Mem. S.A.It. Suppl. Vol. 8, 64

Memorie della
Supplementi

Two platform independent versions of ATLAS12

K. M. Bischof

Institut für Astronomie, Universität Wien, Türkenschanzstraße 17, 1180 Vienna, Austria
e-mail: bischof@astro.univie.ac.at

Abstract. ATLAS12 makes use of some non-standard features of the VAX/VMS compiler, which is highly appreciable if one uses a VMS workstation, but causes problems of portability that make it difficult to compile ATLAS12 on various non-VMS operating systems. This article describes the modifications to ATLAS12 that became necessary in order to get this code running with the open source GNU compiler (g77), which is easily available on most Linux/Unix based systems (including Mac OS X). We also present our parallel and modularised Ada95 version of ATLAS12 and give an overview of CAMAS, our new magnetic stellar atmosphere code.

Key words. Methods: numerical – Stars: atmospheres – Stars: magnetic fields

1. Introduction

This contribution describes the modifications that were necessary in order to convert ATLAS12 into a essentially platform independent code. We will not discuss the usage of ATLAS12 and the underlying physics, since they are well described in the publications of Kurucz (1970), Castelli (1988) and a number of articles in this volume. Here we focus on our changes to the original version of the ATLAS12 code. Finally salient features of our ATLAS related magnetic stellar atmosphere code CAMAS are shortly discussed.

1.1. *ATLAS12: advantages & disadvantages*

ATLAS12 was chosen as a non-magnetic comparison to our new magnetic atmosphere code CAMAS for several reasons: most importantly, it is an opacity sampling code, which is indispensable in order to allow for peculiar abundances in individual stars. Furthermore ATLAS12 is the de facto standard stellar atmosphere program and contains well tested opacity routines. On account of this ATLAS12 is considered a good starting point and comparison for our opacities and models.

In order to adapt the ATLAS12 code to fit our needs some known problems had to be overcome: (a) The equation of state incorporated in ATLAS does not allow for depth dependent abundances. (b) The depth and frequency resolutions are limited by the fixed array lengths and the pretabulated matrix used by the radiation transport subroutine (JOSH). (c) ATLAS12 reads the line data from binary files. The data files distributed with ATLAS are big endian[1] files, which can cause problems, since most of the PCs using Intel or AMD proces-

Send offprint requests to: K. M. Bischof

[1] The endianness describes where the most significant bit is located. Big end first is called big endian, e.g. the values of the 4-byte big endian integers in

sors are little endian, and the big endian switch in the file I/O command is not working in the current version of the g77 compiler. (File I/O will be discussed in detail in the next section).

Evidently, in the absence of a VMS FORTRAN compiler, the platform dependence was the problem that had to be solved first. In a further step, ATLAS12 was successfully ported to Ada95, an object oriented parallel language.

2. FORTRAN77 version

It is becoming more and more common to replace old VMS workstations with systems running under Linux or Mac OS. Faced with the same portability problems as e.g. Sbordone (2005) we chose to abandon VAX/VMS-specific FORTRAN commands and to establish a FORTRAN77 version that can be compiled using the free GNU FORTRAN compiler (g77), since this is the most easily available compiler for Linux/Unix based systems (including Mac OS X). Therefore we consider this version useful for astronomers having no access to (proprietary) VAX/VMS compatible FORTRAN compilers.

Our platform independent g77 version of ATLAS12 can be downloaded from our homepage:
http://fedelma.astro.univie.ac.at/web/bischof/

2.1. Distinctions between VAX/VMS- and g77-FORTRAN

Due to the absence of the additional functionality provided by the VAX/VMS compiler these non-standard features have to be emulated by adding the corresponding commands directly into the code. Most changes concern the definition and the handling of the variables. Some additional portability problems were caused by the use of functions provided by the underlying operating system (e.g.,“ABORT”).

