Astronomical Data Analysis Software and Systems IV
ASP Conference Series, Vol. 77, 1995
R. A. Shaw, H. E. Payne, and J. J. E. Hayes, eds.

The SAO AXAF Simulation System

D. Jerius, M. Freeman, T. Gaetz, J. P. Hughes, and W. Podgorski

Smithsonian Astrophysical Observatory, 60 Garden St., Cambridge, MA 02138

Abstract. As part of our efforts to support the *AXAF* program, the SAO *AXAF* Mission Support Team has developed a software suite to simulate the *AXAF* telescope. The software traces the fate of photons through the telescope, from the X-ray source through apertures, baffles, the telescope optics, and finally to the photons' ultimate interactions with the focal plane detectors. We model relevant physical processes, including geometrical reflection, scattering due to surface microroughness, distortions of the optics due to the mirror mounts, attenuation through baffles, etc. The software is composed of programs and scripts, each specialized to a given task, which communicate through UNIX pipes. Software tasks are centered about functional components of the telescope (e.g., apertures, mirrors, detectors) and provide a comfortable and flexible paradigm for performing simulations. The use of separate programs and the UNIX pipe facility allows great flexibility in building different configurations of the telescope and distilling diagnostics from the photon stream through the telescope. We are able to transparently use symmetric multi-processing (e.g., SPARCStation 10s and SGI Challenges) and can easily use sequential multi-processing (via workstation clusters). Some of the tasks are amenable to parallel processing and have been implemented using the MPI standard.

1. Introduction

As part of SAO's support of the design and calibration of the Advanced X-ray Astrophysical Facility (*AXAF*), we have undertaken the task of simulating the optical performance of the telescope. We seek to create a high fidelity model which takes into account the major factors which will impact the optics' X-ray performance. These include distortions of the optics due to mechanical and thermal stresses in the mounting assembly, manufacturing figure errors in the optics, and scattering due to mirror surface microroughness and surface contamination.

We have a far-ranging set of investigations which we are, and will be pursuing with this system. These investigations range from studies of requirements for calibration, to studies of the impact of detector characteristics upon the observation of astrophysical sources. Among the issues which we are currently investigating are: (1) the ability of the telescope, as designed, to achieve its performance goals; (2) the performance of the mirror assembly during testing at the Marshall Space Flight Center X-Ray Calibration Facility; (3) the development of models for the on-orbit science performance of the telescope based upon ground calibration data; (4) the performance of flight instruments; and (5) the efficacy of ground and on-orbit calibration procedures.

2. System Design

The diverse set of studies which we must perform and the goals of the *AXAF* calibration effort require of our software system a high degree of flexibility and efficiency. Because of limited human resources, we have striven for the former, while keeping open our options for the latter.

AXAF is a complicated system, and the mirror assembly and assorted structures comprise a highly complex set of obstructions, apertures, and reflective surfaces. In order to understand how the various components affect each other, and to diagnose problems (both with our software and the hardware designs), we must be able to insert or remove routines which model the apertures, baffles, collimators, and detectors, or perform diagnostics, at any point in the photons' path. At the same time, we must retain enough efficiency so that we can trace $\sim 10^7$ rays through the system on a routine basis with a reasonable turn-around time. To cap it off, we need to be able to encapsulate existing software which may not follow our final paradigm. For example, our surface reflection module is based upon a mature FORTRAN program written in the days of mainframes and minimal core memory.

One possible approach is to develop a single, monolithic, program, which follows a ray from generation to collection. This avoids interim output (on the order of hundreds of MB) and is quite efficient. We rejected this approach early on, however, because it would require a complicated scheme to control the various system components and allow us the flexibility we desire. From a software development point of view, it forces a recompilation and re-verification each time a new component is added. In an environment composed of scientists, engineers, and programmers with different levels of programming sophistication, ensuring the integrity of such a system would be a nightmare.

Instead, we chose to model each optical or structural component as a separate program, based on the UNIX filter paradigm. A program reads a ray from its standard input stream, manipulates it and writes it to its standard output stream. The rays are encoded as binary data in dynamically extensible records, allowing routines to add more information to the ray packet, as necessary. The programs use a common user interface (based upon the IRAF parameter interface) with parameter storage. As programs are completely independent from each other, validation and verification are quite straightforward. Programs communicate with each other via UNIX pipes, which are easily accessible via the standard UNIX shells. We provide higher level control by embedding the commands in UNIX shell or Perl scripts. We have created a Perl library to easily interface our Perl scripts to the parameter interface, presenting a uniform environment to the user.

