Least squares association of geometrical features by automatic differentiation

Michael Krystek\textsuperscript{1}, Shi Zhaoyao\textsuperscript{2}, Lin Jiachun\textsuperscript{2}

\textsuperscript{1} Physikalisch-Technische Bundesanstalt, Bundesallee 100, D-38116 Braunschweig, Germany
Tel.: +49[531]592 5016 Fax: +49[531]592 5015 E-mail: Michael.Krystek@ptb.de

\textsuperscript{2} Beijing University of Technology, College of Mechanical Engineering and Applied Electronics Technology, 100 Pingleyuan, Chaoyang District, Beijing, China 100124
Tel.: +86[10]67396781 Fax: +86[10]67390989 E-mail: linjc@bjut.edu.cn

Abstract

Least squares association of geometrical features plays an important role in geometrical product specification and verification. Most existing algorithms for the least squares association today usually do not give the covariance matrix associated with the parameters of the respective geometrical feature. The reason is, that the complexity of these algorithms can be very high, because partial differential quotients are needed. If the necessary partial difference quotients are calculated by hand and subsequently coded into an algorithm, there is a high risk to introduce unwillingly errors. This paper shows, how the least squares algorithm can automatically be generated solely from the equation specifying the distance function of the measured points from the geometrical feature.

Keywords: Least squares, association, geometrical features, automatic differentiation.

1. Introduction

Least squares association of geometrical features, such as straight lines, circles, planes, spheres, cylinders, and cones, plays an important role in geometrical product specification and verification, especially in coordinate metrology and form measurement.

Most existing algorithms for the least squares association today usually do not give the covariance matrix associated with the parameters of the respective geometrical feature. The reason is, that the complexity of these algorithms can be very high, because partial differential quotients are needed. This is especially true, if the inverse of the Hessian is involved, which is based on second order partial differential quotients.

If the necessary partial difference quotients are calculated by hand and subsequently coded into an algorithm, there is a high risk to introduce unwillingly errors into the computer program. Even if symbolic differentiation provided by modern computer algebra packages is used, it is mostly not inevitable to introduce transfer errors into the program code. Moreover, many functions grow quite complex when higher order derivatives are calculated by computer algebra packages.

Usually text books dealing with least squares fitting or measurement uncertainty calculations recommend to use finite difference quotients to approximate the necessary partial difference quotients. But this has the disadvantage of round-off errors in the discretization process and causes cancellation of significant decimal figures.
This paper will show, how the above mentioned difficulties can be overcome by using automatic differentiation. The least squares algorithm is automatically generated solely from the equation specifying the distance function of the measured points from the geometrical feature. This technique yields algorithms, which are exact to the numerical accuracy of the used computer.

2. Least squares association

The association criterion mostly used in engineering is the least squares criterion. Least squares association of geometrical features requires to fit a suitable geometrical element to the measured data, i.e. to determine the respective parameters describing the considered shape, like e.g. the centre coordinates and the radius of a sphere. Given the measured data, this method is applied by minimising the objective function

$$\chi^2(p_1, \ldots, p_m) = \sum_{k=1}^n d_k^2(p_1, \ldots, p_m|D_k)$$

by choosing the \( m \) parameters \( p_1, \ldots, p_m \) of the geometrical feature, where \( d_k(p_1, \ldots, p_m|D_k) \) is the distance of the \( k \)-th of \( n \) data points, measured perpendicular to the surface of the considered feature, and \( D_k \) represents the measured co-ordinates of this particular point. In 3D for example, \( D_k \) would be the data triple \((x_k, y_k, z_k)\). To be more explicit, we give here the distance function for the example of a sphere

$$d(x_0, y_0, z_0, r|x, y, z) = \sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 - r},$$

where the measured point co-ordinates are denoted by \( x, y, \) and \( z, \) and the parameters of the centre co-ordinates and the radius of the sphere, respectively, by \( x_0, y_0, z_0, \) and \( r. \) In this example we have \( m = 4. \)

Note that \( n \geq m \) is required, in order to guarantee that the least squares problem is solvable. However, this is usually fulfilled in engineering applications anyway.