The most important modifications that were necessary in order to port the code to g77 FORTRAN are:

- **floating point and variable precision**
  [0. → 0.d0]
  If this precision is not explicitly defined, the g77 compiler may use any precision and the code may yield different results from the VAX/VMS FORTRAN version. Thus the precision of all floating point literals has been defined.
- **declaration and initialisation of all variables**
  *[replace “IMPLICIT REAL*8” statement]*
  The “IMPLICIT REAL*8” statement causes a compilation problem. It was therefore removed. We also decided to explicitly initialise the variables that need to be set to zero, because the defaults for the “-finit-local-zero” switch of the g77 compiler may be different on different machines. The declaration and initialisation of all variables also made the modularisation of the code and consequently the porting to Ada95 feasible.
- **“dummy arrays” and data types**
  *[var(1) → var(*), Real*8 → Double Precision, . . .]*
  The VAX/VMS FORTRAN syntax for the declaration of “dummy arrays” causes compiler errors or warnings and was therefore replaced. The data types Real*4 and Real*8 are deprecated - consequently both data types were replaced with “Double Precision” for variables that do not appear in common-blocks.
- **addressing two-dimensional arrays**
  [NNN(1) → NNN(1,1)]
  A two-dimensional array cannot be addressed like a one-dimensional array in standard FORTRAN77. In contrast to g77 FORTRAN, where one needs two indices to address a two-dimensional array, it is possible in VAX/VMS FORTRAN to address such arrays using only one index. Since ATLAS12 makes use of this shorthand, these array-references had to be changed.
- **values of variables in subroutines**
  [“SAVE var”]
  There are two ways of memory allocation: static allocation, which is the VAX/VMS default and allows subprograms to retain

---

the ATLAS12 data files are computed by “1^{st} byte * 2^{24} + 2^{nd} byte * 2^{16} + 3^{rd} byte * 2^{8} + 4^{th} byte”

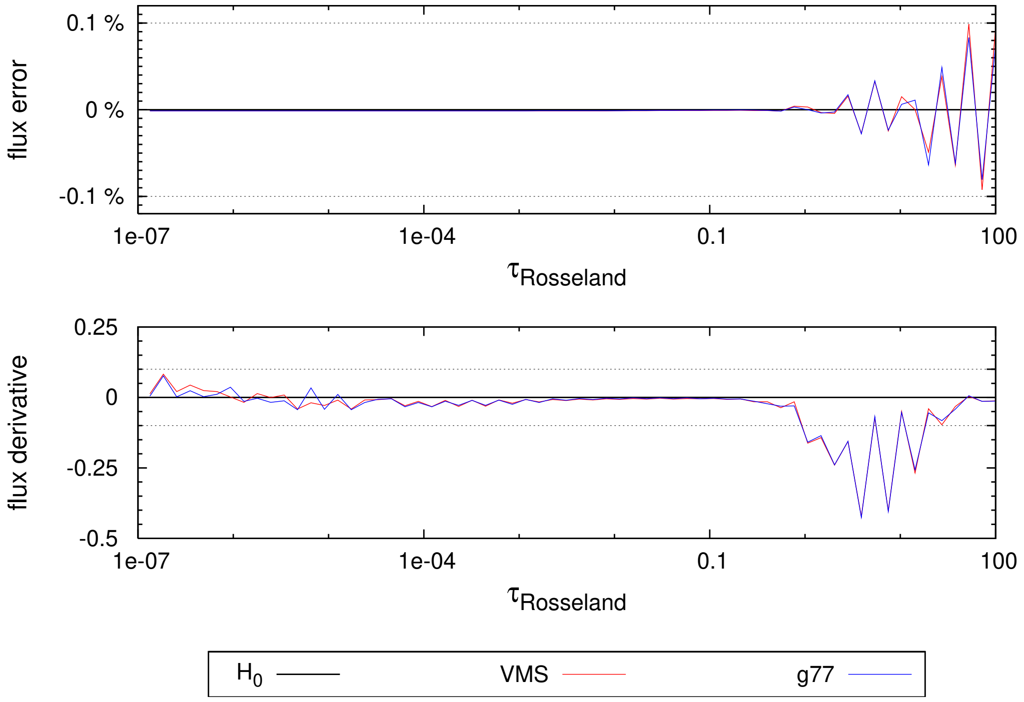

Fig. 1. The results of 25 iterations on a 10 000 K main sequence model with solar abundances. This plot shows the small differences in the flux error (upper panel) and in the flux derivatives (lower panel) between the model computed with the VMS compiler and with the Linux g77 compiler after 25 iterations. The dashed lines indicate the respective convergence limits. It can be seen that the differences do not preclude convergence; being very small they can be interpreted as due to different floating point representations on the two machines.

variable values, and stack allocation, which is the g77 default and does not retain values of variables defined in subprograms unless they are declared using the "SAVE" command. Since ATLAS12 was originally written for a VMS workstation it assumes that subroutines are able to retain the values of all their "private" variables until the next call. When translating the code to the g77 FORTRAN version we checked which variables are reused after the function-call and retained only the values of these variables because saving all variables was considered a waste of memory.

