State-of-the-art of software tools for modeling x-ray optics and beamlines

M. Sánchez del Río

European Synchrotron Radiation Facility
BP 220, 38043 Grenoble-Cedex, France

ABSTRACT

Many software programs are available in the market for the design of optical instruments. However, most of them are not suitable for modeling x-ray optical elements. The simulation of the x-ray source characteristics (emission, geometry) strongly depends on the type of generator used (synchrotron insertion devices, FEL, x-ray tube, laser-generated plasma, etc.). The optical elements are usually grazing mirrors, multilayers and crystals, with very different characteristics than optical elements used in other photon ranges. In addition, the reflectivities of such optical devices must be calculated taking into account tabulated optical constants.

In the last years we have developed a new approach for creating a common tool for x-ray optics and sources modeling, lumping together small programs from different origins and authors, adding an unified x-ray database and gluing all this in a user-friendly and powerful data manipulation environment. The result is the XOP code which is now used in many synchrotron facilities. In addition to the first level calculations available in XOP, we have incorporated an interface to the popular x-ray tracing code SHADOW that allows the simulation of the complete beamline and produces accurate values of beam sizes, divergences, flux and energy resolution.

I will review the present state of XOP and the SHADOW Visual Interface. I will then present the plans for a new version in preparation. Then I will discuss new ideas and possible requirements for simulating the forthcoming x-ray optics for fourth generation x-ray sources.

Keywords: x-ray optics, x-ray source emission, x-ray databases, modeling, ray-tracing, optimisation

1. INTRODUCTION

A previous step before the construction of any x-ray system, like a synchrotron beamline or a diffractometer to be used with a conventional x-ray tube, is an accurate conceptual design of the optics. The beam should be transported to a given image plane (usually the sample position) and its characteristics should be adapted to the experimental requirements (flux, monochromatization, focalization, time structure). The designer's goal is not only to verify a minimum set of requirements, but also to optimise both the source and the optics to obtain the best possible configuration.

The first step is the choice of the type of source (synchrotron source, plasma source, conventional source, free electron laser). In many cases this is done by other constraints the need to use existing sources or budgetary constraints. In the case of synchrotron sources, it should be also defined the type of insertion device and its parameters. We will concentrate in the problem of the optical design and its optimisation. The optics should be adapted to the characteristics of the source, in order to utilize the largest fraction of the flux available, conserve the good beam properties (emittance, coherence, collimation, etc.), and match the experimental requirements, usually expressed in terms of beam size (cross section) energy bandwidth and useful flux (number of photons). The design of an optical system could be divided in three different phases:

- Drafting a collection of possible optical systems and perhaps source configurations
- Selection of the best optical system after an accurate modeling of the performances
- Optimisation of the system

Other author information: (Email: srio@esrf.fr; Telephone: +33-476 882 513; Fax: +33-476 882 542

Although an experienced scientist or engineer could perform these tasks with the only help of i) his experience, ii) the study of existing systems with similar characteristics, and iii) some simple analytical calculations, it is highly recommended to use modern computer tools. Software codes can be used to confirm the predictions made "by eye", and, more important, they permit to less experience scientists and engineers to advance in the design with the confidence that each step gives correct results and the final system will work. In other words, modeling tools give an excellent platform for virtual x-ray experiments (fast and cheap), thus they are very useful before building x-ray instruments in the real world. From the simulation point of view the three proposed design phases could be assigned to these particular steps, which will be analysed in the following paragraphs:

- Pre-calculations.
- Ray tracing of a selected configuration.
- Intensive ray-tracing. Systematic search.

2. PRE-CALCULATIONS

The first phase in the design of an x-ray system is the selection of a set of configurations (source and optics) that could satisfy the target requirements. The first step is the selection of the correct source. For synchrotron beamlines one should define the particular type of source (bending magnet, wiggler and undulator), and its main parameters (length, magnetic field, etc.). Most of these parameters are fixed in many cases, like electron energy and current. Others are limited by technological (like the magnetic gap) or room constraints (like the insertion device length). In the case of an apparatus using a conventional x-ray generator, one should specify the appropriated tube (rotating or fixed anode), target material, electron focalisation, wave filtration, etc. Then, the optics should transport and transform the beam coming from the source to a given position, where usually the sample is placed. The optics should conserve the good properties of the beam coming from the source (flux, collimations, etc.) and modify it to match the specifications in monochromaticity, beam collimation, cross section, etc. Obviously, the first ideas should come from the experience of the designer and the study of existing instrumentation. Once a first system is proposed (for example, to use an undulator source, a monochromator, and one or two mirrors for focusing the beam), calculations should be done to draft the initial parameters. At this point, simple formulas "by hand" can predict roughly the behaviour of the individual optical components. It is essential have an estimation of the main parameters in order verify that the results produced by further computer simulations are reasonable.

