Astronomical Data Analysis Software and Systems VIII
ASP Conference Series, Vol. 172, 1999
D. M. Mehringer, R. L. Plante, and D. A. Roberts, eds.

Automated Programming and Neutrino (Astro)Physics

Michele Maris

Astronomical Observatory of Trieste (OAT), Via G.B. Tiepolo 11, I34131, Trieste, Italy

Abstract. A project for the creation of a code for accurate numerical predictions for solar neutrino oscillations is presented. The project was realized through an automated program generator which scans the simulation parameter space. The software validation procedure and the production guidelines are illustrated.

1. Introduction

After the recent Super Kamiokande announcement of a convincing detection of neutrino mass (Kajita 1998), there has been increasing demand for exhaustive and accurate numerical simulations of neutrino oscillations in (astro)physical environments. Large numerical simulations are a fundamental tool when the numerical accuracy has to be compatible with the experimental one. From the point of view of computer work, two of the most computationally intensive tasks are the determination of neutrino oscillations parameters and the prediction of results of new experiments. Both require an "exploratory" scan of the full parameters space composed by: the source model (Sun, accelerator beam or cosmic rays), neutrino parameters (the neutrino mixing angle in vacuum θ_v and the difference of square masses Δm^2), the oscillation mechanism (MSW, Vacuum Oscillations, etc.), and the neutrino experimental set-up.

The COALA project[1] is designed for the creation and management of a large data base of simulation to be used in the analysis of solar neutrino data under the assumption that solar neutrinos are affected by matter enhanced oscillations or vacuum oscillations during their propagation. The project is presented as a practical example of application of automated programming, both for simulations management, results analysis and results display in a theoretical field where exploratory simulations are essential.

2. Software Development and Simulations Execution Guidelines

Formal guidelines where adopted along the software production cycle and its subsequent employment. The experience gained during this stage of the project suggested a more general, abstract and automated procedure, leading to the concept of Automated Program Generator (Maris & Staniszkis 1999).

[1] `http://www.pv.infn.it/~maris/nuesite/coala.html`

The code for the COALA project is written in standard FORTRAN 77, managed by the CERN `CMZ` software[2] and is based on a previous version written in Borland `TurboPascal 4.0` for a 486 – 33 MHz PC (Bertotti & Maris 1996). A set of coherent rules, mostly related to names composition, was applied rigorously to improve the readability and reduce the documentation. Variables and function names are composed by a set of prefixes, postfixes and keywords organized as a dictionary, in which describes a specific program feature or physical quantity.

A four stages validation chain was adopted and its first stage is the *FORTRAN* validation where formal correctness of the code and the absence of run time errors is ensured. In the *DEBUG* validation step the code is observed while working through a debugger and test messages. In the *NUMERICAL* validation step the code is checked with the results produced by a comparison code. In this step any critical functions of the `FORTRAN` code are replicated by an equivalent one written for `Mathematica`, the two functions are then quantitatively compared within a significant set of sample values of the input variables, and a validation document is written. In the *PHYSICAL* validation step the results are studied to be sure of their physical consistency. This scheme ensures the numerical correctness and stability of the final code (Maris 1997) and a `Mathematica` package for neutrino (astro)physics was produced.

The compiled code is simplified by suppressing all `FORTRAN` user interfaces. No string parsing or complex input management is performed by it. The FORTRAN code, executed in batch mode, receives all the data from the standard input, and produces only one simulation at each call. Due to such limitations all services required by the FORTRAN code are given by a hierarchy of script files which operate as a *Virtual Scanner*. From the top to the bottom of the hierarchy there are script files which either generate the script files which form the underlying level, or execute the compiled FORTRAN code or the already generated script files for a specific simulation. In this way the user interface and the services are isolated from the computation, the `FORTRAN` code is both standardized and simplified and the effort to port, maintain, update and reuse the code is reduced and many simulations can be generated in a fast and effective way.

The `FORTRAN` programs are chained through the script files hierarchy to form a simulation pipeline. The data exchange is ensured through files, stored into a database of intermediate results. Each of them carries both the data and the parameters required to generate it. In this way each simulation parameter is introduced in the pipeline only once, and the files are promptly identified by their header. The creation of an archive of intermediate results increases the disk load but it is fundamental for the *NUMERICAL* and the *PHYSICAL* validation steps, since they are used to ensure the quantitative consistency between the various simulations steps. In addition, new investigations not planned in the original research plan may be performed in a short time. Numerical results are stored in `gzip` compressed ASCII files. Up to 80% compression efficiency was achieved, which means that the final archive does not exceed the dimension of a binary archive, while the use of ASCII files ensures portability and allows

[2] `http://www.cn.cern.ch/cmz/`

scanning of the data without a special browser. To avoid the production of wrong results due to the failure of some steps in the pipeline, each program generates its output file only at the end of its task. Subsequent programs interpret the lack of a file as a failure of one of the previous steps.