Most algorithms used to solve the least squares problem are based on the quadratic approximation of the objective function obtained from a truncated Taylor series expansion, i.e.

$$\chi^2(p) \approx \chi^2(\hat{p}) + g^Tr + \frac{1}{2}r^THr,$$

where \( r = (\hat{p}_1 - p_1, \ldots, \hat{p}_n - p_n)^T \) denotes the residuals obtained by subtracting the values of the actually used parameters from the optimal parameter values \((\hat{p}_1, \ldots, \hat{p}_n), \) \( g = \nabla\chi^2(\hat{p}) \) the gradient of the objective function with respect to the parameters taken at the optimum, and \( H = \nabla\nabla\chi^2(\hat{p}) \) the respective Hessian matrix. The derivatives \( \nabla \) have to be taken with respect to the parameters. The Hessian is a \( m \times m \) matrix, which is positive definite near the optimum. Choosing suitable starting values for the parameters, those algorithms proceed by finding iteratively better and better approximations, until a certain stopping criterion is fulfilled. Since extensive literature exists about least squares optimisation algorithms, we need not to go into detail here. Newton type methods (with or without Levenberg-Marquardt regularisation), just to mention one popular family of superlinear converging algorithms, are based on the following iteration:

(a) solve \( \tilde{H}^{(i)}r + g^{(i)} = 0 \) for \( r = r^{(i)}), \)
(b) set \( p^{(i+1)} = p^{(i)} + r^{(i)}), \)
until \( \mathbf{r}^{(i)} < \varepsilon \) for a suitable small positive \( \varepsilon \). The matrix \( \mathbf{\hat{H}} \) is here either the original Hessian or a more or less close approximation to the Hessian, which is symmetric, but not necessarily positive definite during the iteration and thus requires appropriate measures. However, after the algorithm has converged to a (local or global) optimum, the Hessian is guaranteed to be strictly positive definite. Moreover, it can be shown, that the estimate of the uncertainty matrix \( \mathbf{U} \) of the parameters, which, of course, is always strictly positive definite, is related to the inverse of the Hessian by

\[
\mathbf{U}(\hat{\mathbf{p}}) = \frac{2}{n - m} \chi^2(\hat{\mathbf{p}}) \mathbf{H}^{-1}(\hat{\mathbf{p}}).
\] (4)

Note, that according to this relation, the least squares association should not be used, unless \( n \gg m \), i. e. the number of measured points is sufficiently large compared to the number of parameters, in order to obtain small uncertainties of the parameters.

3. Automatic differentiation

The preceding section has clearly shown, that partial derivatives are essential, not only to enable the calculation of the parameters of a geometrical feature under consideration, but also to obtain subsequently the uncertainty matrix associated to these parameters. Thus any least squares association strongly depends on a method, which allows to calculate the necessary derivatives accurately.

Two methods are frequently used today to compute partial derivatives of given functions numerically, firstly to derive the necessary formulae analytically by hand or by using a suitable computer algebra system, and subsequently to code the expressions in a computer program, or secondly to apply a finite difference approximation. However, there is another technique available since long, namely automatic differentiation (sometimes also called algorithmic differentiation). Unfortunately this approach is not widely known within the engineering community, although it provides the possibility to compute partial derivatives of arbitrary order of functions efficiently and accurately. Thus it is well suited to calculate the gradients and Hessian matrices necessary for a least squares association of geometrical features, as it has been briefly explained in the preceding section.

The idea behind automatic differentiation is, that differentiation in principle, as is well known from calculus, is a rule based procedure, which thus can easily be programmed to be done by a computer program. The computer program parses a given expression and uses so called term rewriting methods to apply successively the rules as given in table 1 to each subterm (subtree) resulting from the parsing process. Details of the underlying ideas to construct suitable algorithms can, for example, be found in [1] or [2].

<table>
<thead>
<tr>
<th>rule</th>
<th>expression</th>
<th>derivative</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( a = c )</td>
<td>( da = 0 )</td>
<td>( c ) is a constant</td>
</tr>
<tr>
<td>2</td>
<td>( a = x )</td>
<td>( da = 1 )</td>
<td>( x ) is the dependent variable</td>
</tr>
<tr>
<td>3</td>
<td>( a = u + v )</td>
<td>( da = du + dv )</td>
<td>( u ) and ( v ) are expressions, depending on the variable ( x )</td>
</tr>
<tr>
<td>4</td>
<td>( a = u - v )</td>
<td>( da = du - dv )</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>( a = u \cdot v )</td>
<td>( da = u \cdot dv + v \cdot du )</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>( a = u/v )</td>
<td>( da = (du - a \cdot dv)/v )</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>( a = f(u) )</td>
<td>( da = du \cdot f'(u) )</td>
<td>( f ) is an arbitrary function, ( f' ) is the derivative of ( f )</td>
</tr>
</tbody>
</table>
If such an algorithm is applied to the distance function as given in equation (2) as an example for a sphere, the results summarised in figure 1 and table 2 are obtained.

\[
d = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2} - r.
\]

Fig. 1. Abstract syntax tree (AST) for the formula \( d = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2} - r \).

