXXX := 0;
for i := 1 to N
  do X := X + x_i;
     Y := Y + y_i;
     XX := XX + x_i^2;
     XY := XY + x_i y_i;
     XXX := XXX + x_i^3;
     XXY := XXY + x_i^2 y_i;
     XXXX := XXXX + x_i^4;
  od;
compute n1, n2 and n3 (using 11);
compute D (using 10);
compute A, B and C (using 9);
```

2.2.2 the uncertainties in A, B and C

In [1] a clearly structured overview is given of *multiple regression.* This is a more general form of linear regression. In this section we will apply a reduced form of this method for determining the variance of the three constants A, B and C.

Multiple regression is used in cases where more than two variables are to be considered. An example of such a problem is when one variable, y, depends linearly on two others, x and z:

$$y = A + Bx + Cz \tag{12}$$

This problem can be analyzed by a straightforward generalization of linear regression. For a number of measurements (x_i,y_i,z_i), $1 \leq i \leq N$, the principle of maximum likelihood can be used again to construct a set of equations that can be solved for A, B and C. In [1] this is worked out in detail. Moreover, a way how to compute the variances (and co-variances) of the results is given. Before we can use this we have to translate our problem to that of multiple regression. This is quite simple: comparing equations 8 and 12 shows that it suffices to merely substituting variable z by x^2. So our problem is a special case of multiple regression. Hence, we can use the method presented in [1].

The variance of y is now defined as:

$$\sigma_y^2 = \frac{1}{N}\sum(y_i - A - Bx_i - Cx_i^2)^2 \tag{13}$$

(Again it could be argued that the factor N must be replaced by $N - 2$.)
With (an adapted form of) the method described in [1] it can be proven that the variances of A, B and C are:

$$\sigma_A^2 = \frac{\sum x_i^2 \sum x_i^4 - (\sum x_i^3)^2}{D}\sigma_y^2,$$

$$\sigma_B^2 = \frac{N\sum x_i^4 - (\sum x_i^2)^2}{D}\sigma_y^2,$$

$$\sigma_C^2 = \frac{N\sum x_i^2 - (\sum x_i)^2}{D}\sigma_y^2$$

where D is the determinant of the matrix

$$\begin{pmatrix} N & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{pmatrix}$$

2.2.3 an example

Consider the following five measured points:

$$\begin{array}{ll} (x_1, y_1) = (0,2), & (x_4, y_4) = (3,1) \\ (x_2, y_2) = (1,1), & (x_5, y_5) = (4,2) \\ (x_3, y_3) = (2,0). & \end{array}$$

Applying equations 9 gives (N=3)

$$A = \frac{72}{35}, \quad B = -\frac{12}{7}, \quad C = \frac{3}{7},$$

so, the best fitting parabole is

$$y = \frac{72}{35} - \frac{12}{7}x + \frac{3}{7}x^2,$$

and equation 13 (with factor N-2) gives

$$\sigma_y^2 = \frac{56}{735}$$

3 THE ALGORITHM

In this chapter the algorithm will be discussed in two steps. First the problem will be specified and the different tasks of the program are identified. The second part of the chapter will be used to discuss the implementation.

3.1 Informal Specification

The algorithm to be designed will compute the scaling factor from the measurement data by using the depths of the grooves. These depths can be computed in two runs. In the first run a line model for the surface is determined and the measured points are classified in order to identify the grooves. In the second run a parabole fit is performed on the points lying in a groove. From the resulting parabole model the deepest point of this groove can be computed. As was discussed in chapter 1, the scaling factor can be computed from the depth of a groove in a measured profile. Finally the result will have to be evaluated and written to permanent storage.
All this results in a sequence of four tasks :

1. read measurement data from disk
2. classify measured points to identify grooves
3. determine depth of selected grooves
4. analyze results and write these to disk

The following sections each of these tasks will be discussed in more detail.

3.1.1 reading data

In order to compute the scaling factor information is needed from the following files :

- the measurement information file
- the measurement data file
- the scaling factor file
- the calibration object file

The *measurement information file* contains general information on a specific measurement. Examples of items specified in this file are :

- number of sampled points
- accuracy position during measurement

- name pick-up element used
- name calibration object used
- date of measurement
- etc. etc.

