Reprinted from SPIE Vol. 554—1985 International Lens Design Conference

Automatic Lens Optimization: Recent Improvements

Donald C. Dilworth

Optical Systems Design, Inc.
3 Johnson Avenue, Medford, MA 02155

Abstract

Several refinements to the PSD optimization algorithm have yielded improved convergence and enhanced ability to cope with unusual nonlinearities encountered in the course of optimization.

Introduction

Most lens design calculations today are carried out on large computer programs with an optimization algorithm based on the well-known damped-least-squares method or a variation. The computation costs of a typical design on these programs are frequently a major item in the design budget, and a method of improving the performance of the algorithm would be of considerable interest. Even so, there has been very little work published in the last decade toward this end.

One reason for this lack of progress is the intrinsic difficulty of the subject: minimizing a non-linear problem in many dimensions, subject to absurd (to the programmer) boundary conditions. Although some alternate approaches have been proposed (1,2), there has been no recent breakthrough in mathematical understanding, and it is probable that the design of algorithms, like the design of lenses themselves, will continue to require an intuitive approach and much experimentation. This report describes a number of seemingly efficacious notions and their effects, or lack of, on the convergence rate.

Methodology

Since the nonlinearity of the lens design problem is the chief impediment to faster convergence, we are interested in any approach that either characterizes or compensates for it. Evaluating the results of a potential improvement to the algorithm can be deceptive since what works well for one lens may not work for another. Our approach has been to evaluate a set of design problems, of varying difficulty, to see if any new scheme is consistently better than a baseline algorithm, which for our purposes is usually the ordinary DLS method or the previously described pseudo-second-derivative (PSD) method. Two of the methods outlined below seem to satisfy this criterion.

Characterization Methods

The PSD algorithm method basically obtains information from the change of derivatives in subsequent iterations to infer something about the second partials (3). This additional knowledge improves convergence, particularly if the statistically expected effect of the inhomogeneous second partials on this process is considered (4). It is possible to extend this insight one step further:

The second partials are utilized in the normal PSD process when the Jacobian matrix is constructed:

$$L_{jk} = \sum \frac{\partial f_i}{\partial x_j} \frac{\partial f_i}{\partial x_k} + \sum f_i \frac{\partial^2 f_i}{\partial x_j \partial x_k} \qquad (1)$$

where the $\vec{f}$ are image errors and the $\vec{x}$ are design variables. We use the approximation

$$\frac{\partial^2 f_i}{\partial x_j^2} \approx \frac{\left.\frac{\partial f_i}{\partial x_j}\right|_{\Delta x_j} - \frac{\partial f_i}{\partial x_j}}{|\Delta x_j| + \sqrt{\sum_k \Delta x_k^2}\,\Big|_{k \neq j}} \qquad (2)$$

The radical in this equation is derived from the assumption that the statistically expected value of the inhomogeneous second partial is about the same as that of the homogeneous second partial. In other words,

$$\frac{\partial^2 f_i}{\partial x_j \partial x_k} \approx \frac{\partial^2 f_i}{\partial x_j^2} \quad . \tag{3}$$

Although the approximation is effective as an optimization algorithm, it begs the question: which of the two variables' second partials is it similar to? Inhomogeneous derivatives involve two variables while the homogeneous derivative but one.

Idea Number 1

The most obvious method of resolving this question is to take the geometrical mean:

$$\frac{\partial^2 f_i}{\partial x_j \partial x_k} \approx \sqrt{\frac{\partial^2 f_i}{\partial x_j^2} \frac{\partial^2 f_i}{\partial x_k^2}} \quad , \tag{4}$$

which leads to

$$\frac{\partial^2 f_i}{\partial x_j^2} \approx \frac{\frac{\partial f_i}{\partial x_j}\Big|_{\Delta x_j} - \frac{\partial f_i}{\partial x_j}}{|\Delta x_j| + \sqrt{\sum_k \Delta x_k^2 \left|\frac{\partial^2 f_i / \partial x_k^2}{\partial^2 f_i / \partial x_j^2}\right|}}\Bigg|_{k \neq j} \quad . \tag{5}$$

Attractive as this idea appears, we have never experienced an improvement in convergence with it. A related idea, however, is much better:

