Internal Report IASF-BO 569/2010

March 12th, 2010

PORTING A NUMERICAL CODE FOR THE SOLUTION OF THE KOMPANEETS EQUATION IN COSMOLOGICAL CONTEXT ONTO DIFFERENT PLATFORMS

P. Procopio[1], C. Burigana[1], A. De Rosa[1],
M. Genghini[1], and C. Gheller[2]

[1]*INAF-IASF Bologna, via P. Gobetti 101, I-40129, Bologna, Italy*
[2]*CINECA, via Magnanelli 6/3, I-40033 Casalecchio di Reno, Bologna*

Internal Report IASF-BO/569/2010

March 12th, 2010

PORTING A NUMERICAL CODE FOR THE SOLUTION OF THE KOMPANEETS EQUATION IN COSMOLOGICAL CONTEXT ONTO DIFFERENT PLATFORMS

P. Procopio[1], C. Burigana[1], A. De Rosa[1], M. Genghini[1], and C. Gheller[2]

[1]*INAF-IASF Bologna, via P. Gobetti 101, I-40129, Bologna, Italy*
[2]*CINECA, via Magnanelli 6/3, I-40033 Casalecchio di Reno, Bologna*

SUMMARY – In this report we describe the efforts done in order to port a FORTRAN based numerical code on different platforms. More in details, we managed three migrations: from the original platform, a VAX machine, to a computer based on Alpha processors; from Alpha based machine to a machine equipped with IBM Power 5 processors; from the IBM machine to an Intel based one. The code makes usage of the NAG numerical libraries and this turned out to be a crucial aspect during the porting, also because compilers and compilation options have a great influence on the final accuracy and performance of the code, depending on different machines. Here we report the main phases of these adaptations of the code to the new libraries and machines, with particular emphasis to S/W aspects.

1 Introduction

The KYPRIX code, capable to solve the Kompaneets equation (Kompaneets 1956) in cosmological context under general assumptions (Burigana et al. 1995), is a powerful and versatile tool for computing the evolution of a photon distribution function in different scenarios (Burigana et al. 1991a, Burigana et al. 1991b). It is a FORTRAN based code and it makes use of the NAG numerical libraries in order to solve various integrals and the Kompaneets equation itself. Indeed, the role of the libraries is crucial and a particular attention must be payed to the behavior, the performances, and the accuracy that these routines can achieve in relation to the machine they will run on.
In the last years KYPRIX is being used to study the spectral distortions of the Cosmic Microwave Background Radiation (CMBR), most recently including the effects that different recombination and reionization scenarios could have on the CMBR spectrum itself (Procopio 2009). Considering the most recent results on CMBR spectrum distortions (Mather et al. 1994, Fixen et al. 2009, Kogut et al. 2009), it is clear that we must probe the CMBR photon distribution function with unprecedented sensitivity and on a wider spectral range. A physical description of these distortions is not the scope of this report. We remember here that the accuracy and performance of the code KYPRIX are good enough to permit its usage for analyzing the improved data of at least the next two/three decades of progress. For these reason it is important to keep the code updated and performing as best as possible and also an encouragement to make it compatible with different machine architectures.
Besides the principal intent of this report, when possible, we will dedicate particular attention to the compilation options related to the NAG libraries under a more general point of view, considering different compilers and different machine architectures. Same routines can be used for different scientific purposes, so different codes written for different tasks, if using NAG library, may be cross-checked using the compiling options, compilers and CPU architectures as reported in this document.

2 First port and update of the code

The first version of KYPRIX was written by C. Burigana between the late '80s and the early '90s. It was developed on a Digital VAX machine under an VMS operating system (OS). It was strongly used for some years. Since the late '90s the code was not updated until 2004. Since then, KYPRIX was constantly implemented and kept updated.
The first big update of the 2004 consisted mainly in re-writing the formalism related to the usage of the NAG numerical libraries: from the NAG Mark 8 distribution, the code was readapted to the Mark 20 one. Considering the new cosmological scenario emerged at the end of '90s, an important physical implementation was realized: the introduction of the cosmological constant in the equations calculating the evolution of the scale factor (Procopio 2005, Procopio & Burigana 2005a).
In order to adapt the code to the new NAG release, a remarkable part of it had to be modified or newly written. About this task, a crucial step was related to the computation of the quantity $\Phi = T_e/T_r$, where T_e is the electron temperature and T_r is the radiation temperature. The continuous refresh of Φ during a typical run of the code is a unique potentiality of KYPRIX. In the first release of KYPRIX, the computation of Φ was done in a completely different way with respect to now: in fact, the NAG Mark 8 distribution the NAG source codes were delivered with the libraries and modifying the source of the D03PGF routine it was possible to recover at each time step the solution vector for the whole frequency space directly in the subroutine where the partial differential equation is defined. Thus, Φ was computed directly by NAG integration routines at each time step according to the adopted