The x-ray source is generally described in terms of its geometrical characteristics (source size and divergences), the emission spectrum and perhaps its time structure. For a conventional source, the dimensions of the source are those of the impact region on the target (typically from 0.1 to 0.5 mm) and the emission can be considered isotropic in the useful radiation fan. The energy spectrum has a background created by brehmstrallung and some characteristic peaks which depend on the target material. A plasma x-ray source can also be considered as an isotropic source with size corresponding to the impact region (laser focused on a target). The detailed spectrum is very complicated, but it is in general divided in four parts: i) a near thermal continuum in the sub keV region (that, in a first approximation, could be assimilated to the black body emission), ii) characteristic lines at few keV, iii) an exponential decreasing spectrum originated by mixed free-bound and free-free bremsstrahllung radiation to energies of the order of 10 keV, and iv) a suprathermal tail extended to 100 keV and beyond. For a comprehensive treatment of radiation of hot dense plasmas see Ref. 1.

The size of a synchrotron source is related to the characteristics of the stored electron beam and the energy of the emitted photons. In a first approximation it is given by the projection onto the central plane of the insertion device (id) or bending magnet (bm) of the electrons travelling along the length of the id or bm. This is generally valid when the emittance of the machine is large enough to work far from the diffraction limit. For wigglers and dipoles, the vertical divergence is given by the natural divergence of the emitted photons whilst the horizontal divergence depends on an aperture slit that selects an angular window of the radiation fan. For undulators, the divergences depend strongly on the photon energy (because of the harmonic structure of the energy spectrum) and on the electron beam emittance. The bm and wiggler spectra can be easily calculated from universal curves, but numerical calculations are unavoidable in the undulator case. The spectral characteristics of the synchrotron radiation can be calculated analytically, and this is advantageous respect to other sources of x-rays (x-ray tubes, FEL), which require heavy computer simulations.

For the active optical elements (gratings, mirrors, crystals, multilayers), one can consider independently the focusing effects (usually given by the surface shape, but also by other effects like varied ruling in gratings and asymmetry in crystals) and the reflectivity. In a first stage, reflectivity considerations are more important: one should first be sure to dispose of useful photons, and then think on how to handle and focus them. For x-ray mirrors, the reflectivity depends on the surface coating. Moreover, the photons not reflected are absorbed and its power is converted basically in heat. The calculation of the absorbed power is a step previous to the design of a suitable cooling system, needed if high power load has to be dissipated. This happens very often at third generation synchrotron sources.

The main difficulty for the calculation of the source and optics parameters arises from the fact that one often needs to combine the simulations of different programs, usually from different origins and authors, which sometimes run on different platforms. We have created the XOP[2–5] package with the idea of overcoming these difficulties and present to the user a flexible, unified, and reliable graphical interface (GI) to run such programs. XOP is the result of a coordinated activity between ESRF (M. Sanchez del Rio) and APS (R. Dejus). XOP is distributed for free to the scientific community. XOP is buid using different software pieces written and made available to the users by several authors.

XOP's functionality has been expanded and now over 20 applications are available. It is used for synchrotron radiation simulations related to three different application groups:

- source simulations
- characteristics of optical elements
- multipurpose data analysis and visualization

Regarding synchrotron source simulations, XOP include programs to calculate the radiation spectra (flux and power versus photon energy) of bending magnets, wigglers and undulators. The simulations consider the parameters of the storage ring (e.g., energy and emittances) and the characteristics of the insertion device (number of magnets, period length, magnetic field, etc.). In addition to the standard flux and power calculations, the undulator codes available in XOP can also compute other quantities, such as flux densities on a screen surface and polarization characteristics, and they also consider the effect of the electron beam emittances.

The optics codes implemented in XOP can be classified into four groups:

- DABAX (DAtaBAse for X-ray applications) interface and related codes. A web server for DABAX is also available.[6]
- mirror reflectivity and filter transmitivity, with the possibility to be coupled with the source models.
- crystal diffraction profiles.
- multilayer reflectivities.