– **different I/O-format and file-format**
[reorder bytes after reading]
The binary files distributed by R. L. Kurucz contain big endian encoded integers. The read option that allows the selection of the endianness is not recognised by the g77 compiler, and so these files are now read byte by byte on little endian machines, after which a byte-reordering routine is applied. Tests of the read routines showed that out of approx. 9 million selected lines from highlines.dat and lowlines.dat only 3 lines were selected differently by the Linux and VAX/VMS codes when computing a 10 000 K main sequence star model. This negligible discrepancy most probably results from the different floating point representations of the machines.

– **replace system dependent calls**
[ABORT → EXIT(0)]
Some statements in ATLAS12, mainly the routines that help the user to compute the runtime, and the exit statements, contain

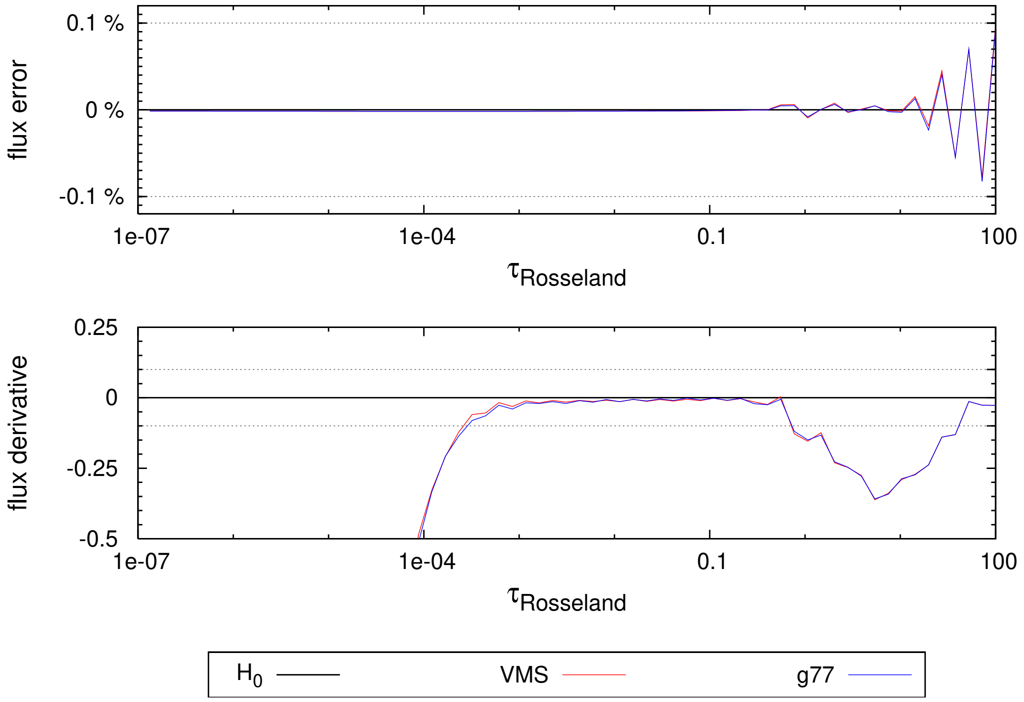

Fig. 2. The results of 25 iterations on a 10 000 K main sequence model with peculiar abundances (C, N, O: −1 dex; Cr, Fe, Ni: +1 dex). This plot shows that the differences in the flux error (upper panel) and in the flux derivatives (lower panel) also remain small if the abundances are changed in the ATLAS9 starting model. (In Fig. 1 no changes to the starting model were made.)

VAX/VMS system calls. These had to be replaced by their Unix/Linux equivalents.

– **GOTO cannot jump into if-block with unmet condition**
  [PFGROUND]
  In the subroutine PFGROUND it may occur that a GOTO statement leads into an if-block with unmet condition. The VAX/VMS compiler allows this action whereas the g77 compiler refuses to compile this construct. Therefore minor changes to the program structure proved necessary.

