CALCULATING A BEST FIT LINE

(Using QS-Pro's Data Source Expressions)

Synopsis

Given a number of measurements in the form of 2 dimensional co-ordinates, x,y, we can estimate the line from which they were taken. The technique uses linear regression and minimizes the sum of the squares of the errors in the y axis between the actual (measured) points and their on-line (calculated) counterparts.

Note that it is NOT necessary to understand the formulae given below, or their derivation, in order to use them successfully. If you are a mathematophobe you can safely skip the next section and move on to 'Implementation as Data Source Expressions'.

Application Note CAN-202 extends the principles outlined here to 3 dimensions and calculates the coefficients of a best fit plane.

Deriving the Formulae

The general formula for a line is…

$$y = A + Bx \tag{1}$$

The value to be minimized, the sum of the square of the errors, is given by…

$$S_r = \sum_{i=1}^{n} e_1^2 \quad = \sum_{i=1}^{n} \left(y_i - (A + Bx_i)\right)^2 \quad = \sum_{i=1}^{n} \left(y_i - A - Bx_i\right)^2 \tag{2}$$

Differentiating this expression wrt each of the coefficients A and B, gives us…

$$\frac{\partial S_r}{\partial A} = -2\sum(y_i - A - Bx_i) \tag{3}$$

$$\frac{\partial S_r}{\partial B} = -2\sum x_i.(y_i - A - Bx_i) \tag{4}$$

The summations have been simplified for ease of reading but all summations are from i=1 to i=n. The error, S_r is at a minimum when the differentials are zero, so setting these expressions equal to zero gives us a system of linear equations which we can solve for A and B.

$$0 = -2\sum(y_i - A - Bx_i) \quad \text{hence…} \quad \sum y_i = n.A + B.\sum x_i \tag{5}$$

$$0 = -2\sum x_i.(y_i - A - Bx_i) \quad \text{so…} \quad \sum x_i.y_i = A.\sum x_i + B.\sum x_i^2 \tag{6}$$

These equations, known as the 'normal equations', can be solved easily by substitution, the 'elimination of unknowns' method…

Considering eq.5: we can re-arrange this to give an expression for *A* in terms of *B*…

$$A = \frac{\sum y_i - B.\sum x_i}{n} \tag{7}$$

We can then substitute for *A* in eq.6 and obtain an expression for *B*…

$$B = \frac{n.\sum x_i.y_i - \sum x_i.\sum y_i}{n.\sum x_i^2 - \sum x_1.\sum x_1} \tag{8}$$

Hence, by evaluation of equations 8 and 7, we can calculate the values of the coefficients of the best fit line to, in theory, any number of arbitrary points.

Implementation as Data Source Expressions

In practice the points are likely to be derived from measurements of a nominally flat product probably resting on one or two reference points (which we will take to have $z = 0$), and touched by a number of probes or other measuring devices. Knowing the *x* co-ordinates of the reference points and probes, and their measured readings, (the *y* co-ordinates) we can calculate the best fit line. The following expressions make extensive use of QS-Pro's intermediate variable facility where any name appearing on the left hand side of an equals sign that it is not recognised as a feature or an instrument, is considered to be a new variable which may be needed in later expressions.

First you should get all the probe *x* and y co-ordinates into a standard form. Not only does this make the subsequent expressions shorter and easier to read (and hence debug) but the whole expression system then becomes portable to other applications.

Let's say we are touching the part in four places using instruments Probe1 to Probe4. Probe1 is at *x* location 20, Probe2 at 40, Probe3 at 60, and Probe4 at 80. The *x* and *y* co-ordinates can be entered as…

```
P1X = 20
P1Y = Probe1
P2X = 40
P2Y = Probe2
…
P4Y = Probe4
etc etc
```

This section requires two expressions for every point of contact and clearly will grow if you have more than four points. Any reference points (ie. non-probe, fixed points) must also be included but the *y* co-ordinate would be 0, or at least constant, for these points.

