Optimization of Fortran Hydrocodes

G. BONO[1], R. SMAREGLIA[1], L. PULONE[2], A. BALESTRA[1], C. VUERLI[1], C. CUMANI[1], P. MARCUCCI[1], M. PUCILLO[1], P. SANTIN[1], F. PASIAN[1], L. RUSCONI[3], G. SEDMAK[3]

[1] *Osservatorio Astronomico di Trieste, Trieste, Italy;* [2] *Osservatorio Astronomico di Roma, Roma, Italy;* [3] *Dipartimento di Astronomia Univ. di Trieste, Trieste, Italy*

ABSTRACT. Le tecniche adottate per ottimizzare l'esecuzione scalare di codici idrodinamici monodimensionali su workstation UNIX vengono discusse. In particolare vengono analizzati i problemi concernenti l'uso di diversi compilatori Fortran e l'accuratezza numerica degli algoritmi utlizzati. Vengono infine forniti alcuni suggerimenti per garantire una maggiore portabilita' ed affidabilita' per codici che richiedono notevoli potenze di calcolo.

1. Introduction

Variable stars play a fundamental role in several relevant astrophysical problems concerning distance determination (standard candles), evaluation of the cosmological helium abundance (width of the instability strip), H-B morphology and the second parameter problem (Oosterhoff dichotomy). Futhermore, due to their spatial distribution and relatively high luminosity these objects can provide useful suggestions on the metallicity gradients and the density distribution of the Galaxy (Bono *et al.* 1993).

With the aim of supplying a homogeneous and extensive set of theoretical pulsation models that can be soundly compared with observative data, an investigation of both the amplitudes and the modal behaviour inside the instability strip of Pop. I and Pop. II variable stars has been planned. In accordance with the goals of this project a linear (**LFCN**) and a 1D non-linear (**HYCN**) hydrocodes have been developed (Stellingwerf 1982; Bono and Stellingwerf 1991,1993; Stellingwerf and Bono 1992) to evaluate the pulsation properties, the modal stability and the coupling pulsation/convection on a wide range of astrophysical parameters (mass, chemical composition, luminosity and effective temperature) together with physical inputs (opacity tables, time-dependent convective transport theories, shock propagation).

Extensive numerical tests have been performed on different computer systems (Apollo DN10040, HP 9000/710, Sun Sparc 10) in order to speed up the running time of these codes on desktop computers presently available at the Trieste Observatory. In particular, several *syntactic* changes have been carried out on the algorithms implemented in the codes to optimize their execution. The main differences involving both complex and double precision arithmetic and the relative efficiency among different FORTRAN compilers have been analyzed. Simple procedures to evaluate the round-off error and the numerical accuracy of the code have been presented in this paper. A list of general rules

and compiler options required to reduce the CPU time of high computational complexity FORTRAN codes are summarized and new developments are also discussed briefly in the following sections.

2. Layout of the Codes

The physical content and the numerical methods adopted to describe radial stellar pulsations may been found in several textbooks (Cox and Giuli 1968, Cox 1980) and outstanding classical papers (Baker and Kippenhahn 1962,1965; Castor 1971; Stellingwerf 1975). An envelope model encloses 90% of the stellar radius and a few per cents of the total mass. The core of the star (*i.e.* the region where nuclear reactions take place) does not generally affect the pulsation properties and the modal stability of the star, hence it is generally excluded. The envelope structure is assumed to be spherically symmetric, non-rotating and chemically homogeneous. The effects of magnetic fields and mass loss are neglected as well.

The calculation of a non-linear pulsation model may be split into two different parts. In the first part (linear hydrocode), the momentum and energy conservation equations are linearized and the time dependence for all perturbations is assumed to be exponential (*i.e.* of the form $e^{i\omega t}$ where ω is the complex eigenfrequency). Therefore the time derivative d/dt appearing in the quoted equations can be replaced by a $i\omega$ term. In this context, the inner and outer boundary conditions as well as the physical structure of the static envelope are evaluated. Futhermore, the periods, the growth rates and the eigenvectors (see **LFCN** flow diagram) for each mode (generally restricted to the fundamental and the first three overtones) may also be estimated.