The main XY plotting application in XOP can also be used to visualize generic data and to perform a variety of mathematical operations (like convolution, Fourier transform or non-linear fitting).

XOP has also the possibility to include other large packages of synchrotron radiation modelling and data analysis as "extensions". These modules are installed optionally on top of the basic XOP configuration. As an example, the standard SHADOW ray-tracing code is interfaced into XOP by using a new GI.

A new XOP version is in preparation and will be released in early 2002. As new features, it will include the support of Macintosh MacOS9 and MacOSX platforms (in addition to the current Windows and Unix versions), support for neutron optics, and new extensions. A CD-ROM distribution will also be prepared, as it was done for the current version XOP 2.0.

3. RAY-TRACING

The first phase in the design of an optical system (pre-calculation) permits to define one or few initial configurations that have to be analysed more in detail. This is usually done by ray-tracing. A ray-tracing simulation requires to define every optical parameter of the beamline (e.g., incident angles, coating, slit apertures, etc). Consequently, a complete ray-tracing study usually requires several runs to verify the behaviour of the optical system under different situations, like different photon energy and different values of the "moving" elements (e.g. aperture slits, mirror incident angle, monochromator rotation, etc.).

The ray-tracing technique is a very powerful approach for predicting the performances of an optical system. It can be used to obtain accurate and reliable response parameters of a complete optical system (source and optics). For optical systems in the visible, infrared and UV range, several commercial packages are available. For synchrotron radiation applications, the code SHADOW[7] has become the *de-facto* standard. In this ray-tracing code, a ray is a mathematical entity fully specified by four vectors and two phases: the starting position $\vec{r}$, the momentum $\vec{k}$, the electric fields $\vec{A}_\sigma$ and $\vec{A}_\pi$ (σ and π refers to the perpendicular and parallel components of the polarization, respectively) and their phases ϕ_σ and ϕ_π. Therefore, the SHADOW space variable is a superset of the space of variables used in other graphical approaches for studying optical systems (e.g., the phase-space method). The source is created using the Monte-Carlo method to sample rays with the spatial, angular and energy distribution of the synchrotron sources. In addition, geometrical and grid sources can also be created. Every ray from the source is traced through an optical system consisting of a number of optical elements (mirror, gratings, crystals, etc.) The important values of beam cross section, energy resolution, etc. are calculated by post-processing (histogramming, integration and visualization) the resulting SHADOW files.

An optical surface can be defined in SHADOW using internal defined shapes: plane, parabolic, hyperbolic, ellipsoidal, toroidal and polymomial. It is also possible to define a surface from a mesh consisting on a numerical set of height values (Z) on a regular XY grid. Reflecting mirrors are computed in SHADOW by applying the specular reflection laws for calculating the output direction (geometry model) and applying the Fresnel formulas for weighting the ray with a reflectivity value (physical model). SHADOW includes a database of the photon-matter scattering factors, necessary to compute the refraction index, in the energy range 10 to 100000 eV. This compilation is based on the works by Henke[8] and Sasaki.[9]

Two crystal models are available in SHADOW: mosaic and perfect crystals. The mosaic crystal model is described in Ref. 10 and some calculations compared with experimental data are presented in Ref. 11. The SHADOW model for perfect crystal diffraction is conceptually very simple: In the diffraction process, the momentum of the ray is changed following two assumptions: i) the elastic scattering, and ii) the continuity of the tangential components of the wave vectors through the surface. In addition, electric fields are changed according to the equations of the dynamical theory of the diffraction. Many examples of perfect crystal calculations are shown in Ref. 12. SHADOW also includes models to calculate multilayers, gratings, zone plates and other usual optical elements.

An important part of the ray-tracing calculation is the study of the tolerances to source movements, alignments, sample displacements, etc. SHADOW is well suited for these calculations because it allows to freely displace the source and optical elements whilst conserving the same initial reference frame.

We have incorporated into SHADOW a new user interface (SHADOWVUI) that allows to run SHADOW using a multi-window environment. This allows to the user to modify the optical system very easily, rerun SHADOW with modified inputs and quickly refresh all the screens showing interesting information for the user (XY plots, histograms, etc.). A new powerful feature in SHADOWVUI is the possibility to incorporate macros. These macros permit to run SHADOW in a loop, perform powerful post-processing, make parametric calculations, and compute *a posteriori* some basic operations (tracing in vacuum, vignetting, etc). A beamline viewer application (BLViewer) has been developed to create three-dimensional schematic views of the optical system. This will help the user in verifying graphically the correctness of some important input parameters in SHADOW, like the orientation of the optical elements, which usually confuses some users. This application also permits to visualize rays traced over the whole optical system.