2.2. Examples (VAX/VMS compared to g77)

In order to test whether the porting was successful a comparison was made between the VAX/VMS code running on a DEC VMS workstation and the g77 FORTRAN code running on a Debian Linux PC.

The resulting models computed on these different platforms were compared. In Fig. 1 and Fig. 2 the results after 25 iterations on a converged ATLAS9 10 000 K main sequence model with (a) solar abundances and (b) peculiar abundances (C, N, O: −1 dex; Cr, Fe, Ni: +1 dex) are shown. There are no visible differences between any of the output models. The only variables where small differences can be seen are the flux error and the flux derivative. These differences are of the order of a few parts per thousand, which is about the order of magnitude one would expect from different floating-point representations and different numerical precision. Since the convergence criteria (marked with dashed lines in the figures) are one part per thousand in flux error and 0.1 in flux derivative, it is obvious that these differ-

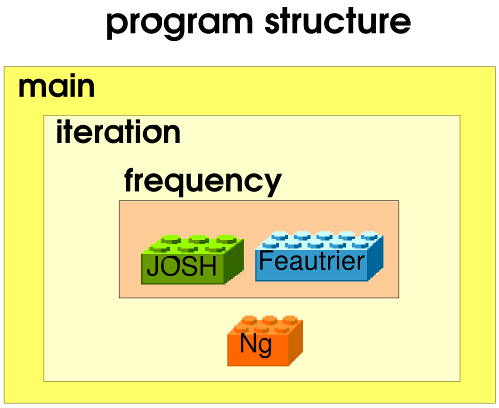

Fig. 3. Structure of our version of ATLAS12. The location of the newly included / replaced packages with standard interfaces mentioned in section 3 is indicated. JOSH and Feautrier are radiation transport routines, Ng is an iteration acceleration routine.

ences may safely be ignored. Fig. 1 as well as Fig. 2 display these negligible differences.

3. Parallel Ada95 version

For reasons explained in detail in e.g. Stift (1998a,b) and Stift & Dubois (1998) our existing line synthesis code COSSAM (Stift 2000; Wade et al. 2001) and radiative diffusion code CARAT (Alecian & Stift 2002) are written in Ada95. In order to include the Kurucz opacity routines into our codes, the whole of ATLAS12 was ported to this programming language.

Ada95 is an object oriented language which encourages modularisation and facilitates the parallelisation of the code. As all modern computing languages, Ada95 allows one to determine the needed array-sizes at runtime and is therefore able to minimise the memory requirements and to handle any number of frequencies (up to the limits imposed by the hardware used).

The *modularisation* of our Ada95 version of ATLAS12 is a big advantage since establishing interfaces for all subroutines ("encapsulation") simplifies the exchange and insertion of subroutines into existing codes. This new program architecture provides *reusable packages* (e.g. the continuum package) and at the same time it becomes fairly easy to test the effects that occur if one replaces the subroutines used in ATLAS12 with the ones used in our codes. In the present case an example for these *replaceable parts* is the solver for the radiation transport (JOSH) which was replaced by a Feautrier solver. An example for the *insertion of packages* is the Ng acceleration.

3.1. Feautrier solver

If one intends to calculate models with different numbers of depth points, it can be considered a serious drawback of the JOSH routine that the grid resolution cannot be specified at runtime. This radiation transport solver interpolates the values from the optical depth scale of the model onto a fixed set of depth points where the integration matrices used in this method (see Kurucz 1970) are pretabulated. Therefore we replaced this routine with the non-magnetic Feautrier solver (Feautrier 1964). This solver allows for a change of the grid-resolution without recompilation, whereas the JOSH routine would need new hard-coded integration matrices.

Due to the different numerical errors and different grid resolutions these solvers yield slightly discrepant results.

3.2. Ng acceleration

The implementation of the iteration scheme proposed by Ng (1974) accelerates the convergence of our models by 10 to 50 %.