The non-linear effects on light, temperature and velocity curves and the modal stability at full amplitude are investigated with a non-linear implicit Lagrangian hydrocode that solves the local conservation equations plus a non-local and time-dependent treatment of convection. In this framework, a turbulent pressure term and an eddy viscosity dissipation term have also been included to provide the proper treatment for turbulent motions both for length scales of the order of a mixing length or smaller (see **HYCN** flow diagram).

Both the radiative and convective transfers have been derived assuming diffusion approximation even in optically thin regions. However, this approximation results quite reasonable both for cluster variables (Bono and Stellingwerf 1992), and classical Cepheids (Bendt and Davis 1971). The physical assumptions, the calibration and model dependence on free parameters and the numerical methods adopted in these codes have been discussed in detail in the papers quoted previously.

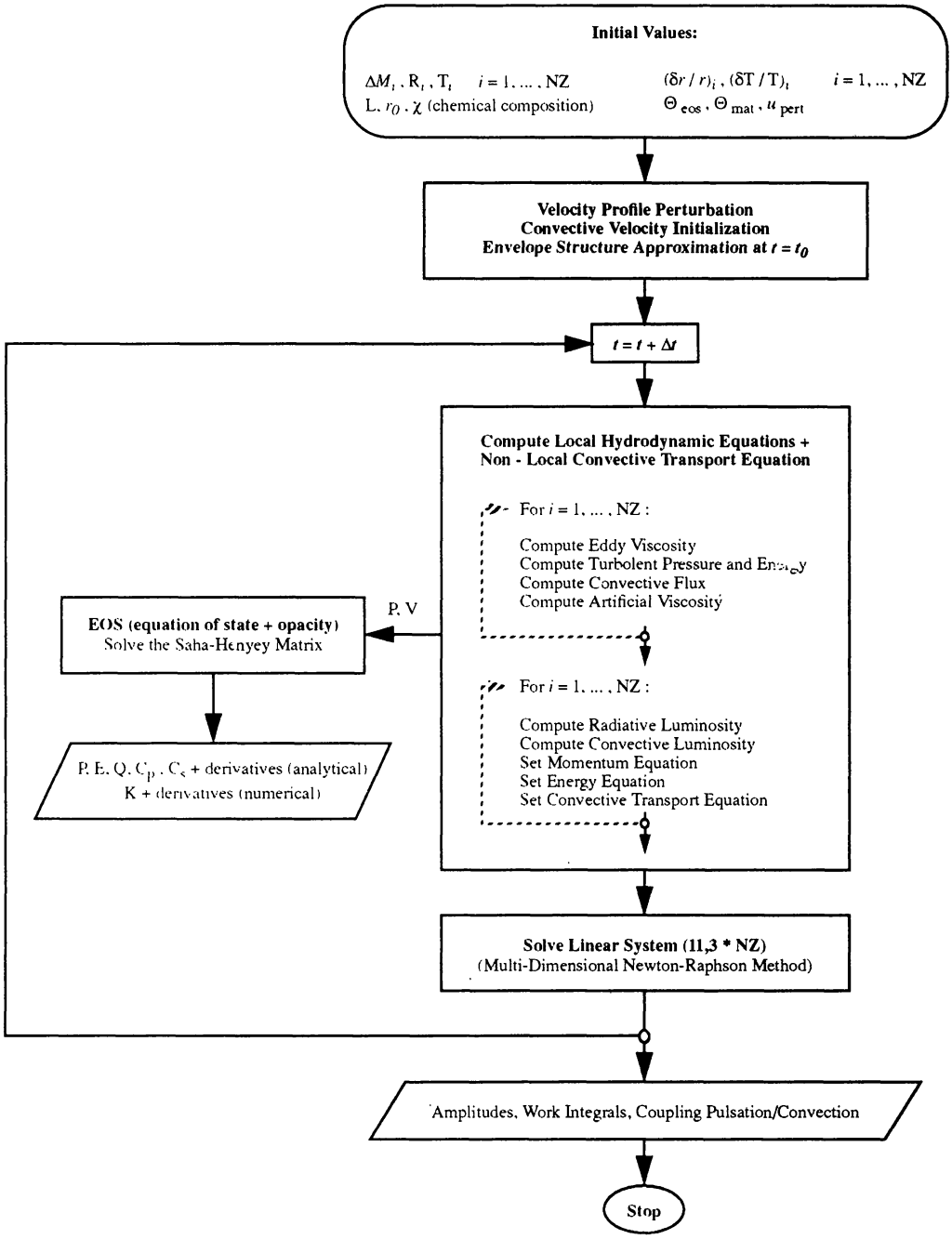
A rough estimate of the numerical complexity of a case may be established by adopting the following relation:

$$NC = N_{per} * N_{ts} * N_{it} * N_{op} \quad (1)$$

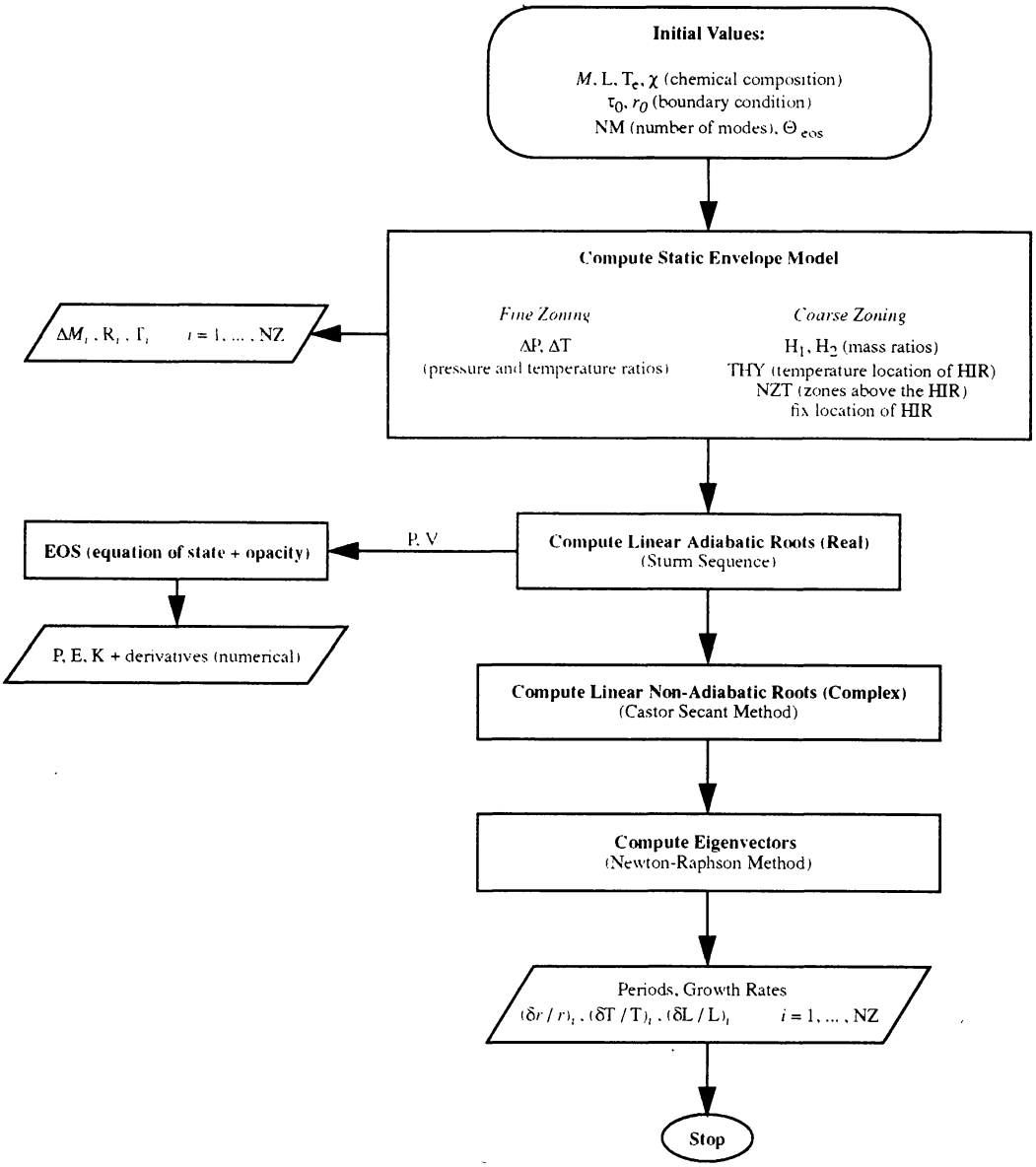
where N_{per} is the number of periods (500-10000), N_{ts} the number of time steps per period (300-500), N_{it} the number of iterations per time step (4-6) and N_{op} the number of floating point operations per time step that can be approximatively assumed to be proportional to the square of the zone number (*i.e.* $100^2 - 150^2$). The lower (NC=10