The SHADOW code is now about 20 years old, and although it still performs a very valuable work, it suffers from some problems. SHADOW's developers[13] are planning to rewrite completely the code. The new structure will overcome many technical and physical limitations of the current version. From the physical point of view, it

is important to include in the simulations coherent beams, and the effect of the optical elements on them. This is important for many tecniques already in use (phase contrast imaging) and will be essential for fourth generations sources, which will be completely coherent. For fourth generation sources, the simulation of the source characteristics, including the time structure require complicated numerical simulations. In addition, the effect of optical elements (like crystal dynamical diffraction) modifies the time structure of the beam.[14] The inclusion of the coherence and time structure effects in SHADOW is a big challenge. I present in the appendix some ideas about new features that could be implemented in the new code.

4. OPTIMISATION

The optimization of an optical system could be defined as the process to find a new optical configuration (usually starting from a given one) that fits better the user requirements, in terms of performances (photon flux, etc), simplicity of the design, and cost. In most cases, optimisation is done "by hand". This means that the designer analyses different configurations starting from a given one. This is an iterative process in which one analyses the dependency of some interesting factors (like flux on an aperture) versus some beamline parameters (like mirror shape and slope error, for example). The parameters, selected from the total set of parameters that define the beamline, are changed manually in the ray tracing. Therefore, the optimisation process it is very time consuming, it requires many runs of the modelling code and there is no guarantee of success. It would be highly desirable to design automatic methods to help users in this difficult task.

The concept of macros included in SHADOWVUI can be a first step in the implementation of an optimisation strategy. For instance, when studying parametric calculations (the variation of one parameters versus a beamline variable), one should run SHADOW many times, one for each point of mirror length and tabulate the resulting flux. This can be done automatically in few lines of macro code and can then save a lot of time to the designer. An extension of a parametric calculation is an *exhaustive search*, where the extrema of a function is found by evaluating the function in the whole argument space. One usually has many arguments and wants a domain finely sampled. In most cases, this exhaustive search would require unfeasible calculation times.

It would be more interesting to use SHADOW as the calculation engine inserted in an optimisation loop. A first step is to define a magnitude (fitness function or figure of merit) that should be minimised during the optimisation process. This figure of merit (fm) could be, for example, the integrated flux, or better density of flux (flux divided by the image size, etc). The implementation of general concepts into a single mathematical multivariate function (figure of merit) is extremely delicate. Moreover, this function has many arguments (a subset of the SHADOW inputs) of different nature (float, integer, list of options). The characteristics of the fm in terms of smoothness, differentiability, divergence, and boundary values should also be considered. Thus, the resulting fm is a non-continuos function of many variables and it is suppose to have many local minima. These facts make impossible to use "standard" (i.e., local optimisation) methods based on the directional search of a better solution of parameters starting from an initial guess. These local optimisation methods are only useful if the fm is smooth and does not present oscillations as a function of the parameters. It is then necessary to use global optimisation methods, which are intended to search the parameters space in order to find better solutions. The most used methods of global optimisation are simulated annealing and genetic algorithms.

Simulated annealing[15] is a stochastic approach for minimizing multivariate functions. The term simulated annealing comes from the physical process of heating and then slowly cooling a substance to obtain a strong crystalline structure. Genetic algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. Genetic algorithms are inspired by Darwin's theory about evolution. We have used successfully genetic algorithms to fit simulations of multilayer reflectivity to experimental data.[16] We also made an attempt[17] to incorporate a Global Optimiser on SHADOW, using either genetic algorithms or simulated annealing. Global optimisation methods can be efficient to find the neighbourhood of a minimum, but they are usually not very good in finding its exact value. The global optimisation methods could be improved using hybridisation (coupling genetic algorithms, for instance, with standard local optimisation techniques).

Although global optimisation is not feasible with the present version of SHADOW because its heavy structure, it is worth to consider these methods for their inclusion in the projected new version of SHADOW.