This scheme uses the results of four (or more) iterations of a function f ($\ldots, f_{n-3}, f_{n-2}, f_{n-1}, f_n$) to construct a new function $\bar{f}$ that is closer to the converged function (see Fig. 5). The iteration process can be written as:

$$A f_{n-1} = f_n \tag{1}$$

where $f(x)$ represents a real function and A a (non)linear operator. If A can be assumed to be linear (or at least a linear approximation to A exists) it is possible to calculate the optimum values of the weighting constants $c_1, c_2, \ldots$. The application of the operator A to the functions on the right hand side of Eq. 2 can be

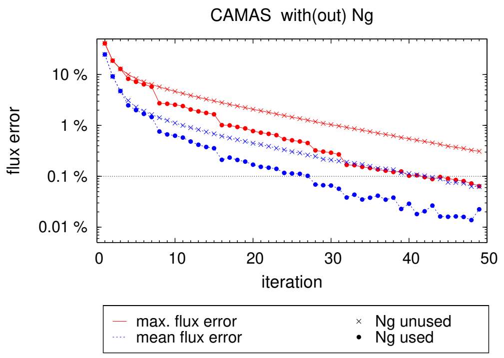

Fig. 4. Application of the Ng-acceleration. The behaviour of the flux errors during the iteration process is shown for a model with Ng-acceleration and a model, were this acceleration was not used. The sudden declines of the errors are produced by the application of the Ng-acceleration.

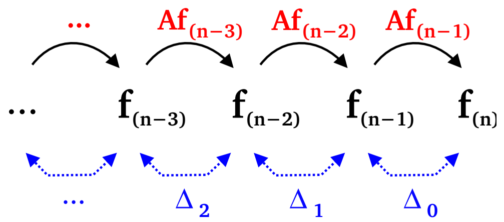

Fig. 5. The Ng acceleration scheme. This scheme uses the results of four (or more) iterations ($\ldots$, f_{n-3}, f_{n-2}, f_{n-1}, f_n) to construct a new function $\bar{f}$ that is closer to the converged function. A is a linear operator that represents or at least approximates the iteration. The differences between two consecutive iteration results f_{n-1} and f_n are denoted by Δ_0.

computed using Eq. 1. This leads to Eq. 3:

$$\begin{aligned} \bar{f} &= (1-c_1-c_2-\ldots)f_{n-1} \\ &+ c_1 f_{n-2} + c_2 f_{n-3} + \ldots \qquad (2) \\ A\bar{f} &= (1-c_1-c_2-\ldots)f_n \\ &+ c_1 f_{n-1} + c_2 f_{n-2} + \ldots \qquad (3) \end{aligned}$$

The converged function satisfies $Af = f$. The difference between the not fully converged function $\bar{f}$ and the expected result after application of the linear operator $A\bar{f}$ can be expressed by:

$$\begin{aligned} A\bar{f} - \bar{f} &= (1-c_1-c_2-\ldots)\Delta_0 + c_1\Delta_1 + c_2\Delta_2 \\ &+ \ldots \\ &= \Delta_0 + c_1(\Delta_1-\Delta_0) + c_2(\Delta_2-\Delta_0) \\ &+ \ldots \qquad (4) \end{aligned}$$

with $\Delta_0 := f_n - f_{n-1}$. The best weights $c_1, c_2, \ldots$ can be found by using the condition that the effect of the operator A on the new function $\bar{f}$ has to be minimised. This can be written as $\|A\bar{f} - \bar{f}\|^2 \rightarrow \min.$

Eq. 4 yields the following system of equations if the derivatives with respect to the weights $c_1, c_2, \ldots$ are calculated:

$$\begin{aligned} \int \Delta_0(\Delta_1-\Delta_0)\mathrm{d}x &= c_1 \int (\Delta_1-\Delta_0)(\Delta_1-\Delta_0)\mathrm{d}x \\ &+ c_2 \int (\Delta_1-\Delta_0)(\Delta_2-\Delta_0)\mathrm{d}x \\ &+ \ldots \\ \int \Delta_0(\Delta_2-\Delta_0)\mathrm{d}x &= c_1 \int (\Delta_1-\Delta_0)(\Delta_2-\Delta_0)\mathrm{d}x \\ &+ c_2 \int (\Delta_2-\Delta_0)(\Delta_2-\Delta_0)\mathrm{d}x \\ &+ \ldots \\ \ldots &= \ldots \end{aligned}$$

The integrals arise from the definition of the inner product. This system is solved in the standard way (since all differences Δ_i are known). Using the calculated weights $c_1, c_2, \ldots$ the new input function for the next iteration step is calculated by:

$$\begin{aligned} f_{n+1} &= (1-c_1-c_2-\ldots)f_n \\ &+ c_1 f_{n-1} + c_2 f_{n-2} + \ldots \end{aligned}$$

Test calculations show that the application of the Ng scheme can save up to 50% of the computation time. In Fig. 4 one can easily see that after the usage of the Ng routine there is a dramatic decline in the flux errors. If the deviations of the computed function from the converged function are small, the influence of the nonlinearity of A gets stronger and the application of the Ng routine is no longer able to speed up the process.