Gflops) and upper (NC=1000 Gflops) limits depend on the growth rates, the selected modes and the spatial resolution of the particular pulsation models. ($\approx$ 10 Mflops) are currently being used on workstations to execute these codes. Thus these numbers imply that a job may run for a time-span of half a day to a few days.

HYCN FLOW CHART



LFCN FLOW CHART



3. Optimization

Before any optimization can be performed on the codes described previously, the results of identical source codes on different computer systems need to be exactly the same within the adopted numerical accuracy or the computer precision. Both in the linear and non-linear hydrocode, double precision has been adopted for real and complex variables. However, although an implicit statement has been included at the beginning of the codes, the eigenvectors of a case obtained by executing **LFCN** on a HP and a Sun workstation could differ of a factor $10^{-4} - 10^{-5}$. The reason for this discrepancy in the output, is that by adopting an implicit double precision statement, the Sun Fortran compiler initializes all real and complex constants as `REAL*8` or `COMPLEX*16` data types. In order to obtain the same treatment on HP, the code has to be compiled with the "-R8" option.

When optimizing a code execution on a particular workstation, the advice and tricks listed in the compiler optimization rules of that particular computer should be followed. However, in spite of the substantial difference between various computer systems, a few general prescriptions may be provided (Bono 1987; Kruger 1990):

• **Subroutine inlining**: the compiler attempts to replace the call statement in the calling routine with the statements of the called subroutine to eliminate the call overhead of subroutines (or functions) that are frequently referenced. However, this procedure increases the size of the `.obj` file, hence the inlining should be restricted to small and "heavy" routines (*e.g.* interpolation functions, dot products, etc.). The compiler is unable to inline a subroutine that includes a statement that maintains "live-on-entry" variables such as `SAVE` or `DATA` within its body. This problem may be avoided using a global variable and a `PARAMETER` statement respectively. Futhermore, the subroutine should not include `ENTRY POINT` or `EQUIVALENCE` statements.

Although the manuals do not stress its relevance, the memory management can inhibit the optimization at different levels. Variable ordering in common blocks has to be: `REAL*8, REAL, INTEGER, LOGICALS` otherwise not only could the optimization fail but the debugging could result unreliable too. In this context, problems could arise also from a call to a subroutine by using an array (A(i,j)) with more arguments than the subroutine definition (A(i*j)).

• **Loop optimization**: one of the most powerful methods utilized to maximize run time efficiency is the pipelining or vectorization of loops. In order to ensure that the optimization proves effective, the loop should not include: branches, calls to external routines, recursion, `I/O` operations. Futhermore, repeated loops running on the same index should be combined, whereas short loops ($i < 10$) should be written into sequential statements. This trick increases the size of the code but eliminates the loop overhead and in case the short loop is nested, has the faculty of vectorizing the external loop. Indeed, several compilers (Apollo, HP9000) whilst optimizing nested loops, are only able to vectorize the innermost. It is worth noting that loop *unrolling* can be performed automatically by the compiler using the `-Wp,-unroll=`n option in the `f77` command line. The introduction of scalar temporaries in a loop allows a more efficient register utilization and reduces repeated unnecessary calls to external functions. Finally, references to arrays characterized by a variable stride (*i.e.* the index `j` of a generic array `A` computed inside a loop does not change linearly with the index loop `i`) should also be

avoided.