implicit scheme. From years source codes are no longer provided with a typical license purchase, so we modified our code in order to keep active the important option of computing the evolution of Φ saving the solution vector and using it in a backward scheme. After some tests, we understood that with this new configuration of KYPRIX, it was not possible to calculate Φ newly at each time step since a segmentation fault was occurring at every try. We resolved the problem defining a quantity that permits to manage the update of Φ in terms of time-steps: in the input parameters one can specifies how often Φ must be updated. Of course, the Φ update step is at least two (or more) time steps (a value of ten time steps typically works fine. The lower limit of this parameter was derived empirically, while the upper is practically free, but it is important to adopt a time resolution fine enough to be appropriate to the physical evolution of Φ.
We had to define also a new X grid inside the D03PCF routine, because of the differences between the two NAG distributions.
All this upgrades and implementations were made on *IFP-Alpha*, a machine equipped with four DEC Alpha processors and running a 64bit Unix OS, at the Istituto di Fisica del Plasma in Milano. This one was a very stable computer on which running the code. Indeed, we largely tested KYPRIX on it, in order to understand how much further we could go with respect to the old version of the code. Finally, we were able to strongly increase the sensitivity of the code[1]: in terms of fractional energy injected, $\Delta\epsilon/\epsilon_i$, where ϵ_i is the energy density of the radiation field before the injection, we pushed forward from a lower limit of $\sim 10^{-3}$ to $\sim 10^{-5}$ (Procopio & Burigana 2005b).
Some factors influenced mainly this improvement: the new release of the NAG libraries, that permitted us to push further with the accuracy level, the computation time of the new processors with respect to those on which the code was developed, and the use of the NAG quadrature routine D01AJF more precise than that previously adopted. It is clear that these facts are strictly related: the more is the requested accuracy level the larger is the computation time. The accuracy level acts directly on the integration time step, determining the hugeness (or, better, the smallness) of the time step itself. This quantity enters in the code as the input parameter ACC (the accuracy level increases as ACC decreases).

3 From Alpha to IBM Power5 (and Power6)

The need for a further platform transfer came out during the first half of 2007, when the machine *IFP-Alpha* went out of service and we had to find another one on which running the code. First of all, we searched for a machine running a 64bit Unix OS, because of the performance needed in order to compute cases considering small spectral distortions on CMBR (and, obviously, also because we largely tested KYPRIX on this OS). Thanks to the agreement that our institution finalized with the Inter-University Consortium CINECA, we had the possibility to explore their resources. After communicating the requirements needed, they created us an account on the super computer SP5[2].
The machine was already equipped with all the libraries we needed. Among the FORTRAN compiler available on SP5 we chose *xlf*. From the very beginning everything seemed to work perfectly. We compiled without any option, just linking the NAG libraries.
The performances of SP5 were very high if compared to those of *IFP-Alpha*. This allowed us

[1]Before doing all these updates and implementations, KYPRIX was advantageous for computing spectral distortions characterized by $\Delta\epsilon/\epsilon_i \gtrsim 10^{-3}$, while for studying those with a smaller $\Delta\epsilon/\epsilon_i$ a different version of the code, named KUDYX (see Burigana et al. 1991b), was used.

[2]This machine is now replaced with a more powerful one, the super computer SP6, equipped with IBM Power 6 processors. It has a peak performance of just over 100 Tflops.

to push further in the level of distortions that can be treated: we reached easily the lower limit of $\Delta\epsilon/\epsilon_i \simeq 10^{-6}$. It is clear that in order to handle such little spectral distortions it is required an high level of precision in computing the Kompaneets equation, thus the accuracy parameter ACC must be set to very little values[3], such as ACC $\simeq 10^{-12} \div 10^{-14}$, according to the initial redshift of the computation (ACC is higher for higher redshift).