In some instances efficiency has won over flexibility, and we have combined several "atomic" operations into one program. One example of this is the program which simulates apertures. At the front end of the mirror assembly is a set of thermal baffle plates, which act to control the thermal environment inside the mirror assembly. It consists of some nine circular plates, with annular apertures to let the photons pass through to the mirrors. Rather than model each plate with a separate program, our aperture program can deal with any number of plates. In addition, we have designed and are prototyping an aperture program which uses an embedded language and a flexible script-driven approach to assemble complex apertures from simple building blocks.

3. Components

As an illustration of how the individual programs work together to model the telescope, we present a short synopsis of the major modules, in the order of their affect upon an incoming photon.

3.1. Ray Generation

The ray generator spews forth rays from a variety of sources, constraining the ray directions to fully illuminate the entrance aperture of the telescope. Sources may be placed at any location in the telescope's field of view, and may have any of several simple geometrical shapes. They may also be diffuse, simulating all-sky background radiation or extended X-ray sources.

3.2. Apertures

The aperture module, as noted above, is used to model both optical and thermal baffles, support structures, and the like. Surfaces may be partially transparent (via input of an absorption depth), and may be composed of struts and annular plates. Complicated apertures will be possible with our new hierarchical aperture module, which can use overlapping geometrical shapes and "cookie-cutters".

3.3. Surface Reflection

The surface reflection program is based upon a NASA routine (the "Optical Surface Analysis Code", or OSAC). OSAC can handle deformed optics, with deformations approximated by Fourier (azimuthal) and Legendre (axial) polynomials. We have built upon the deformed optics module (which we have renamed as SAOsac), integrating it into our filter-based environment, and have extended it to use bi-cubic splines as well polynomials. Deformation in the optics is caused by loads at the mirror support points, due either to thermal or mechanical stresses and (in the case of ground calibration) a one-g gravitational field. The deformations are modeled by a highly detailed finite-element model of the mirror assembly. An IDL program (TRANSFIT, Freeman 1993) was written which takes output from the finite-element modeling code (ANSYS) and interactively fits the distortions in the optics.

3.4. Scattering due to Surface Microroughness and Contamination

After a ray's reflection from an optic has been determined, high frequency mirror roughness (surface microroughness) is modeled by scattering the ray according to a statistical model of the surface roughness. The model is based upon the measured point-spread distribution (PSD) of high-frequency surface height fluctuations of the optical surfaces. A similar process is used to model scattering due to surface contamination.

3.5. Ray Accumulation

There is a bevy of detectors which will be used both on orbit and at ground calibration, including gas proportional counters, CCD's, solid-state spectrometers, and microchannel plate detectors. We have developed models of these detectors to provide us with end-to-end predictions of the telescope performance. In one case (modeling scans of the ray distribution through circular apertures) we were able to parallelize the aperture-detector combination (using the MPI standard), allowing us to gang several workstations together (Nguyen & Hillberg 1995).

4. Conclusion

This system permits an essentially unlimited numbers of rays and all of the flexibility that one could desire. At any point between programs we can siphon off rays for diagnostics or analysis. Manipulating configurations is simple and intuitive. Writing detector simulators or diagnostic programs is straightforward. As an added advantage, when running on multiprocessor computers with symmetric processing (e.g., Solaris 2 or IRIX 5), the system will automatically switch processes to unused CPU's. For CPU dominated calculations, we can chain multiple machines together, each operating on one part of our process chain and passing the result on to the next machine.

The paradigm has proven highly successful, enabling us to provide high precision results without a complicated setup procedure to construct the particular configuration of the telescope that we desire. The efficiency (even allowing for the overhead of the inter-process communications) is reasonable, and on modern workstations has not a proven a barrier to our need for rapid turn-around.

References

Freeman, M., 1993 TRANSFIT - Finite Element Analysis Data Fitting Software, Interim Report SAO-AXAF-DR-93-052 (Cambridge, SAO)

Nguyen, D., & Hillberg, B. 1995, this volume, p. **??**