• **Non-reducible code**: the optimization of a block of statements fails if a non-reducible procedure has been encountered. These problems generally arise when a `GO TO` located inside a loop or an `IF THEN ELSE` move the control of the execution backward (i.e. to statements and/or procedures already processed).

After these simple rules have been applied and before further improvements can be made with regards the optimization, it is necessary to profile the code that evaluates the amount of `CPU` time used by each subroutine. The histogram obtained from the `UNIX` *gprof* programme indicates that the code spends 45% of the total time in the subroutine (HYD) which evaluates the set of equations and solves the matrix. The subroutine (EOS), where the equation of state and the opacity are evaluated, uses 50% of the total time, whereas the remaining 5% is used by the output subroutine.

The optimization of the HYD subroutine represents a difficult task, for an implicit method has been adopted within the code to discretize the set of equations *ergo* the unknowns X_i^{n+1} at time $t+\Delta t$ are expressed in terms of known quantities $(X_{i-1}^n, X_i^n, X_{i+1}^n)$ at time t and in terms of unknowns $(X_{i-1}^{n+1}, X_{i+1}^{n+1})$ at time $t + \Delta t$. Therefore all the equations need to be solved simultaneously and the recursion on the zone number `i` represents a tantalizing but difficult problem to handle.

In view of the considerations made above, this paper will concentrate on the optimization on EOS and on the subroutines that perform the decomposition and the inversion of the Saha-Henyey matrix. The reduction of the call overhead to EOS was brought about with the shifting of the calling statement outside the first loop on the zone number (see **HYCN** flow diagram). The input quantities of this subroutine (pressure and volume) are known for the entire envelope structure, since the loop on the zone number can be easily included in the body of the subroutine. The only disadvantage, at this stage, is represented by the increment in the static memory used by the code. Indeed, all the thermodynamic quantities and the related derivatives become array variables from scalar ones, but fortunately, at present, this *side-effect* has not proved to be a real problem since the size of the executable file is smaller than one Mbyte. The functions performing the matrix and vector dot products have been inlined in the subroutine body as well.

In Table I the running times of a partially optimized (old) and the fully optimized (new) version of **HYCN** are listed. The values showed refer to the computation of the first two periods of a RR Lyrae model centrally located in the instability strip. The two versions have been executed on three different workstations and several numerical tests have been performed to evaluate the relative efficiency by adopting different compiler options. However, the results reported below can be regarded as significant only in the part relating to the code that is being developed. The first set of values was obtained by compiling the codes with the default `f77` command line, the second set by adopting the default optimization option, whereas the third set was obtained by using the highest optimization level.

It is worth noting that by increasing the optimization level, the compilers spent more time compiling and generally increased the size of the object code. However, the run time efficiency might not be strictly increasing. Indeed, optimization levels that result

higher than the default value do not produce faster codes. The Apollo run times are hardly surprising as the optimizations performed on the code have been done bearing the hardware and software features of new RISC workstations in mind.

The last set of values refer to a numerical test compiled by invoking a FORTRAN optimizing preprocessor before using the standard optimizing compiler. This preprocessor improves the recognition and optimization of loops (vectorization, unrolling) and performs a better routine inlining.

Table I. Running times of the old and new **HYCN** version executed on different computer systems whilst adopting different compiler options

Sun Sparc 10 SunOS 4.1.3			Apollo DN10040 DomainOS 10.3			HP 9000/710 HP-UX 8.07		
new	old	ratio	new	old	ratio	new	old	ratio
6'.59"	8'.10"	.85	8'.58"	8'.34"	1.05	6'.18"	8'.17"	.76
	-O3			-O3			+O2	
4'.42"	5'.11"	.91	4'.55"	4'.54"	1.00	2'.57"	3'.11"	.89
	-O4			-O4			+O3	
4'.57"	6'.34"	.75	5'.05"	5'.01"	1.01	2'.57"	3'.05"	.96
-fast -O4 -libmil			-Wp,opt -O			+OP3		
2'.30"	3'.30"	.71	4'.58"	4'.55"	1.01	2'.35"	3'.31"	.73

However, the FORTRAN preprocessor performs the optimization on the source code, and consequently the arithmetic reduction of the optimized code differs from the scalar one and hence the roundoff error is accumulated differently. This effect was checked on quantities (thermal, gravitational and turbulent energies) that are particularly sensitive to the roundoff error propagation and an insignificant difference ($< 10^{-12}$) was found in the final results between the optimized and scalar code.