4 From IBM to Intel

When the machine *IFP-Alpha* went out of order, beside the platform transfer just described, we started thinking to make the code compatible with respect to more diffuse platforms, in particular Intel. The first attempt consisted in trying to set up the code on a 32 bit Linux machine, 4 CPU Xeon dual core, 2 Gigs of memory, with a Gentoo distribution as OS. Initially, we tested the efficiency of the NAG libraries on this machine, by using the supplied example codes. Tests were successfully completed, but when we run KYPRIX on the machine the results were not reasonable. Performing and analyzing several runs, we found out the reason of the wrong results: many of the integrals were badly solved. In addition to this, we got strange error messages, linking to somewhere inside the routines themselves and also to non-existing paths on the machine. It was impossible to recover any significant clue from these error messages. Finally, we stated that high accuracy demanding operations, were not well performed on that machine and so our code could not run properly on it.
In 2009, we started the porting of KYPRIX on a new machine: a multi-core multi-nodes 64 bit Intel based Linux cluster, with 8 GBytes of ram per node, 2.33 or 2.50 GHz Xeon CPU, and a Scientific Linux distribution. During the porting the code on this platform, two kinds of issues emerged. The first consisted in avoiding misinterpretation by the compiler with respect to the variables declaration. This is due to the change of the memory model from 32 bit to 64 bit architecture. As a first test we used the -i8 -r8 compiler option to force the size of the integer and of the real variables to 8 byte. This was unsuccessful, so we changed the syntax related to the variables declaration, specifying the size of each variable.
In a first time, KYPRIX was working good on *intermediate* cases, in which there were no parameters which would lead to a hard performing run of the code. On the other hand, out of these cases, the energy conservation was no longer respected as soon as the parameter y_h[4] reached a value of ~ 0.1, while for $y_h \lesssim 0.1$ the code was performing quite good (in terms of energy conservation again). Of course such a behavior is not trustable and even the results obtained in cases in which the code was running until the end of the integration are not reliable.
The second issue is strictly related to the NAG libraries. Again, we successfully performed all the test provided by NAG, but at the same time we were not able to run correctly the KYPRIX code using the needed NAG subroutines. We accomplished several tests with different compilers (PGI, INTEL, NAG, GFORTRAN) obtaining in any case the same results. We directed our efforts in understanding how the intel compiler interprets the subroutine calls and the variables declaration. Ifort is a FORTRAN90 compiler that is able to compile FORTRAN77 code too. This compiler has some special compilation options to force the use of the FORTRAN77 standard to pass the parameters to the various functions.
After different parameters combination, we found that the right compilation options for the

[3]These values are true for primordial cases or for some particular reionization case, while for intermediate cases and more general run of the code, it is possible to relax the accuracy by a couple of orders of magnitude.

[4]This parameter is strictly related to the epoch of the injected energy, it grows with redshift. For more details see Procopio & Burigana (2009).

code on the Intel based machine:

$$-\mathrm{O0}\ \ -\mathrm{g}\ \ -\mathrm{vms}\ \ -\mathrm{align}\ \ -\mathrm{f77rtl}\ \ -\mathrm{assume\ dummy}-\mathrm{aliases}\ .$$

In particular, $-\mathrm{O0}$ and $-\mathrm{g}$ are the usual options for optimization and debugging; $-\mathrm{vms}$ makes FORTRAN to use the *std hp vms*; the options $-\mathrm{f77rtl}\ -\mathrm{assume\ dummy}-\mathrm{aliases}$ force the iFort compiler to use the FORTRAN77 standard. These latter turned out to be crucial for the correct behavior of the code on Intel platforms.
Using this options the Kyprix code successfully runs on x86_64 linux machine.
We can also state that the main difference between the IBM AIX system and the Intel linux system results to be a more pedantic behavior of the IBM compiler in distinguishing F77 from F90 standards.