This information is required before any other information can be retrieved.
The *measurement data file* contains the sample points of the measurement that represent the profile. The points form a sequence $(p_1, p_2, ..., p_n)$, where point p_i is a tuple of the form (x_i, y_i), which is sorted by increasing x-value with the relation :

$$x_{i+1} := x_i + \textit{resolutionstep}$$

This resolution step is a measurement parameter determining the horizontal distance between two consecutive sample points. Before these points are loaded from disk a memory position must be allocated for each point. The number of positions to allocate is taken from the measurement information file. When the memory is allocated the points are all read from disk.
The *scaling factor file* contains the scaling factors of all accuracy positions for a particular pick-up element. The name of this pick-up element is used as the identifier for the scaling factor file. The accuracy position determines which factor in the file must be read. This scaling factor is used in two ways :

1. as the range. The range of the measurement is **[−scalingfactor, .., scalingfactor]**. Using these limits on the measurement range we can detect whether the measurement runs out of the scale.

2. for comparison. When a new scaling factor is determined from the measurement data we need to know the previous value of the scaling factor. The old and the new version are checked on consistency. If these factors show a significant discrepancy the scaling factor file must be updated.

The *calibration object file* contains the certified depths of the calibration object. This information is copied from the calibration certificate of the object used. The name of the object is used as the filename. The real depth of the grooves is needed for the computation of the scaling factor as was shown in chapter 1.
All these files have a strict format convention which is described in appendix B. If any of the files either does not conform to these conventions or is not available the execution of the program will be aborted. It is not possible to perform the computation using incomplete information. Of course, the user will have to be notified on the nature of the problem, i.e. which file created what problem.

3.1.2 classification of the points

Once all measured points are loaded, the first run can be executed. In this run a line model for the surface is approximated and the locations of the grooves are identified. The line model that is being approximated is the tangent on top of the profile. This tangent will touch the profile at the flat surface but not at the grooves. By testing whether a point is in the neighbourhood of this tangent it can be identified as lying on the surface or in a groove.
To get an initial approximation of the tangent the first number of points are taken to be representative for this tangent. The line fit on these points provide the initial model $(\bar{A}, \bar{B})$ for the tangent. All points on the flat part of the profile will have to fall within this model. The only change to this model will be a refinement by integrating new points in the model.
Once an initial model has been determined the loop can be initiated that will process all points in sequence.
At a certain point p_i the model will be equal to (A_i, B_i). (For p_{init}, the first point after the initial points : $A_{init} = \bar{A}$ and $B_{init} = \bar{B}$). Using this intermediate model for point p_i, the next point p_{i+1} can be classified :

- $y_{i+1} - A_i - B_i \times x_{i+1} < 2 \times \sigma_i$
  This means p_{i+1} lies inside the model (A_i, B_i) with standard deviation σ_i.

- $y_{i+1} - A_i - B_i \times x_{i+1} \geq 2 \times \sigma_i$
  This means p_{i+1} lies outside the model (A_i, B_i) with standard deviation σ_i.

If the point p_{i+1} lies within the model, the point is incorporated into the model in the way described in section 2.1.1. This results in a new model (A_{i+1}, B_{i+1}) with standard deviation σ_{i+1}.
If, however, the point does not fall inside the line model, this model will not be updated. The new model (A_{i+1}, B_{i+1}) will be equal to the previous model (A_i, B_i).
When a number of consecutive points lie outside the line model it must be decided whether these belong to a groove that can be used for calibration. In figure 2 the result of a case analysis is presented (the dotted lines represent the measurement range). These cases identify all the possible situations when a sequence of one or more points fall outside the current line model.