4. Conclusions

The methodology adopted to increase the running time efficiency of 1D Lagrangian hydrocodes has been presented in this report. The changes and *syntactic* rules performed to optimize the execution of these codes on different computer systems provided a reduction of the order of 25% with respect to the partially optimized version.

In the light of the considerations discussed in the above paragraphs, concerning the significant reduction of the elapsed time required for the running of these codes, two diverse routes may be undertaken.

The first route involves the development of new methods to treat the local conservation equations and the convective transport equation characterized by a low computational complexity. The Smooth Particle Hydrodynamics represents a tempting direction to embark on but unfortunately the exact energy conservation is to date an open problem. Improvements on the algorithms adopted to evaluate the equation of state and opacity derivatives or on the algorithm adopted for the matrix inversion (Canfield, McClymont and Puetter 1984, Kronsjo 1985) could also provide significant reductions of CPU time.

The second route implies the use of commercial software packages like P4, LINDA and EXPRESS that are able to simulate a parallel distributed system connecting a heterogeneous network of workstations (Pucillo 1993). In this framework, the execution of a routine or a block of routines can be split on different CPU, thereby virtually increasing the running time efficiency with the number of workstations physically connected.

The underlining philosophy as far as the job executions are concerned is *divide et impera*. However, preliminary tests performed using three workstations portray a consistent dependence on the network traffic. The subject has to be analyzed in detail and the procedures adopted to exchange data among different processes, in particular, need more insights, before any reliable conclusions may be furnished.

Acknowledgements

This work has been partially supported by the National Group of Astronomy (GNA-CNR) and the Consorzio per la Magnetofluidodinamica dell'Università di Trieste.

References

Baker, N.H., Kippenhahn, R.: 1965, *Astrophys. J.* **142**, 868
Bendt, J.E., Davis, C.G.: 1971, *Astrophys. J.* **169**, 333
Bono, G.: 1987, Degree Thesis, Univ. of Bologna
Bono, G., Stellingwerf, R.F.: 1991, in *New results on standard candles* , ed. F. Caputo, Mem. Soc. Astron. It. **63**, 357.
Bono, G., Caputo, F., Stellingwerf, R.F.: 1993, *Astrophys. J.*, submitted
Bono, G., Stellingwerf, R.F.: 1993, *Astrophys. J.*, submitted
Canfield, R.C., McClymont, A.N., Puetter, R.C.: 1984, in *Radiative Transfer*, ed. by Kalkofen W. (Cambridge Univ. Press)
Castor, J.I.: 1971, *Astrophys. J.* **166**, 109
Cox, J.P., Giuli, R.T.: 1968, *Stellar Structure*, (New York: Gordon and Breach), (27).
Cox, J.P.: 1980, *Theory of Stellar Pulsation*, (Princeton: Princ. Univ. Press.)
Kruger, A.: 1990. *Efficient* FORTRAN *Programming*, (J. Wiley & Sons Inc.).
Kronsjo, L.: 1985. *Computational Complexity of Sequential and Parallel Algorithms*, (J. Wiley & Sons Inc.).
Pucillo, M.:1993, *in this proceedings*
Pucillo, M., Pasian, F., Bono, G., Mazzali, P., Smareglia, R.: 1993, in *Astronomical Data Analysis Software and Systems*, Victoria, B.C.
Stellingwerf, R.F.: 1975, *Astrophys. J.* **195**, 441
Stellingwerf, R.F.: 1982, *Astrophys. J.* **262**, 330
Stellingwerf, R.F., Bono, G.: 1992, in *New Perspectives on Stellar Pulsation and Pulsating Variable Stars*, Proc. IAU *Colloq. No.* 139, Victoria, B.C., ed. by J.M. Nemec, J. Matthews, (Cambridge Univ. Press), in press.