5 Conclusions

We successfully finalized the porting of the code KYPRIX through different platforms and environments. These includes VAX machines (original platform, disused), DEC machines, IBM Power5/Power6 and Intel based machines. All the working machines which the code was run (and is running) on have a 64 bit OS. Up to now, we did not have a successful KYPRIX run on a 32 bit machine[5].
The reliability of the code through the different systems was validated by several cross tests and runs between the different machines. The high accuracy level requested from the KYPRIX code can be also considered a further test to probe the performances at the extreme accuracy limit of the NAG routines used.
At the same time, starting from our task, we additionally examined several compilation options and their action, with respect to different compilers and different machines. In our case, the most critical aspect, was to find an option that, during compilation, could force the compiler to distinguish any possible differences of declaration between different FORTRAN distribution. This turned out to be very important, because many compilers may create degeneracies between different FORTRAN distribution and thus generating often confusion in the variable declaration interpretation phase.

6 Acknowledgments

Some of the calculations presented here have been carried out on an alpha digital unix machine at the IFP/CNR in Milano, and at the IBM SP5/512 machine and at super computer SP6, equipped with IBM Power 6 processors, at CINECA-Bologna by using some NAG integration codes. We acknowledge the support by the ASI contract I/016/07/0 “COFIS”.

[5]Actually, the only 32 bit machine tried is that one which gave us strange errors, as described in Sec. 4

Appendix A: compilation command on different machines

In order to summarize some practical rules, here we collect the command lines we use to compile our code on the different machines which it runs on.
On the SP6 IBM machine we use the *xlf* compiler, along with the following compilation options:

$$-\mathrm{O3} - \mathrm{qextname} - \mathrm{qalign} = 4\mathrm{k}\,,$$

where qextname instructs the compiler to add an underscore to the end of externally defined symbols or the specified symbols, while qalign specifies the alignment of data objects in storage, which avoids performance problems with misaligned data.
On the Intel based machine the compiler used is $iFort$ and the compilation options used are:

$$-\mathrm{O0} - \mathrm{g} - \mathrm{vms} - \mathrm{align} - \mathrm{f77rtl} - \mathrm{assumedummy} - \mathrm{aliases}\,.$$

For further comments on any of these symbols see Sec. 4.

Appendix B: computational time

As stressed in Procopio & Burigana (2009), under the same cosmological case, the computational time depends mainly on two parameters: ACC, the accuracy parameter of the D03PCF routine (discussed in Sec. 2 and Sec. 3), and the value of $NPTS$, that is the number of points of the main grid on which the Kompaneets equation is discretized by the NAG routine.
For a typical case involving cosmological reionization (integration starting at redshift ~ 30) the CPU time does not usually exceed 20 minutes, setting ACC $= 10^{-13}$ and $NPTS$ at its maximum value (36001), on the SP6 machine. While for very primordial case (integration starting at redshift $\sim$ *some unit* $\times 10^5$), holding the same ACC and $NPTS$, the CPU time rise to around 10-12 hours, on the same machine.
Holding the same input and integration parameters, the time for the same runs made on SP6 increases of a factor 2-3 on the Intel based machine.

References

[1] Burigana C. et al. 1991a, A&A, 246, 59

[2] Burigana C. et al. 1991b, ApJ, 379, 1

[3] Burigana C., De Zotti G., Danese L. 1995, A&A, 303, 323

[4] Fixsen, D. J., et al. 2009, arXiv:0901.0555, Submitted to ApJ

[5] Kogut, A., et al. 2009, arXiv:0901.0562, Submitted to ApJ

[6] Kompaneets A.S. 1956, Zh. Eksp. Teor. Fiz., 31, 876 [Sov. Phys. JEPT, 4, 730, (1957)]

[7] Mather J.C. et al. 1994, ApJ, 420, 439

[8] Procopio P., 2009, PhD Thesis, http://annali.unife.it/IUSS/vol2/procopio.pdf

[9] Procopio P., 2005, Degree Thesis, Universitá degli Studi di Bologna

[10] Procopio P., Burigana C. 2005a, Int. Rep. IASF-BO 419/2005, July

[11] Procopio P., Burigana C. 2005b, Int. Rep. IASF-BO 420/2005, July

[12] Procopio P., Burigana C., 2009, A&A, 507, 1243