Idea Number 2

The rightmost quantity in Eq. 1 ideally contains information about the nonlinearity of the problem. Perhaps this quantity can be used to resolve the question about the mixed partials: let us assume that the contribution to the inhomogeneous second partial in Eq. 4 from the variable of different kind (k) is the same as that of the variable of like kind (j) times the ratio of this quantity, which we call SEC, for the two variables. A very linear variable (on the average) would logically contribute little to the mixed partial while a non-linear one would increase the value of the latter. In other words:

$$\frac{\partial^2 f_i}{\partial x_j \partial x_k} \approx \sqrt{\frac{\partial^2 f_i}{\partial x_j^2} \frac{\partial^2 f_i}{\partial x_j^2} \frac{SEC_k}{SEC_j}} \quad . \tag{6}$$

This approximation leads immediately to

$$\frac{\partial^2 f_i}{\partial x_j^2} \approx \frac{\frac{\partial f_i}{\partial x_j}\Big|_{\Delta x_j} - \frac{\partial f_i}{\partial x_j}}{|\Delta x_j| + \sqrt{\sum_k \Delta x_k^2 \frac{SEC_k}{SEC_j}}}\Bigg|_{k \neq j} \quad . \tag{7}$$

We call this method the PSD III algorithm, and it appears to be one of the most effective methods we have tried. Sample convergence curves are shown below for the methods described in this paper, and its merit is evident.

One appealing feature of this method is that if the values of the second partials are indeed given more accurately with this method, they should become even more accurate in subsequent iterations since better values are input to the approximation. As the second

derivatives become more refined, they lead to yet more accurate values.

Compensation Methods

An alternative approach is to bypass the details of the nonlinearities and instead concentrate on the variables. We observe that, in subsequent iterations, many of the variables (but not all) tend to move in the same direction as on the previous iteration. Let us explore the implications of this:

Idea Number 3

Using the normal DLS method as a baseline, let us obtain the solution vector on a given iteration and then do a linear transformation over all variables such that:

1. Variable number one is exactly parallel to the last solution vector.

2. All other variables are exactly normal to variable one.

We call this the Directed-Vector-Method. The vector transformations are straightforward and will not be presented here for the simple reason that this method has consistently given exactly the same solutions as the DLS method. We postulate that no scheme involving only linear transformations among variables can be of value, except for cases where metric effects are involved -- and these are better treated differently.

Metric Considerations

Which brings up the subject of metric considerations. It is easy to show that the presence of damping in a DLS optimization is equivalent to requiring that the sum of the squares of the elements of the solution vector, suitably weighted, is minimized along with the merit function. A high damping therefore tends to give a small solution vector.

But if we were to measure a given lens thickness in miles, instead of MM, for example, any change would tend to be very small numerically, and the damping effect would not apply to that variable as much as to the others. By scaling the dimensions of a variable we can therefore influence the degree to which the program tends to favor its use. Let us utilize this effect.

Idea Number 4

Let us assume that some variable is not doing its share of the work of reducing the merit function, for whatever reason -- because of nonlinearities, inappropriate metric, errors in the many approximations outlined above, etc. Can we detect these cases and compensate for them?

We can define the VALUE of a variable in term of its contribution to the change in the merit function each pass. The merit function, φ is to be minimized, but the gradient of φ, G, is to be driven to zero. It is more useful to consider the effect of a given variable on G than on φ directly.

We define the VALUE of a variable as

$$\text{VALUE}_j = \Delta x_j \frac{\partial G_j}{\partial x_j} / G_j \tag{8}$$

where G(j) is the derivative of the merit function with respect to the variable X(j). If the VALUE of the variable is very small we would like the metric in effect on that variable to increase -- thereby forcing the program to utilize it in a more forceful manner if possible. This scheme could, in principle, account for a whole range of optimization problems since it does not involve details of the nature of those problems but only seeks to reduce their symptoms.

One obvious way to achieve this effect is to define a new metric on the next iteration as

$$\text{METRIC}_j' = \text{DAMP}^A \ \text{VALUE}^B \ \text{METRIC}_j^C \tag{9}$$

where the exponents determine the magnitude of the influence we wish to apply to the situation. The the damping factor, DAMP appears because a high damping will tend to reduce the VALUE of most variables and we do not want the metric to be affected on this

account. Proper values of exponents A and B will have a compensating effect for a change in VALUE from this cause. The old METRIC appears because we want to avoid great fluctuations from one pass to another, and any degree of sluggishness may be ordered with a proper value of exponent C. From these arguments we can predict that exponents A and B should probably be negative, while C should be positive, and all should be less than unity, in absolute value.

