Complex lens design: searching for a needle in a haystack

Sergey Mashchenko
Department of Physics and Astronomy, McMaster University, Hamilton, ON, L8S 4M1, Canada
E-mail: syam@physics.mcmaster.ca

Abstract. Optical design of complex (multi-element) lenses is traditionally considered to be part science and part art, primarily because of the enormous complexity of the problem. Recent advances in high performance computing (HPC) made it feasible to adopt a purely scientific approach in discovering new lens designs. In this paper, I formulate the task of finding a new lens design that satisfies a given set of constraints as a search for the global minimum of a function of unknown and very large (∼ 30 – 100) number of dimensions. I address the significant complication that only a tiny fraction of the volume of the free parameters space is physically accessible. I propose a smart lens drafting algorithm which circumvents this difficulty. I present my numerical code which can be used to discover novel complex lens designs in a fully automatic fashion. I discuss the HPC aspects of the problem of searching for minima of high dimensionality functions.

1. Introduction
A purpose of a lens is to produce an image. A single-element lens produces a low quality image, which is usually not good enough for many applications, including photography, astronomy, medicine and so on. Compound, or complex lenses (consisting of more than one element) can produce significantly better quality images. Unfortunately, there is no general theory for complex lenses. As a result, the field of optical lens design is considered to be partly art, partly science [1].

Modern optical lens design consists of two relatively independent steps:

(i) "Art". Choosing a good starting point: a draft design which would already be able to produce an image (not necessarily of good quality), with the main lens parameters (focal length, aperture, type of projection etc.) been not too far from the target values. The following approaches are used [2]:

- A mental guess.
- A previously designed lens in the company files.
- Purchase of a competing lens and analysis of its structure.
- A search through the patent files.

(ii) "Science". Lens optimization. Starting from the draft design, lens parameters are gradually optimized in an attempt to achieve the target lens parameters (optical,
Figure 1. Four randomly oriented two-dimensional slices through the neighborhood of a local minimum of the merit function for a complex lens consisting of five elements (all surfaces are conic) with an aperture stop. The local minimum is at the centre. Number of dimensions is 39. The square size is one in scale-free units. Black areas are physically inaccessible. I use logarithmic scale to plot the merit function (~ 30 orders of magnitude from white to almost black color).

It is usually accomplished by minimizing a specially designed multi-dimensional merit function. Both approximate (aberration theory) and direct (ray tracing) methods can be used in designing the merit function.

2. Automatic lens design

The current two-step approach to complex lens design makes it impossible to design complex lenses in a fully automatic fashion. I address this issue by reformulating the complex lens design problem as a single-step procedure of finding the global (or a very good local) minimum of the multi-dimensional merit function.

Monte Carlo minimization is the method of choice for this type of problem. Specifically, a random initial point is generated, and a multi-dimensional minimizer is used to descend to the nearest local minimum. The process is repeated many times, until the global (or sufficiently good local) minimum is found.

To avoid the unknown biases introduced by approximate theories, and to make the whole multi-dimensional space of the lens parameters fully explorable, in my algorithm I use the direct method (tracing of hundreds and thousands of rays through the system) to design the merit function.
2.1. Properties of the global merit function

My merit function is computed by (1) tracing of hundreds to thousands of rays of different wavelengths, coming from sources located at different distances from the lens and from the optical axis of the system, (2) finding the optimal focal plane, where these rays produce the best quality image, (3) computing the $\chi^2$ deviation of the rays in the focal plane from the ideal (theoretical) positions, and (4) computing and applying a large number of punishment coefficients (multipliers to $\chi^2$). Rays about to be lost (approaching the edges of lens surfaces, the aperture stop, and the rear lens flange) are artificially dimmed (their weight in $\chi^2$ computation approaches zero as they are approaching the edge); this ensures that the merit function stays smooth even if some rays disappear or re-appear.

The most important punishment coefficients are for vignetting, focal length (to drive the system toward the target focal length value), glass weight (to minimize the weight of the lens), back focal length (to ensure that it is larger than a given minimum value), medium (both glass and air) thickness (to keep it within the allowed range), deviation from a nearest real glass (see § 2.3), and distortion (deviation from a given mapping function, which could be gnomonic, stereographic, orthographic etc.).

A serious complication is that the merit function is mostly "vacuum": only a tiny fraction of the total hyper-volume of the lens parameter space is physically accessible. This can be seen in Figure 1, which shows the neighbourhood of a local minimum of 39-dimensional merit function (for a five elements lens design). In one dimension, the extent of the physically accessible space (white blob) is $\sim 0.3$ of scale-free units (the size of the square window is 1). In 39 dimensions this will result in the volume filling factor of $0.3^{39} \approx 4 \times 10^{-21}$. The probability to detect this physically accessible blob by using a straightforward Monte Carlo approach (generating random initial points within this hyper-cube) is hence negligibly small. The solution is to design a
"smart" algorithm for generating random draft lens designs. I describe my drafting algorithm in § 2.2.