5. CONCLUSIONS

I have sketched the process of designing an x-ray optical system in three phases, drafting, ray-tracing calculations and optimisation. In the first phase, in addition to general analytical formulas, the XOP code could be used to calculate main parameters an optical system, like source spectra, optical element reflectivity and simplified heat load estimations. The ray-tracing code SHADOW is at present the most adequate ray-tracing engine for x-ray applications, and the graphical user interface SHADOWVUI, available in XOP, is a very convenient tool to make easier and more flexible the ray-tracing calculations. Some ideas about optimisation are given. No automatic optimisation procedure exists at present. The codes presented could be very useful for aiding scientist in the design of a beamline, but it is clear that they can be improved and extended. It would be desirable a coordinated international collaboration for providing to the scientific community with accurate and modern computer codes for computer aided beamline design and optimisation.

6. Appendix. Ideas for an improved version of SHADOW

This appendix describes some requirements and ideas that could be considered for the design of a ray-tracing program. The ideas sketched here could serve as an starting point for the upgraded version of Shadow.

Shadow has been for almost 20 years the *de-facto* standard code for synchrotron radiation calculations. The persistence of a computer code for such a long time is a good prove of its quality. They are several facts, in my opinion, that contributed to this fact:

- Modularity. SHADOW is separated in two main program, one for the generation of the source *gen_source* and the second one for tracing the source through the optical system *trace*. Their input and output are standarized in two main concepts: a list of variables dumped into a file (usually called *g-file*) that define the inputs of *gen_source* and *trace*, and the binary files containing the collection of rays forming the beam beam files at different positions along the beamline (*begin.dat, star.xx, mirr.xx* and *screen.xxyy*).
- Iterative (sequential) tracing of the optical elements defined individually forming all together the optical system (beamline or instrument).
- Independent satellite programs to visualize and post-process the resulting beam files.
- Accurate database of optical constants, which is the base for *ab-initio* modeling a large collection of elements.
- Definition of a ray as an array of several (up to 18) variables, thus permitting to define physical entities like intensity, polarization, energy, etc.
- Stochastic creation of the source (in addition to grid models) using the Monte Carlo inverse method.
- Deterministic tracing. Every ray from the source is traced to the sample plane trough the optical elements. The effect of finite reflectivity or transmitivity of an optical element is expressed in a weight (or intensity, derived from the electric fields) and not in the elimination of rays according to its probability of being reflected or transmitted. This important concept allows simulations of most systems using only few thousand rays.
- Use of vector algebra to operate with rays and transform efficiently from one reference frame to another.
- Portability: Shadow is written using standard languages and is available under many Unix and Windows platforms.

Some facts that became obsolete and must be improved.

- Number of rays limited at compilation time.
- Hard coded databases (optical constants) which cannot be easily modified.
- Difficulties to define and code new optical elements.

- Difficulties for using it as a callable library, or better from an scripting language, for being used inside a loop or for automatic scoring and optimisation.
- Large number of files written out.

Following Shadow's model, a complete ray-tracing code should be separated several modules. The most important is the ray-tracing engine (RTE), or code to perform the ray tracing calculations. Other modules should serve as pre and post-processors, visualization applications and graphical interface. An additional module could be an optimisation tool.

The RTE should be a standalone program completely independent from the graphical interface (GI). The description (input parameters) of the ray-tracing components (source, optical elements and related screens) should be kept in an independent entity (object, structure, or SHADOW's *start.xx* file). The source code should be freely available and existing tools (like *sourceforge.net*) can be used to coordinate different versions. The program should not only trace a "bunch" of rays (i.e., a fixed number of rays, like the present SHADOW) but also trace ray per ray until some condition (e.g., good statistics) is fulfilled. For that two steps are considered independently. The first step is the calculation (*setup*) of the parameters for the optical elements (e.g, positioning a monochromator with the correct Bragg angle given the photon energy, or bend a mirror with a radius of curvature calculated using the focal distances and incident angle), which could be compared to an "alignment" of the beamline. A second step is the tracing of the ray or bunch of rays through the system. At present, SHADOW does these two steps iteratively (element by element). It should be better to first align the whole beamline, and then trace the single ray or bunch of rays.