This is not much to go on, however, and to test the theory we are reduced to making a great many trials, with a range of values for each exponent, using a number of lenses -- and looking for strong trends in the data.

Fortunately this labor has been rewarded: this method, which we call the floating-metric (FM) method, has in some cases improved the convergence of PSD I as much as has the PSD III method, described above. (The PSD I method is the earliest version, in which the radical in Eq. 2 is replaced with the constant .0001.) The best values of each exponent are not quite the same for each lens, but good results are obtained using (A,B,C) = (-.25, -.25, +.5).

To be effective with the PSD methods, the metric must be folded into the first derivatives, but not the second -- thereby acting as a reduced damping for those variables having a small VALUE. The method is less effective in runs using the PSD III method, indicating that perhaps the two methods have similar effect and are fighting each other -- but in some cases just the ordinary DLS method with the FM enhancement gives excellent results. The FM method can be used with any form of DLS or derived technique.

Experimental Results

A better optimization algorithm should, of course, give faster convergence on typical lens design problems. But closer inspection reveals some complications to this simple criterion: what if two methods generate quite different lenses whose performance is similar? Or what if the convergence rate of one is slower but the minimum is lower? One must test many lenses to be sure that what works today will be a reliable tool for future lenses with different requirements. We have seen examples of cases where, using the FM method with a particular set of exponents, extremely fast convergence is obtained on some test lenses -- but no progress at all on others. So one must judge a new algorithm with caution.

Since the object of this study is to develop better algorithms for use in the SYNOPSYS lens design program, which is a commercial program that will be used on every conceivable type of problem, one of the most important qualities of an algorithm is its reliability. The following examples will show the typical differences we have observed in many more cases where we contrast the convergence of the several methods. The conclusions are discussed below.

Triplet Example

An excellent example of a lens for which there is more than one solution is the following triplet, where the stop has been placed on surface one rather than the central element. The starting lens and the convergence curves for several runs are shown in Figs. 1 and 2.

It is apparent that the ordinary DLS method (curve I) did little to improve this lens while PSD III (curve A) was excellent. In this case, the FM method, when applied to a pure DLS run, gave results (curve D) nearly as good as the PSD methods. The FM method improved PSD I (B) but not PSD III (E). This pattern applies to most of the cases we have tested.

The lenses resulting from the PSD methods were similar to each other; the configuration in Fig. 3 is typical.

(Another optimization trick, supported in some circles, is to scale the diagonal of the matrix L by an adjustable quantity which is itself optimized in sub-passes; this method gives the most improvement per iteration. It seems to be a dangerous method, however -- and this comment may apply to any algorithm that gives a great improvement very quickly: if the initial lens is poorly corrected, this trick can alter a few variables by a large amount so that the lens scarcely resembles the starting configuration even on the second pass. If the lens form has been intelligently selected this can ruin the design. Fig. 4 shows what happened when this trick was used in a PSD I run on the triplet, and Fig 5 shows the same with a PSD III run. In the latter run the damping optimization was done for only two passes, but the lens form was already seriously altered (and curiously, stumbled into a region of lower merit function, curve F). In the vast majority of cases, this trick does more harm than good, and we generally avoid it.)

Conference Problem

Another test case we are fond of using is Problem Number One from the previous Lens Design Conference in which a double-Gauss lens was opened up and reoptimized. The starting design is shown in Fig. 6, and the convergence curves in Fig. 7. These examples show that a single run is sufficient to design this lens. PSD III is the best method, and the results of this run are shown in Fig. 8.

Other Test Cases

Having shown two typical examples of the convergence rates of the PSD methods, we would like to mention a test case that is apparently not a good example -- the so-called "banana valley" test of Rosenbrock. We have found that all of the most successful algorithms we have tested performed essentially identically on this problem, and it therefore seems not to be a good indicator of what will work on real lenses. We have also found some very bad algorithms that converge quickly in some cases, on this test.