1. This is the ideal case, where a sequence of points falling outside the current line model represents one of the grooves. This groove falls completely within the measurement range and can be used to compute a scaling factor. This sequence is registered as a *valid* groove. If some of the points fall outside the range the form of the groove would be influenced. When this happens the parabole fit will not be reliable and the result can not be used. Such a groove will be registered as being *invalid.*

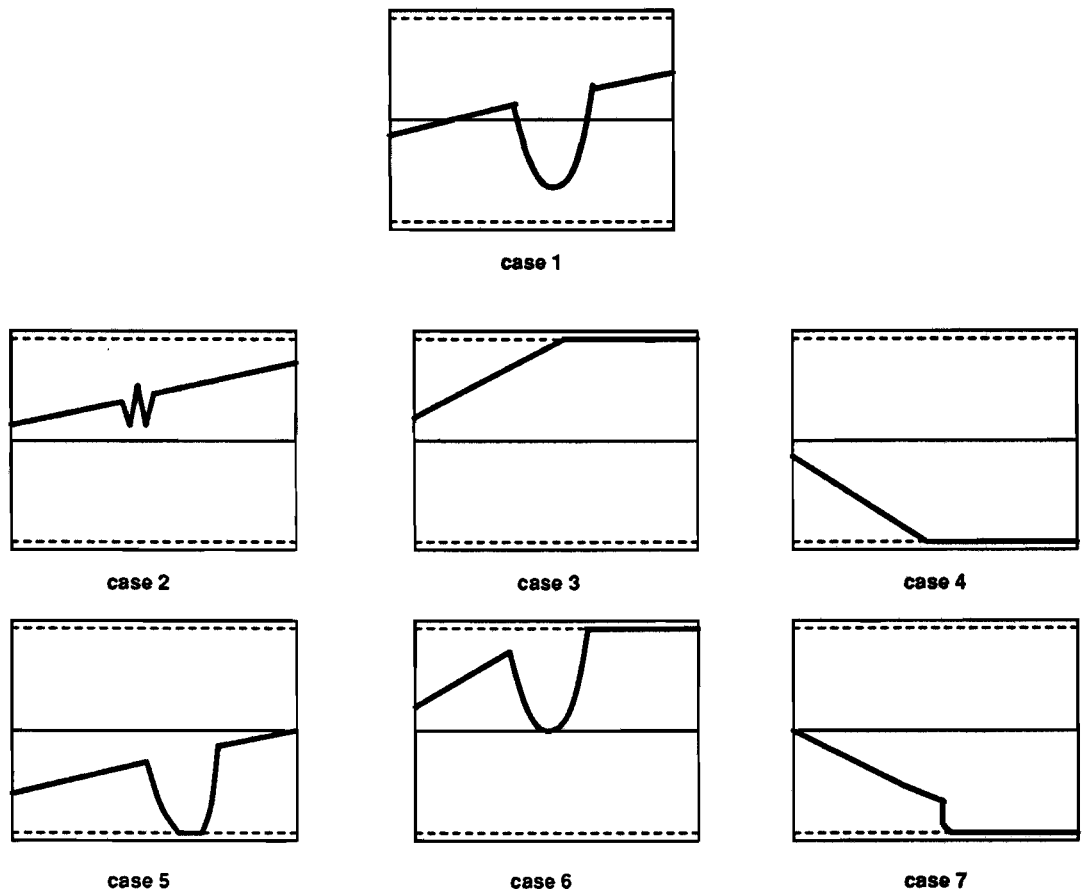

Figure 2: cases of point sequences falling outside the current line model

2. This case represents a disturbance, which can have two possible causes :

   - a disturbance in the object surface
   - a disturbance in the measurement

   In both cases these points must be ignored. They can neither be used for updating the line model nor be identified as grooves. The problem is to discern these disturbances from regular grooves. The assumption used to distinguish between a groove and a disturbance is that the disturbances will be smaller than the smallest groove in the calibration object. In the program a threshold is defined that indicates the minimum number of points a disturbance must contain to be classified as a groove. In practice this works well. If this method provides problems the threshold can be set to a higher number of points.However, if the largest disturbance is larger than the smallest groove

this object can not be used anymore. For this to happen the object must, however, be very badly damaged.

3. In this case the points will not follow the model because the range has been reached. Once the upper range has been reached the classification can be stopped because we can not discern the surface points from the groove points.

4. Idem dito, only the lower bound of the range has been reached.

5. In this case the lower bound of the range has been reached by the lower part of a groove. Because the shape of the groove is distorted by the cut-off at the range value, the parabole fit can not be performed. This groove will have to be marked as *invalid.*

6. This case is a variant of case 3. In this case the classification can be stopped because the distinction between points on the surface and points on a groove has disappeared. The groove itself is not valid and no more grooves can be detected.