Simulations descriptions, data, scripts and log files are stored into a database to improve the software reusage, the simulation documentation and to allow the reconstruction of the simulation chain when needed. For this purpose simulations are classified according to the execution date and their scopes.

Whenever possible standard and automated results presentations and analyses are applied, and once again script files generators play a central role. In this case scripts programs drive plotting packages, table formatters, or various kinds of software agents which, as an example, compile statistics about data stored in the data base.

3. The Virtual Scanner

The simplest model for a program which can explore a mesh over a given parameter space is the *Virtual Scanner* (*VuScanner*). In its simplest form a VuScanner linearizes the cartesian product of the mesh which generates all the sets of possible parameters combinations and runs the specific simulation process on them. Constraints are applied to skip forbidden combinations of parameters.

A VuScanner is composed by a set of nested loops with the same structure. Each loop is in charge of scanning the list of values associated to a given parameter, the loops are nested according to the parameters space hierarchy. The most external loop scans the first parameter in the hierarchy, the next loop the second parameter, and so on. The user links his own routines in the loops structures only at predefined points.

At the beginning of the project for any application a specific VuScanner was written, but a general purpose VuScanner based on MatLab 4.0 is presently in use. Here, a recursive algorithm is applied to scan a table which reports the list of parameters to be scanned, their values, the constraints and the sequence of user routines to be linked to the VuScanner. This VuScanner may be used both to generate a script file which will be subsequently executed (code generation mode) or to drive the execution of codes directly (simulation driving mode). The more recent MatLab VuScanner runs directly code on a PC with a Pentium II processor and generates script files for VMS-DCL, MS-DOS batch language, MatLab and in the future for UNIX. Moreover such version of VuScanner is at the root of the IDL VuScanner described in (Maris et al. 1999).

4. Results

A FORTRAN77 package for solar neutrino (astro)physics was developed, tested in an accurate way and successfully used to explore the behaviour of neutrinos for several hypothesis of neutrino masses. The code works on VMS-ALPHA machines and on PC Pentium II without any relevant modification. During the validation phase a `Mathematica` package for the neutrino physics was created. A series of different prototypes of Virtual Scanners were created to drive such simulations and to accomplish the analysis of results. A large database of simulations (up to 2 GBytes) was generated and studied. The main scientific achievements obtained from this code are described in Bertotti & Maris (1996), Liu, Maris, & Petcov (1997), Maris & Petcov (1997, 1998), Maris (1997), Chizhov, Maris & Petcov (1998), and Dolfini et al. (1998). The most relevant of them is an accurate and exhaustive study of the Day/Night asymmetry expected for the event rates in the Super Kamiokande solar neutrino detector. This study leads to the quantitative study of the enhancement of such effect through the selection of those solar neutrinos which cross the Earth Core, which is now applied by the Super Kamiokande collaboration (Nakahata 1998).

5. Conclusions

A Virtual Scanner may be practically applied to studies in the field of solar neutrino (astro)physics. Its use reduces the manpower required to manage such complex simulations, and allows a deepest systematic study of the subject through quantitative predictions.

Acknowledgments. The author is deeply indebted with S.T. Petcov of SISSA – Trieste for fruitful discussions, with E. Calligarich, director of the Pavia INFN Section who allowed the use of the ICARUS computing facilities at Pavia University, with F. Pasian of OAT – Trieste for support and suggestions about the formalization of automated programming, with G. Schiulaz of OAT – Trieste for improvements in the english style.

References

Bertotti B. & Maris M. 1996, Ph.D. thesis of M. Maris, Pavia, DFNT

Chizhov, M., Maris, M., & Petcov, S. T. 1998, Phys. Rev. D, submitted

Dolfini, D., et al. 1998, in preparation

Kajita, T. 1998, in Proc. of the XVIII Intl. Conf. on Neutrino Physics and Astrophysics, (North-Holland: Elsevier), in press

Liu, Q. Y., Maris, M., & Petcov, S. T. 1997, Phys. Rev. D, 56, 26

Maris, M. 1997, in Proc. of the Fourth Solar Neutrino Conference, ed. W. Hampel (Heidelberg: Springer-Verlag), 342

Maris, M., Pasian, F., Smareglia, R., Maino, D., Burigana, C., Staniszkis, M., & Barriga, J. 1999, this volume, 145

Maris, M. & Petcov, S. T. 1997, Phys. Rev. D, 56, 7444

———, 1998, Phys. Rev. D, 58, 113008

Maris, M. & Staniszkis M. 1999, this volume, 470

Nakahata, M. 1998, in Proc. of the XVIII Intl. Conf. on Neutrino Physics and Astrophysics, (North-Holland: Elsevier), in press