We also have found that in cases where the ordinary DLS method gives good results the PSD methods do not necessarily do any better. But this is itself a good test, since it would be unacceptable if they did worse in these cases, and they do not.

Conclusions

We have been studying the PSD optimization algorithm and continue to discover new avenues where seeming problems or obstacles can be addressed or overcome by means of mathematical tricks of some kind. Of course, most of these bright ideas turn out not so effective -- but it is through this kind of research that progress will continue to be made in this field, and the investigation is advised.

As of now, it appears that the PSD III algorithm is the fastest and most reliable for a wide variety of problems, although the PSD I plus FM option is also very good. Both of these methods are available in the SYNOPSYS lens design program.

References

1. E.D. Huber, Appl. Opt. 21, 1705 (1982).
2. P.N. Robb, Appl. Opt. 18, 4191, (1979).
3. D.C. Dilworth, Appl. Opt. 17, 3372, (1978).
4. D.C. Dilworth, SPIE Vol. 399, 159, (1983).

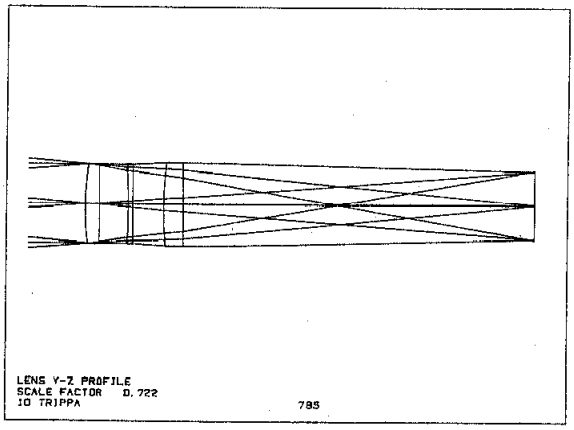

Figure 1. Triplet starting configuration.

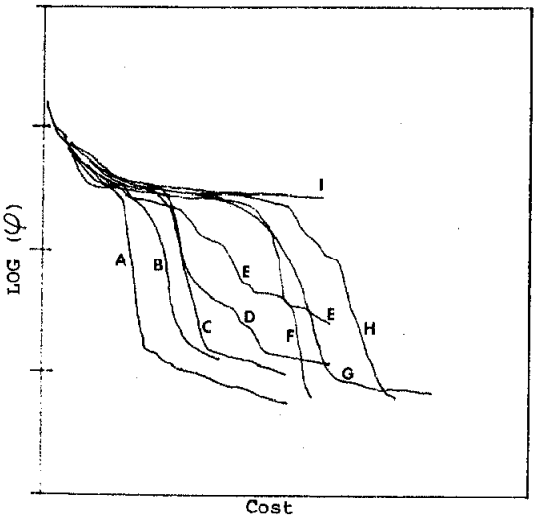

Figure 2. Convergence curves for triplet lens. Curves are as follows: A = PSD III, B = PSD I-FM, C = PSD I, D = DLS-FM, E = PSD III-FM, F = PSD III with damping search (OD), G = PSD I (OD), H = PSD I-FM-(OD), I = DLS.

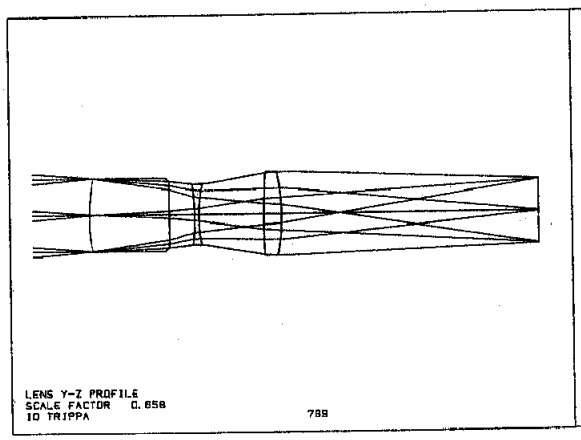

Figure 3. Triplet final configuration A.