7. Identical to case 6.

Once all points have been classified a list of grooves is produced. From this list one or more grooves will have to be chosen to compute the scaling factor. When more than one groove is used the results can be averaged resulting in more accurate results; presuming the results from the different grooves are independent. Since the largest grooves showed to produce the lowest standard deviation for the scaling factor the largest two are to the next step. If only one groove is available the process will also be continued. The user will be notified of the fact that the result may be less reliable.

3.1.3 determining the depth

In this phase the depth of a groove detected in the previous activity is determined. First a coordinate transformation is performed on the points to position the surface, approximated by the line model (A, B) along the x-axis (with a translation along the vector $(0, -A)$ and a rotation with the angle $\text{atan}(B)$ around the origin). When a parabole fit is performed on this transformed groove, the depth of the groove will be equal to the y-coordinate of the extreme (negative) value. Next the parabole fit is performed on the sequence of points as was described in section 2.2. In practice this parabole fit approaches the shape of the groove sufficiently , as can be concluded from the standard deviation of the fit. This standard deviation is smaller than the accuracy indicated in the calibration certificate.
When another calibration object is used, it must be verified that the parabole fit will still be a good approximation of the groove.

Once a parabole model has been developed then the extreme y-value can be computed easily by using the analytic solution.
For a parabole of the form :

$$y = A + B \times x + C \times x^2$$

the solution for the extreme y-value is :

$$y = A - \frac{B^2}{4 \times C}$$

The resulting absolute value of y is equal to the distance between the x-axis and the deepest point in the groove. This again is approximately equal to the depth the groove. Note that the uncertainty of this depth is equal to σ_y of the parabole approximation (A, B, C).
From the depth of a groove the scaling factor is computed by the formula :

$$\mathrm{f}_{scale} = \frac{\mathrm{d}_{real}}{\mathrm{d}_{meas}}$$

Once this scaling factor is determined the standard deviation of this factor must be computed. The rules for propagation of uncertainties in [2] can be used for the propagation of the standard deviation, since we define the uncertainty in the results to be equal to twice the standard deviation. The standard deviation σ_{scale} for f_{scale} can now be derived as follows:

σ_{scale}

$=$ { propagate σ like uncertainty }

$$\sigma\left(\frac{\mathrm{d}_{real}}{\mathrm{d}_{meas}}\right)$$

$=$ { propagation rule }

$$\mathrm{d}_{real} \times \sigma\left(\frac{1}{\mathrm{d}_{meas}}\right)$$

$=$ { propagation rule }

$$\mathrm{d}_{real} \times \left(\frac{1}{\mathrm{d}_{meas}}\right)^2 \times \sigma_{meas}$$

$=$ { definition f_{scale} }

$$\frac{\mathbf{f}_{scale}}{\mathbf{d}_{meas}} \times \sigma_{meas}$$

The standard deviation for the measured depth, σ_{meas}, is equal to the sum of the deviations of the line fit and the parabole fit :

$$\sigma_{meas} = \sigma_y^{line} + \sigma_y^{par}$$

3.1.4 analyzing and saving the results

This final phase of the program is used to show the results and give advice on the desired action to be performed by the user. On the basis of the computed scaling factor ($\mathbf{f}_{scale}$), the corresponding standard deviation (σ_y) and the old scaling factor from the file ($\mathbf{f}_{prev}$) the following decision rules are executed :

- $\sigma_{scale} > 0.1$: The standard deviation of the final result is too large, so the result is not very reliable. Advise to rerun the measurement. The factor 0.1 is determined by practice.
- $|\mathbf{f}_{scale} - \mathbf{f}_{prev}| < 2 \times \sigma_{scale}$: The old scaling factor $\mathbf{f}_{prev}$ is in accordance with the computed scaling factor, with a probability of 95%. Advise to keep the current scaling factor.
- $|\mathbf{f}_{scale} - \mathbf{f}_{prev}| \geq 2 \times \sigma_{scale}$: The old scaling factor $\mathbf{f}_{prev}$ is not in accordance with the computed scaling factor, with a probability of 95%. Advise to change the current scaling factor.

The results will be displayed on screen and also saved in the **.c** file.

3.2 Program Design

Starting from the "informal" specification in the previous section a program was constructed. In this section the interesting parts of the program will be presented and explained. The interesting parts are steps 2 and 3 in the task sequence presented in section 3.1.