One should also consider that ray tracing calculations in future could require hundreds of millions rays, so a program easily parallelised to be run on cheap and powerful Linux clusters (using for example the Message Passing Interface[18]) is desirable. It would be interesting to define an input format for describing the optical element surfaces, in order to easily import the outputs from finite element computation and may be CAD files. There is also some interest in computing non-sequential systems, for example for x-ray telescopes. This would require a completely new structure of the code. In my opinion, the main code should be kept "sequential", but it should be implemented a flexible way of introducing new optical elements containing, for example, a collection of optical surfaces like nested mirrors.

Another important issue is the computer platform and programming language. It should be written in a language easily portable to all popular platforms (Unix, Windows and Macintosh). It would be desirable to use a compiled program (Fortran or C) for efficiency reasons. However, there is the drawback of portability: it should be compiled for every system. A code written in a scripting or interpreted language is easier to maintain, and in many cases easier to program. One can take profit of the sophisticated programming features (like matrix operations, vector algebra) available in commercial programs, like MatLab[19] or IDL.[20] There are other free alternatives that could be considered, like SciLab[21] or Python.[22] Interpreted codes are usually much slower than compiled ones for large loops, but this problem could be overcome by using intelligent vector calculations. Object oriented programming could be a good solution, but a well organised structured code would also be a good choice. It is necessary, however, to make exhaustive testing before making the crucial choice of the programming language. A "free" language should be preferred, but in the case where there are not sufficient human resources for performing a large computer project, it could be more convenient and less expensive to pay licenses for commercial software that make quicker the software development process.

The work for the design and implementation of a graphical interface (GI) adapted for a ray-tracing program is of the same magnitude than the programming effort required for the ray-tracing engine. In addition, a graphical interface implements a large set of tools not needed in the ray-tracing engine, like graphical libraries, widgets, etc. These tools evolve very rapidly on time, and a solution chosen today requiring a given amount of work, may be perhaps implemented in a short future with much less effort. In addition, many useful packages for creating high performance graphics are not freely available, and our human resources should not be employed to write generic tools from scratch. Therefore, it is convenient to separate completely the graphical interface from the ray-tracing engine. In such a way, it would be always possible to use the bare ray-tracing engine without user interface and to rewrite the graphical interface if new tools appear.

The ray-tracing code must be written taking into account that a GI layer will be on top of it in order to define communication mechanisms to exchange information between the ray-tracing engine and the GI. This exchange of information could be done at three levels:

- Disk files. This is the simplest case of exchange of information. The GI presents widget menus to the user and receives the information from her. This information is preprocessed if needed (e.g., changing variable units, etc.) and written to a disk file in a format that the ray-tracing engine can accept. Then the GI starts the ray-tracing engine (spawns a process). the ray-tracing code runs the system defined in this input files and writes the results to other files. During this running time the GI is idle and cannot accept inputs from the user. When the ray-tracing engine finishes, the user can request the GI to post-process and visualize the resulting files.
  Pros: the communication by disk files is very simple and only two signals are exchanged: the order sent by the GI to start the ray-tracing engine and the termination signal. It is very portable between computer platforms.
  Cons: Writing and reading disk files is a slow and inefficient way of communication. It also requires a high disk space and tools to clean it. Another big problem is the error management. If the ray-tracing code gives errors, these are not tracked to the interface. The currently available GIs for SHADOW 2.0 (ShadowGUI and ShadowVUI) use this mode of communication.

- Using a shared memory area. The information to be exchanges between the ray-tracing engine and the GI can be allocated in a memory area that should be accessible from both the GI and the RTE. The GI and RTE are separated processes that run in the same computer, and both read and write information in the shared memory. This area is static, like common blocks or global variables.
  Pros: it avoids the use of disk files. It is not difficult to code in C or in other languages that can talk to C.
  Cons: two independent processes are running, so it is more difficult to manage than a single one. In order to send a signal from one process to another, a shared flag variable that is continuously checked from the other is needed. Therefore, at least one process, if not both, must be in a permanent loop checking the state of some communication variable flags.

- Dynamic allocation of memory, and communication via pointers. The most efficient way of communication should be via pointers to a memory area (like a structure containing some inputs or the beam). In such a way, the GI could receive the data from the user, and put them in a structure. The GI would pass a pointer to this structure to the RTE, which would perform the calculations and pass back to the GI a pointer to the beam(s). Then the GI can access the memory zone allocated by the RTE, so it can access directly the arrays with the beam without making its own copy.
  Pros: high performance and flexibility.
  Cons: the programming languages used for both the GI and RTE must allow structures, pointers and dynamic allocation of memory. Moreover, they must be compatible. This reduces the choice for programming languages, and, may be, compilers and platforms. It is much harder to code.