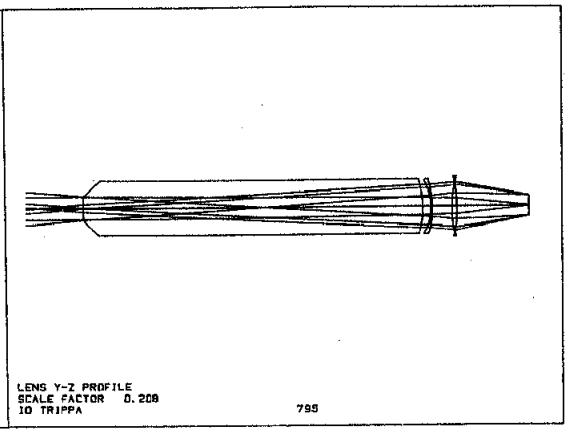

Figure 4. Triplet final configuration G.

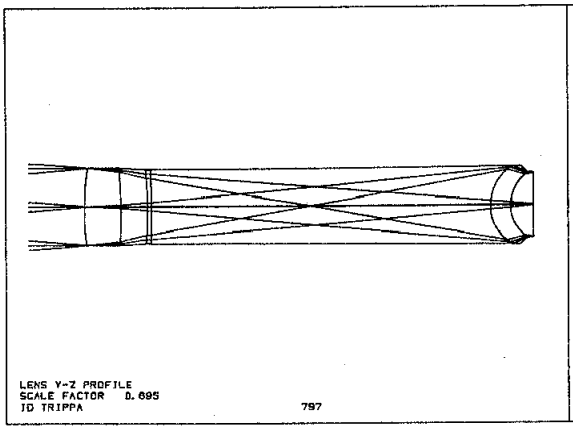

Figure 5. Triplet final configuration F.

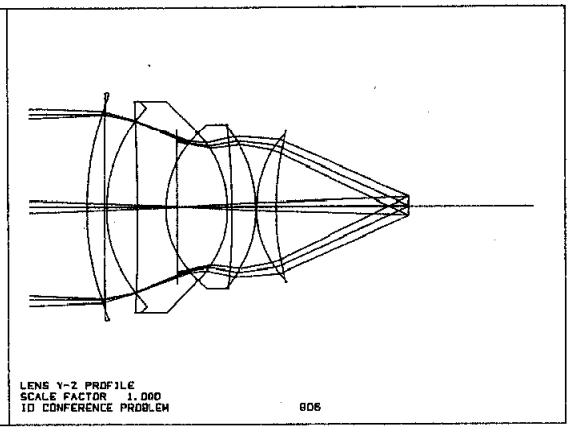

Figure 6. Conference problem starting lens.

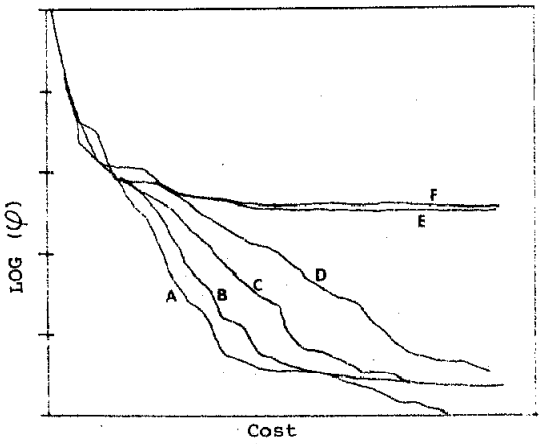

Figure 7. Convergence curves for conference problem. Curves are as follows: A = PSD III-FM, B = PSD III, C = PSD I-FM, D = PSD I, E = DLS, F = DLS-FM.

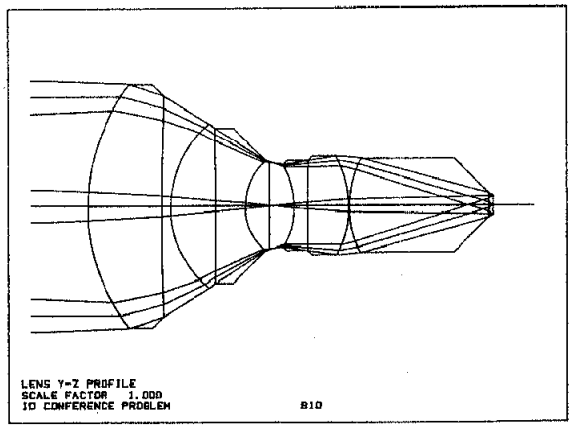

Figure 8. Results of PSD III on the conference problem.