3.2.1 point classification

As described earlier this task will process all sample points , in order to compute the line model *(A,B)* with the standard deviation σ_y. The procedure *update_linemodel* uses the input point p to update the line model *(A,B)* with variance σ_y^2. Because the incremental method described in section 2.1.1 is used, five additional parameters (X, XX, Y, YY and XY) are used. This way the summations appearing in the original formula for A, B and σ_y^2 (equations 1, 2 and 3) do not have to be recalculated when a point is added.

Procedure 1 *update_linemodel(p : Point)*

begin {*# points* $\geq$ *2, else error in division* }
$\#points := \#points + 1;$
$X := X + p.x; XX := XX + p.x^2;$
$Y := Y + p.y; YY := YY + p.y^2;$
$XY := XY + p.x \times p.y;$

$\Delta := \#points \times XX - X^2;$

$A := \frac{XX \times Y - X \times XY}{\Delta};$

$B := \frac{\#points \times XY - X \times Y}{\Delta};$

$\sigma_y^2 := \frac{YY - 2 \times A \times Y + 2 \times B \times XY + \#points \times A^2 - 2 \times A \times B \times X + B^2 \times XX}{\#points - 2}$
end;

All variables should have an initial value of 0.
Once this algorithm for updating the line model is determined the complete classification algorithm can be developed.
This procedure processes the sample points one by one, classifying them as either points on the line model or points outside the line model.
Once a sequence of points outside the line model is terminated by at least one point on the line model, this sequence is classified as a groove or as a disturbance. If this sequence is a groove it will be classified either as a valid or an invalid groove.
Finally, a list of intervals is returned, where each interval represents a groove. Each interval has got a corresponding boolean *valid*, which is **true** when that groove is valid and **false** otherwise.

Procedure 2 *classify_points()*

```
{initial values}
inside_groove := false;
# grooves := 0;
# points := 0;
point := first_point;
while (|point.y| < range) ∧ (#points ≤ #initialpoints)
{ determine initial model }
do
        update_linemodel(point);
        point := point.next;
od;
{ now the initial line model, (Ā, B̄), is determined }
while (point.y < range) ∧ (#points ≤ #samplepoints)
        do
```

```
if distance(point,line model) < (2 × σ_y)
then
{ point inside line model }
        update_linemodel(point);
        if inside_groove
        then
                { end of interval reached }
                groove[# grooves].end := point
                if dist(grooves[# grooves].begin, grooves[# grooves].end) ≥ threshold
                then # grooves := # grooves + 1;
                { groove large enough, increment counter }
                else # grooves := # grooves ;
                { groove too small, ignore this disturbance }
                inside_groove := false;
        fi;
else
{ point not inside line model }
        if not inside_groove
        then
                { start of interval reached }
                inside_groove := true;
                groove[# grooves].begin := point;
                groove[# grooves].valid := true;
                { groove initially valid }
        fi;
        if not (point.y > -range)
        then    groove[# grooves].valid := false
                { y-value drops out of range, current groove invalid}
        fi;
fi;
point := point.next;
od
```

When this procedure stops it will have determined the line model (A, B), σ_y and a sequence of *# grooves* records of the form :

$$(\texttt{begin}\ (point), \texttt{end}\ (point), \texttt{valid}\ (boolean))$$

From this set the two largest grooves are selected to determine the scaling factor. When only one valid groove is available this one will be used and the user will be notified of this fact.
The computation of the parabole fit is discussed in section 2.2.1. The computation of the depth and finally the scaling factor from the resulting parabole model is discussed section 3.1.3.

4 CONCLUSION

In this article an algorithm for automatic calibration of a roughness measurer was discussed. This algorithm interprets measurement data and computes the resulting correction factor and the corresponding standard deviation, if possible.
The implementation of this algorithm resulted in a speedup and produced more accurate results. Currently the roughness measurer is being calibrated before every experiment, which results in a high reliability of the experiments.
The calibration process can, however, still be improved. The communication between the computer and the measurement device still requires a lot of intervention by the user. The user has to provide the computer with information on the current accuracy position of the measurer. This can result in inconsistent measurement information. Another example is the synchronization at the start of the measurement run, which seems to require athletic skill of the experimenter.