Table 2. Example for the partial differentiation of the code list with respect to the parameter \( x_0 \) for the distance formula \( d = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2} - r \).

<table>
<thead>
<tr>
<th>code list</th>
<th>derivative of the code list</th>
<th>after optimisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t1 = x - x_0 )</td>
<td>( dt1 = 0 - 1 )</td>
<td></td>
</tr>
<tr>
<td>( t2 = t1 \times t1 )</td>
<td>( h1 = t1 \times dt1 )</td>
<td></td>
</tr>
<tr>
<td>( t3 = y - y_0 )</td>
<td>( dt3 = 0 - 0 )</td>
<td></td>
</tr>
<tr>
<td>( t4 = t3 \times t3 )</td>
<td>( h3 = t3 \times dt3 )</td>
<td></td>
</tr>
<tr>
<td>( t5 = t2 + t4 )</td>
<td>( dt5 = dt2 + dt4 )</td>
<td></td>
</tr>
<tr>
<td>( t6 = z - z_0 )</td>
<td>( dt6 = 0 - 0 )</td>
<td></td>
</tr>
<tr>
<td>( t7 = t6 \times t6 )</td>
<td>( h5 = t6 \times dt6 )</td>
<td></td>
</tr>
<tr>
<td>( t8 = t5 + t7 )</td>
<td>( dt8 = dt5 + dt7 )</td>
<td></td>
</tr>
<tr>
<td>( t9 = \text{sqrt}(t8) )</td>
<td>( h7 = dt8 / t9 )</td>
<td></td>
</tr>
<tr>
<td>( d = t9 - r )</td>
<td>( dd = dt9 - 0 )</td>
<td>( dd = -t1 / t9 )</td>
</tr>
</tbody>
</table>
The code list resulting from a suitable parsing process is given in the left column. This code block calculates the distance \( d = \sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 - r} \), as can easily be verified by successively reinserting the temporary expressions. Internally the code list is stored as an abstract syntax tree (AST), as is depicted in figure 1, which can automatically be generated by a recursive descent parser from the input expression representing this formula.

The middle column of table 2 shows the derivative of the code list, as obtained by a simple application of the rules given in table 1 to the first column of table 2 line by line. The computer program is doing just this by pattern matching, while traversing the AST in preorder, i. e. depth first, left to right.

As can be seen, the resulting code list is much longer and contains superfluous code like \( \text{dt6} = 0 - 0 \) or \( \text{dd} = \text{dt9} - 0 \). This is generally the case after the differentiation rules have been applied. Depending on the input expression, the derivative of the code list can be longer than the original code list by a factor of three to ten. Thus a subsequent optimisation process is needed to simplify the code resulting from the differentiation algorithm. The optimisation algorithm is based on techniques, which have been developed for optimising compilers and uses the well known algebraic rules, as e. g. commutativity and associativity, for algebraic simplification, as well as techniques like constant propagation, constant folding, common code elimination, and dead code elimination. The result of the application of the optimisation process for the example of a sphere is shown in column three of table 2. In comparison with column 2, this code is much more condensed, since it uses subexpressions of the first column. This strategy is strongly recommended, because the original code needs to be calculated anyway, in order to obtain the value of the objective function.

By application of the outlined methods, a computer program can be written, which automatically generates another program, which in turn is able to solve a particular least squares association problem. The input for the program generator is just the distance function of the problem under consideration and the choice of a particular solving method to be used. The generated program is subsequently compiled as usual and linked with a library providing the necessary supporting functions, which implement the required optimisation strategies.

4. Conclusion

It has been shown, how a computer program can automatically be generated, which subsequently can solve a particular least squares association problem. This can be achieved by combining well known methods of compiler construction techniques with the method of automatic differentiation, as well as with suitable optimisation strategies.

The advantage of the proposed approach is, that by this means any least squares association problem can be solved quickly and accurately. It is no more necessary, that a potential user is able to write algorithms of the required complexity. He can just rely on a tool which solves his problem and he can be sure, that the calculated parameters and the associated uncertainty matrix are as accurately calculated as is possible corresponding to the state of the art.

Since the verification of the correctness of algorithms is nowadays not a problem anymore, the generator can be used to make a reference software package for a certification of the relevant least squares association programs packages, which are used in coordinate metrology.

References