Tiny local minimum islands (like the one in Figure 1) is not the only type of structure exhibiting by the global merit function. As one can see in Figure 2, local minima are connected by filaments. The fact that these filaments are not visible in Figure 1 suggests that they are of a lower dimensionality. In my serial farm runs for 5–8 elements designs I found that the local minima blobs have sizes $\sim 0.3$ of scale-free units, whereas the distance between the initial, draft design point and the minimum is much larger: $\sim 2 - 10$ (typically $\sim 5$). The fact that in my experiments initial random points never land directly inside local minima islands strongly suggests that filaments occupy much larger volume than local minima blobs. The existence of lower-dimensionality filaments can also be deduced from theoretical arguments, specifically from the presence of degeneracies in optical designs (e.g. changing both the surface curvature and the refractive index in concert can keep a lens element’s power constant).

2.2. Smart lens drafting

My algorithm for generating random draft lens designs (used as initial points in the global Monte Carlo minimization of the merit function) is based on ray tracing, and is very fast: it usually takes seconds to generate one good random design, on one CPU. Draft designs produced by the algorithm are capable of creating an image, with the focal length, aperture, back focal length, rear flange opening and other lens parameters reasonably close to the target values. There is some control over the amount of optical aberrations, distortions, and vignetting present in draft designs. The algorithm also assigns random glass to the lens elements, using my optical glass model (see the next section).

2.3. Glass model

Multi-dimensional optimizers are not designed to handle non-continuous (discrete) parameters. Most of the lens parameters (surface curvatures, vertex coordinates etc.) are naturally continuous, but some are not. To make the whole multi-dimensional space of lens parameters explorable and optimizable, one has to reduce the number of discrete lens parameters to an absolute minimum. Ideally, only the number of individual lens elements should be left as a discrete parameter; all other discrete quantities should be converted to pseudo-continuous ones.

Optical glass is one of such discrete parameters. Though it is physically feasible to produce
Figure 4. Glass punishment function (the same coordinates as in Figure 3). The range is from no punishment (white) to severe punishment (black). Left panel: at the beginning of the optimization process, when the image quality is poor. Right panel: at the end of the optimization, when the image quality is high. The punishment function is computed separately for each element of the lens. The merit function is multiplied by all of the individual punishment functions.

Optical glass with smoothly varying properties, lens design and manufacturing is based on a discrete set of ∼ 100 commercially available types of optical glass.

The commercially available optical glass is reasonably uniformly scattered in the plane with $x \equiv v_d n_d^3$ and $y \equiv n_d$ coordinates (see Figure 3; here $v_d$ is the Abbe number and $n_d$ is the refractive index for $d$-line). I found that the following equation is a good fit for the refractive index $n$ as a function of the wavelength $\lambda$ (in nm) and $n_d$ and $v_d$ parameters:

$$n(\lambda, n_d, v_d) = ((-1.1286752 \times 10^{-8} \Delta \lambda + 4.6105463 \times 10^{-6}) \Delta \lambda - 1.3883940 \times 10^{-3}) \Delta \lambda + 5.2671310 \times 10^{-1}) \Delta \lambda(1 - n_d)/(109.87 v_d) + 0.99864003 n_d + 1.5353551 \times 10^{-3}.$$  \hspace{1cm} (1)

Here $\Delta \lambda = \lambda - 596$ nm. For the 109 glass types from the Ohara Corporation online catalog and seven spectral lines ($A'$, $r$, $C$, $He$–$Ne$, $e$, $F$, $g$), the standard deviation of equation (1) from the real glass is $1.8 \times 10^{-4}$.

At the draft design stage, I generate random glass by choosing a random point in the $(x, y)$ plane. Such glass is physically plausible, but it is not manufactured.

At the optimization stage, the glass properties of each lens element are described by the two continuous parameters, $x$ and $y$. Refractive indices for all wavelengths are computed using the equation (1). Initially, $x$ and $y$ are allowed to vary freely (within the physically plausible region); as the quality of the lens is improving, the system is driven towards commercially available glass types by utilizing a punishment function (see Figure 4).

At the end, the optimized model glass is quite close to real types of glass. The final optimized lens design is produced by switching to the nearest real glass types and re-optimization of the lens.
2.4. The code

I have written a code which implements the algorithm described in the previous sections. It is written in C and has more than 10,000 lines of code. The code consists of draft and optimization modules, and requires no interaction. As it may take weeks, months or even years to discover a high quality design on a single computer, the code should be run in parallel as a serial farm on a cluster.

The code makes a heavy use of GSL\(^2\) (GNU Scientific Library), specifically the random number generation (see §4) and multi-dimensional minimization routines. As my merit function is based on ray tracing, its partial derivatives are not readily available. This narrowed down my choice for minimizer to the Nelder-Mead simplex algorithm \(^3\). I modified the GSL’s routine for Nelder-Mead simplex minimization to make it suitable for parallel (serial farming) applications.