This communication between the roughness measurer and the computer can be the focus of a future project.

References

[1] *syllabus Regressie Analyse (2S100).*
fac. Wiskunde en Informatica (T.U.E). 1984

[2] J.R Taylor, *An Introduction to Error Analysis*
University Science Books, Mill Valley. 1982

[3] *Kalibrierschein 2599 PTB 89*
Physikalisch-Technische Bundesanstalt, Braunschweig. 1989

A MANUAL for CALIBRATION PROGRAM

In this appendix the use of the program will be discussed. The actions will be discussed step by step.

1. **before starting** The computation of the scaling factors requires information on the calibration object and the pick-up element used. So be sure that the following files are available in the directory before you use this program:

   (a) calibration object file : This file must have the same name as the name that will be given in the measurement run! It contains the depths of the object as stated in the corresponding calibration certificate. These numbers must be ordered in the same way the holes are detected by the roughness tester, e.g. the smallest on top.

   (b) scaling factor file This file contains the list of scaling factors for the pick-up element in use. The name of this file must be exactly equal to the pick-up element name given during the measurement run.
   The scaling factors are ordered in increasing accuracy position. When no scaling factor is available for a certain position the value 100 should be given.

2. **perform measurement** Perform a measurement on the calibration object in the accuracy position that is being calibrated (be certain to give the right names for the pick-up element and the calibration object as was discussed in the previous point). The resulting measured profile is directly drawn on the screen. Check this profile a priori on the next two points :

   (a) the smallest groove is on the left

   (b) the first points stay within range.

   These conditions are crucial for the successful execution of the program. In order to get good results from the calibration also try to keep as much holes as possible within range. The larger the hole that can be used for the calibration, the larger the accuracy of the result !

   If one of the conditions is not satisfied reposition the object and repeat the measurement!

   A successful measurement will produce two files:

   - *test.*a*01*
   - *test.*m*01*

Here *test* is the name of the measurement session. During such a session a number of measurements can be performed, each with a different accuracy position. To keep these different measurements apart, an extension is added which is the sequence number of the measurement in the session. The two files above are the files corresponding to the first measurement in the session *test*. The extension **.a** identifies the file containing general information on that particular measurement. The **.m** extension identifies the file containing the raw measurement data.

3. **transform measurement data** The current version of the program reads only the extended version of the measurement data files. This means that the **.m** files need to be transformed. This is done by the program **ASCFILE**. The results are given in the file *test*.d*01*.

4. **apply program** The correction factor can be computed by the program, by executing the command :

```
>calibrate
```

The program will now ask for the file containing the measurement data.

```
filenaam voor meetpunt (".d") - file :
```

At this point the name of the file produced in the previous step must be given. In this case this is the file *test*.d*01*. The program will process the data as described previously in this article.

When done the program will display the resulting scaling factor and the advice on the use of this result on the screen. The results will also be put in the file *test*.c*01*.

One possible advice is to adjust the scaling factor in the scaling factor file. However, be careful when adjusting. Before adjusting this file try to confirm this advice by repeating the procedure. Also try to find out why the scaling factor has changed.

[NOTE] : Currently a new version of calibrate, called FASTCAL, is available. This program reads the .m-files directly. The only difference with CALIBRATE is that the program asks for the name of the .m-file. When FASTCAL is used step 3 can be eliminated.

B FORMAT STANDARDS for FILES

In this chapter the format of all file types used in this project will be defined. In these formats we will use the type definitions of Turbo Pascal.

B.1 the measurement information (.a) file

This file contains general information on a certain measurement.
`cut_off_length`(*integer*) `resolution_step`(*real*) `no_samples`(*integer*) `accuracy_position`(*integer*)
`name_calibration_object`(*string*)
`name_client`(*string*)
`comment`(*string*)
`name_pick_up_element`(*string*)
`name_operator`(*string*)
`name_this_file`(*string*)

- `cut-off length` : The cut-off length used in the measurement session.
- `resolution_step` : The horizontal distance, in micrometers, between two samples.
- `no_samples` : This is the number of samples taken during the measurement and stored in the .m file.
- `accuracy_position` : The accuracy position of the roughness measurer during the measurement.