From these basic co-ordinates we enter expressions to calculate the various summation components required by the formulae. First the number of points of contact – in this example four, so…

```
n = 4
```

Then the summation components…

```
SumX = SUM ( P1X, P2X, P3X, P4X )
SumY = SUM ( P1Y, P2Y, P3Y, P4Y )
SumXX = SUM ( P1X * P1X,  P2X * P2X,  P3X * P3X,  P4X * P4X )
SumXY = SUM ( P1X * P1Y,  P2X * P2Y,  P3X * P3Y,  P4X * P4Y )
```

These expressions get longer if you have more points of contact with the part.

We now have all the components we need to calculate the coefficient of *x* (*B*), and the constant coefficient, *A*, from equations 8 and 7 respectively…

```
CoefB = ( n * SumXY – SumX * SumY ) / ( n * SumXX – SumX * SumX )
CoefA = ( SumY – CoefB * SumX )  /  n
```

On their own these coefficients are not particularly meaningful but they allow you to calculate the *y* co-ordinate of the line at any *x* co-ordinate. This makes possible a number of commonly required measurements…

Straightness

Straightness is usually expressed as the difference between the maximum deviation in *y* of any point from the line and the minimum deviation. Often the line can be taken from two mounting points of the part in a fixture but this reference line may not necessarily be the best line through the measured points and the straightness figure so calculated may therefore be exaggerated. The true straightness figure is the maximum minus the minimum deviation from the best fit line.

In our example above we have four points. The deviation of each point from the line is calculated by subtracting each measured *y* value from the *y* value calculated from the *x* co-ordinates…

```
Dev1 = P1Y – ( CoefA + P1X * CoefB )
```

Note the parentheses here since we are subtracting the calculated *z* from the measured *z*. Removing the parentheses would therefore require inversion of the signs within…

```
Dev1 = P1Y – CoefA - P1X * CoefB
```

Deviations 2, 3, and 4 are calculated similarly from P2, P3, and P4…

```
Dev2 = P2Y – ( CoefA + P2X * CoefB )
Dev3 = P3Y – ( CoefA + P3X * CoefB )
Dev4 = P4Y – ( CoefA + P4X * CoefB )
```

…and we can then use the MAXI() and MINI() functions to calculate the maximum and minimum deviation and hence the straightness…

```
Straightness = MAXI ( Dev1,  Dev2,  Dev3,  Dev4 )  - MINI ( Dev1,  Dev2,  Dev3,  Dev4 )
```

Obviously, like the best fit line expressions, flatness can be extended to more than 4 points of contact.

Parallelism

Parallelism is often measured between two lines, one of which may sometimes, but not always, be the reference line where it will be sufficient to calculate the maximum deviation of a best fit measured line to this. In the general case however two best fit lines must be calculated and the parallelism between them determined. Parallelism is measured similarly to flatness but is the maximum distance between corresponding points (same *x* co-ordinates) on each line minus the minimum. In the case of two finite straight lines with common *x* co-ordinates, the maximum and minimum separation will always be found at the ends, so we must calculate the *y* of each line at its ends and then find the difference.

Calculating two best fit lines is simply a matter of doubling up the expressions for one line…

We start with, say points 1-4 on the first plane (P1X, P1Y, … P4Z) and points 5-8 on the second (P5X, P5Y, … P8Z)

Calculate *n*, *SumX*, *SumY* etc for Line1, and the same for Line2 but, to keep the expressions separate, the Line2 values must have different names: say *m* for the number of points, and *SumU*, *SumV* etc for the summation components. The Line2 expressions can then give us coefficients *CoefD* and *CoefE*. The expression for the second line would be $v = D + Eu$.

From the two best fit lines we can calculate the end *y* values: *End1* and *End2* for Line1 and *End3* and *End4* for Line2. Note that these will use the *x* (and *u*) co-ordinates of the ends, which may or may not be the same as the co-ordinates of any of the measuring probes.

For example, suppose the two lines have ends at x=0 (End1X = 0) and x=100 (End2X = 0). The end *y*s will be…

```
End1Y = CoefA + CoefB * End1X
End2Y = CoefA + CoefB * End2X
End3Y = CoefD + CoefE * End1X
End4Y = CoefD + CoefE * End2X
```

And the parallelism would be calculated as…

```
Par = ABS ( ABS ( End1 – End3 )  -  ABS ( End2 – End4 ) )
```

Parallelism may be also be expressed as an angle between the lines…

```
Angle = ATAN ( Par / ABS ( End2X – End1X ) )
```