A last issue is whether to recycle old pieces of code or to rewrite everything from scratch. In my opinion, the reuse of existing codes for the RTE (SHADOW) and for the GI (SHADOWVUI) is essential unless a large and well organised team of programmers would be available. New programs from scratch have the additional problem of the lack of confidence from users: they still prefer to use "obsolete" codes benchmarked by tens of publications rather than "beta-testing" modern software.

ACKNOWLEDGEMENTS

I are indebt to Franco Cerrina (author of SHADOW) for his enormous contributions to the software discussed here, and to Roger Dejus, coauthor of XOP. I acknowledge helpful discussion and encouragement received from Andreas Freund. I also want to thank people currently working in several aspects related to the XOP project: Emmanuel Marie (DABAX web server), Cyril Poirier (BLViewer), Corentin Jouan (Macintosh port), and Christophe Boisseau and Lucia Alianelli who are developing XOP for neutrons. I extend my gratitude to people who wrote programs, pieces of code and tabulated data, and made them available to us for being included in XOP.

REFERENCES

1. D. Attwood, *Soft X-rays and extreme ultraviolet radiation: principles and applications*, Cambridge University Press, Cambridge, 1999.
2. R.J. Dejus and M. Sanchez del Rio, "XOP: A graphical user interface for spectral calculations and x-ray optics utilities," *Rev. Sci. Instrum.* **67-9**, pp. CD–ROM only, 1996.
3. M. Sanchez del Rio and R.J. Dejus, "XOP: A multiplatform graphical user interface for synchrotron radiation spectral and optics calculations," *SPIE proceedings* **3152**, pp. 148–157, 1997.
4. M. Sanchez del Rio and R.J. Dejus, "XOP: Recent developments," *SPIE proceedings* **3448**, pp. 340–345, 1998.
5. **http://www.esrf.fr/computing/scientific/xop**.
6. **http://www.esrf.fr/computing/scientific/dabax/**.
7. C. Welnak, G.J. Chen and F. Cerrina, "SHADOW: A synchrotron radiation and x-ray optics simulation tool," *Nucl. Instr. Meth.* **A347**, pp. 344–347, 1994.
8. B.L. Henke, E.M. Gullikson, and J.C. Davis, "X-ray interactions: photoabsorption, scattering, transmission, and reflection at E=50-30000 eV, Z=1-92," *Atomic Data and Nuclear Data Tables* **54**, pp. 181–342, 1993.
9. S. Sasaki, "Numerical tables of anomalous scattering factors calculated by the Cromer and Liberman method," *KEK Report,* **88-14**, pp. 1–136, 1989.
10. M. Sanchez del Rio, S. Bernstorff, A. Savoia and F. Cerrina, "A conceptual model for ray tracing calculations with mosaic crystals," *Rev. Sci. Instrum.* **63**, pp. 932–935, 1992.
11. M. Sanchez del Rio, M. Gambaccini, G. Pareschi, A. Taibi, A. Tuffanelli and A. Freund , "Focusing properties of mosaic crystals," *SPIE proceedings* **3448**, pp. 246–255, 1998.
12. M. Sanchez del Rio, "Ray tracing simulations for crystal optics," *SPIE proceedings* **3448**, pp. 230–245, 1998.
13. F. Cerrina, private communication.
14. S. D. Shastri, P. Zambianchi and D.M. Mills, "Femtosecond x-ray dinamical diffraction by perfect crystals," *SPIE proceedings* **4143**, pp. 69–77, 2001.
15. S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing," *Science* **220**, pp. 671–680, 1983.
16. M. Sanchez del Rio and G. Pareschi, "Global optimization and reflectivity data fitting for x-ray multilayer mirrors by means of genetic algorithms," *SPIE proceedings* **4145**, pp. 88–96, 2001.
17. C. Boeddicker and M. Sanchez del Rio, "Global optimisation of optical systems," *Internal trainee report* **http://www.esrf.fr/computing/scientific/raytracing/PDF/GlobalOptimization.ps**, 1994.
18. **http://www-unix.mcs.anl.gov/mpi/**.
19. **http://www.mathworks.com**.
20. **http://www.rsinc.com**.
21. **http://www-rocq.inria.fr/scilab/**.
22. **http://www.python.org**.