- `name_calibration_object` : Name of calibration object identifies the file containing the depth information from the calibration certificate.
- `name_client` : administrative
- `comment`: optional
- `name_pick_up_element` : This name identifies the file containing all scaling factors for the six accuracy positions.
- `name_operator` : adminstrative
- `name_this_file` : It is not quite clear why this information must be present in this file.

B.2 the rough measurement data (.m) file

This file contains a sequence of *no_samples* integers in the range [-2048..2047] separated by a single space . The numbers correspond with the measured (12-bit) value. To obtain the value in the micrometer scale this value must be normalized, this is a division by 2048, and multiplied with the scaling factor. This scaled value is stored in the .d file.
The reason to use this rough format is storage efficiency. The integer format requires only 2 bytes per sample, whereas a floating point version will require 6 bytes. The trade off, however, for this storage efficiency is the overhead introduced by the need to transform the data whenever it is read from the file.
The current version of the program can only deal with **.d**-files. A future version is planned that can deal with this rough format.

B.3 the scaled measurement data (.d) file

This file contains a processed version of the rough measurement data file. The file consists of *no_samples* lines of the form :

`x_value`(*real*) `y_value`(*real*)

This represents the two coordinates of a sample point. The x-value is simply increased by the *resolution_step* for each point. The y-value is the scaled version of the values in the **.m**-file.

B.4 the calibration result (.c) file

The calibration result file contains the values as computed during the program.

C THE VARIANCES of A and B

In this appendix we show how the variances of A and B which are introduced in section 2.1.1 can be derived. The discussion here is restricted to the calculation of only the variance of B, σ_B^2. The derivation of the variance of A, σ_A^2, is similar to this.