I found that for lenses consisting of more than 2 – 3 elements the simplex algorithm does

\(^1\) http://www.oharacorp.com/catalog.html

\(^2\) http://www.gnu.org/software/gsl

---

**Figure 5.** Optimized design for a 5 elements SLR lens with the focal length 50 mm and the f-number 1.4 (bottom panel). All surfaces are conic. Optimization included the location and diameter of the aperture stop, and the constraints imposed by the camera (minimum back focal length, location and diameter of the rear lens flange). The resolution curves (top left panel) and transmission curves (top right panel) are shown for three different values of f-number (1.4, 2.8, 5.6).
not directly converge to a true minimum. Instead, the code converges to a complex saddle point. To address this I implemented a "multi-hop" approach: the code performs a sequence of re-optimizations ("hops"), each re-optimization starting from the previous best result, with gradually shrinking initial search radius. Once the distance traveled during a few consecutive hops becomes very small, a convergence to a true local minimum is achieved. Typically between 20 and 50 hops are required to get to a local minimum.

At this point, my code only uses the simplest global optimization approach – searching for the local minimum in the neighbourhood of randomly generated initial (draft design) points. My observation that almost all good designs are among the best ∼ 5% designs at the end of the first optimization hop allowed me to accelerate the search by a factor of 10 by using the following strategy: after the first serial farm run, when only first hop is performed, I choose the best 5% of designs for the second stage (multi-hop) run, where true local minima convergence is achieved. The only acceptance criterion for each initial random point run is a convergence to a true local minimum. Only a small fraction of found local minima correspond to high quality designs. More complex strategies (like simulated annealing) may be tried in the future, but it is not clear if they will result in a more efficient code.

This is still a work in progress. I am currently working on implementing such lens features as internal focusing and zoom.

3. Preliminary results
Figure 5 shows one of the preliminary results obtained using my code – an optical design for a standard (focal length 50 mm) fast (f-number 1.4) photographic SLR lens. Only these two lens parameters (focal length and f-number) plus the camera mount mechanical constraints were initially specified; all the rest (number of lens elements, glass types, surface parameters, parameters of the aperture stop) were discovered by the code (ran in parallel as a serial farm on a cluster) fully automatically, without any human intervention.

4. HPC aspects
Monte Carlo based numerical codes require a high quality pseudo-random number generation. GSL implements many different random generators, with the default one, MT19937, being of very high quality: its Mersenne prime period is ∼ 10^{6000}, and it is equi-distributed in 623 dimensions [4].

Unfortunately, any pseudo-random number generator is as good as its initialization procedure, and usually employed initialization methods may not be adequate for Monte Carlo based codes executed many thousands of times in a parallel (serial farming) setting.

GSL provides an important feature which makes it possible to design an almost perfect parallel Monte Carlo setup: in addition to using seed numbers, it also allows to read and write the full state of the random number generator – a small binary file – which uniquely defines the exact position of the generator in the Mersenne prime period.

My serial farming random number generation setup consists of the following elements:

- Only once, a large number N of unique random seed numbers is generated. (N should be larger than the maximum number of jobs utilizing this algorithm one would ever run in parallel on a cluster or clusters with a shared file system). These seeds are used to initialize the random number generator, and the corresponding generator states are written to the shared file system as N binary state files.

- Any time a code utilizing this algorithm is run, the first thing it does is to borrow an available state file. The file is used to initialize the random number generator, and is made unavailable to any other job for the duration of the current job. Periodic checkpointing (updating the state file with the current state) is performed. At the end of the run (either
normal, or initiated by the KILL signal), the file is updated with the final state of the generator, and then released back to the pool of available state files.

Given the extremely large Mersenne prime period of the MT19937 algorithm, the above setup essentially guarantees that pseudo-random number sequences generated in a parallel (serial farming) setting will always be unique.

5. Conclusions
The ongoing evolution of HPC hardware has reached the point when fully automatic complex lens design became feasible. In this paper, I presented my numerical code which can be used to discover new lens designs when run as a serial farm on HPC clusters. The code is based on the algorithm for smart lens drafting, optical glass model, and parallel pseudo random number generation setup I have developed.

Acknowledgments
This work was supported by the Shared Hierarchical Academic Research Computing Network (SHARCNET).

References
[1] Smith W J 2008 Modern Optical Engineering: The Design of Optical Systems, Fourth Edition (New York: McGraw Hill) p 409
[2] Kingslake R and Johnson R B 2010 Lens Design Fundamentals, Second Edition (Burlington, MA: Elsevier) p 9
[3] Nelder J A and Mead R 1965 Computer Journal 7 308
[4] Matsumoto M and Nishimura T 1998 ACM Transactions on Modeling and Computer Simulation 8 no 1, 3