For reasons of convenience the notation of the variance of B, σ_B^2, is temporarily changed in $\sigma^2(B)$, and we introduce c_i as an abbreviation:

$$c_i = x_i - \frac{\sum_{i=1}^{N} x_i}{N}, \tag{14}$$

for $1 \leq i \leq N$.

We assume that all the uncertainties in the measurements are *uncorrelated*, which means that uncertainties in separate measurements are mutual independent.

Without proof we mention the following property:

$$\Delta = N \sum_{i=1}^{N} c_i^2 \tag{15}$$

The derivation then becomes:

$$\begin{aligned}
& \sigma^2(B) \\
= \quad & \{ \text{ equation 2 } \} \\
& \sigma^2\left(\frac{N(\sum x_i y_i) - (\sum x_i)(\sum y_i)}{\Delta}\right) \\
= \quad & \{ \ \} \\
& \sigma^2\left(\frac{N}{\Delta}\left(\sum x_i y_i - \frac{\sum x_i}{N} \sum y_i\right)\right) \\
= \quad & \{ \ \} \\
& \sigma^2\left(\frac{N}{\Delta} \sum (x_i - \frac{\sum x_i}{N}) y_i\right) \\
= \quad & \{ \text{ abbreviation 14 } \} \\
& \sigma^2\left(\frac{N}{\Delta} \sum c_i y_i\right) \\
= \quad & \{ \text{ variance property } \} \\
& \left(\frac{N}{\Delta}\right)^2 \sigma^2\left(\sum c_i y_i\right) \\
= \quad & \{ \text{ uncorrelated measurements } \} \\
& \left(\frac{N}{\Delta}\right)^2 \sum c_i^2 \sigma^2(y_i)
\end{aligned}$$

$$
\begin{aligned}
&= \quad \{ \text{ idem } \} \\
&\quad (\tfrac{N}{\Delta})^2 \textstyle\sum c_i^2 \sigma^2(y) \\
&= \quad \{ \text{ eliminating temporary notation } \} \\
&\quad (\tfrac{N}{\Delta})^2 \textstyle\sum c_i^2 \sigma_y^2 \\
&= \quad \{ \ \} \\
&\quad (\tfrac{N}{\Delta})^2 \ \sigma_y^2 \textstyle\sum c_i^2 \\
&= \quad \{ \ \} \\
&\quad \tfrac{N}{\Delta}\sigma_y^2 \ \tfrac{N}{\Delta} \textstyle\sum c_i^2 \\
&= \quad \{ \text{ equation 15 } \} \\
&\quad \tfrac{N}{\Delta}\sigma_y^2
\end{aligned}
$$

So, finally we have

$$\sigma_B^2 = \frac{N}{\Delta}\sigma_y^2$$

For the variance of A, an analogous derivation can be made and this leads to

$$\sigma_A^2 = \frac{\sum x_i^2}{\Delta}\sigma_y^2.$$

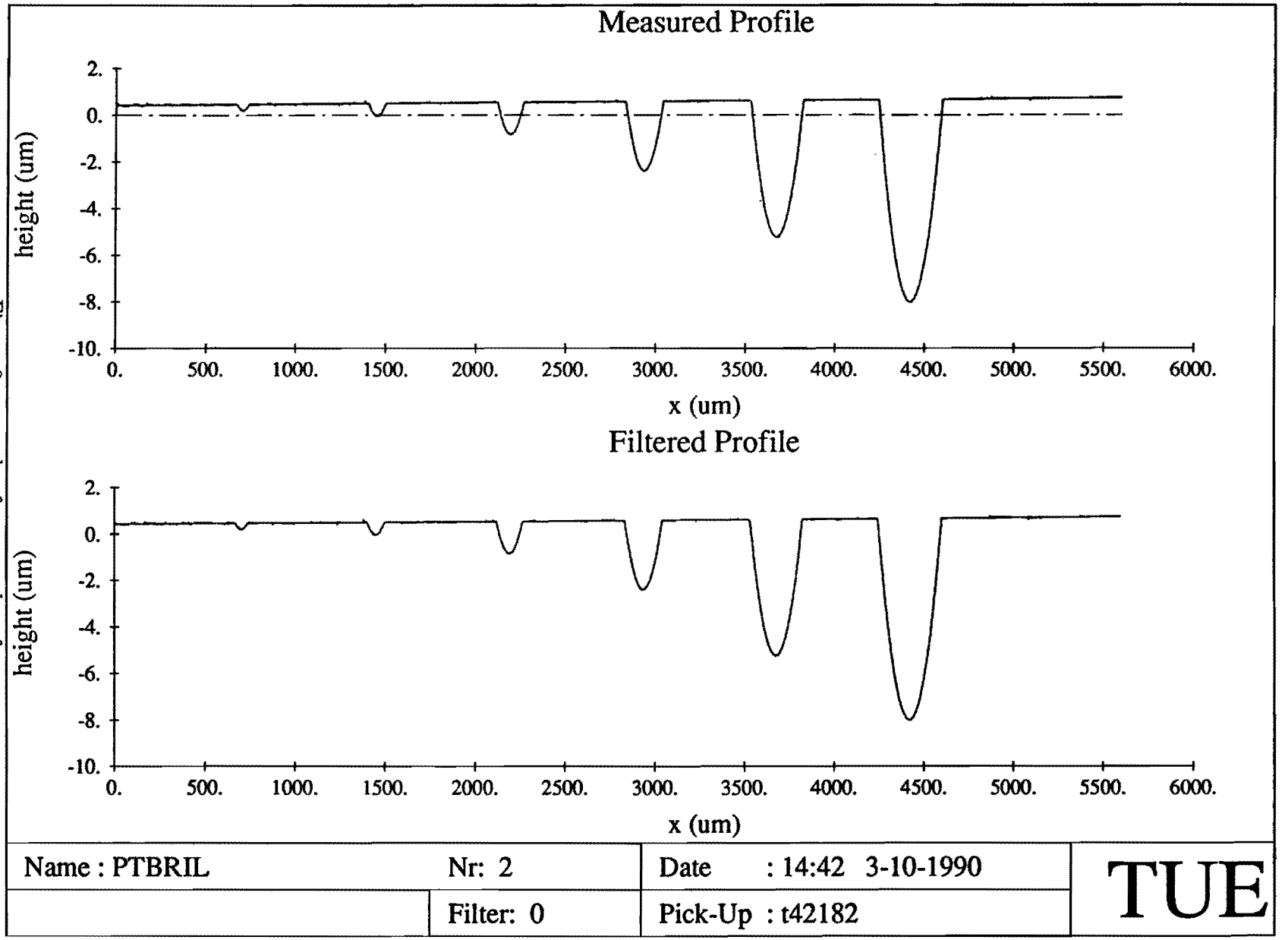

Figure 3: example of measured surface