3.3. Parallelisation

Porting the code to Ada95 has another advantage: with a modularised Ada95 code it is easy to create a thread-parallel version. Since

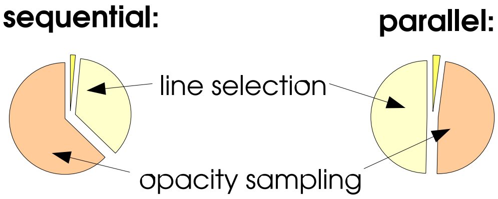

Fig. 6. The effect of the parallelisation on the runtime distribution. The diagram on the left shows that a sequential version of ATLAS12 spends most of the CPU time on opacity sampling in the frequency loop. Parallelisation of this loop saves a considerable amount of wall clock time. Due to the small synchronisation overhead the decrease of the runtime depends linearly on the number of processors. The diagram on the right shows that using a second CPU saves half the wall clock time spent on opacity sampling.

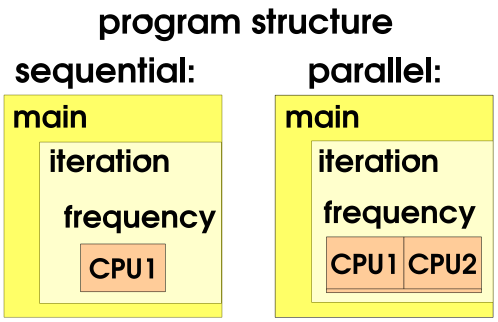

Fig. 7. Program structure of the parallelised version of ATLAS12. The opacity calculations (and opacity integrations) are distributed over the available CPUs. The summation of the subtotals from the different CPUs is indicated by the small box under the boxes labelled “CPU1” and “CPU2”.

ATLAS12 spends most of the computation time on opacity sampling (see Fig. 6), this part of the program was parallelised. As the calculations of the opacities at different wavelengths do not depend on each other, there is very little synchronisation overhead and the computation time needed for the integration of the opacities over all frequencies decreases linearly with the increase of the number of processors used by the program. Figure 7 shows the structure of our version ATLAS12 version in Ada95. The difference between the parallel version and the sequential version lies in the distribution of the opacity calculations over the available CPUs and the addition of the subtotals from the different CPUs (indicated by the small box below the boxes labelled “CPU1” and “CPU2” in Fig. 7).

In a further step the line-selection has also been parallelised. Obviously this results in a runtime distribution very similar to the runtime distribution in the sequential case. The saving of total computation time behaves analogously to the decrease of integration time described in the last paragraph.

3.4. Examples (VAX/VMS compared to Ada95)

For comparison 25 iterations were carried out on an ATLAS9 10 000 K main sequence model with (a) solar abundances and (b) peculiar abundances (C, N, O: −1 dex; Cr, Fe, Ni: +1 dex). In Fig. 8 as well as in Fig. 9 it can be seen that the differences to the original version become larger, which is most probably due to the different available data types, but are still unimportant. The higher numeric precision of the data types declared in the Ada95 version suggests that the results of the Ada95 code are less subject to rounding errors.

4. CAMAS - a “magnetic” atmosphere code

Working on magnetic CP stars one has to compute the influence of the magnetic field on the structure of the atmospheres. Therefore we have established a new magnetic stellar atmosphere code called CAMAS (**C**odice per le **A**tmosfere **Ma**gnetiche **S**tellari).

This can be regarded as an ATLAS related code, since it presently makes use of the ATLAS12 continuous opacities. The line lists for CAMAS were extracted from VALD (Kupka et al. 2000) and the line opacity routines were taken from the COSSAM code. The chemical equilibrium calculations in CAMAS are different from ATLAS. CAMAS uses an equation of state with hydrogen chemistry (Mihalas 1967). For the temperature correction procedure a variant of the Lucy Unsöld method (Dreizler 2003) is employed.

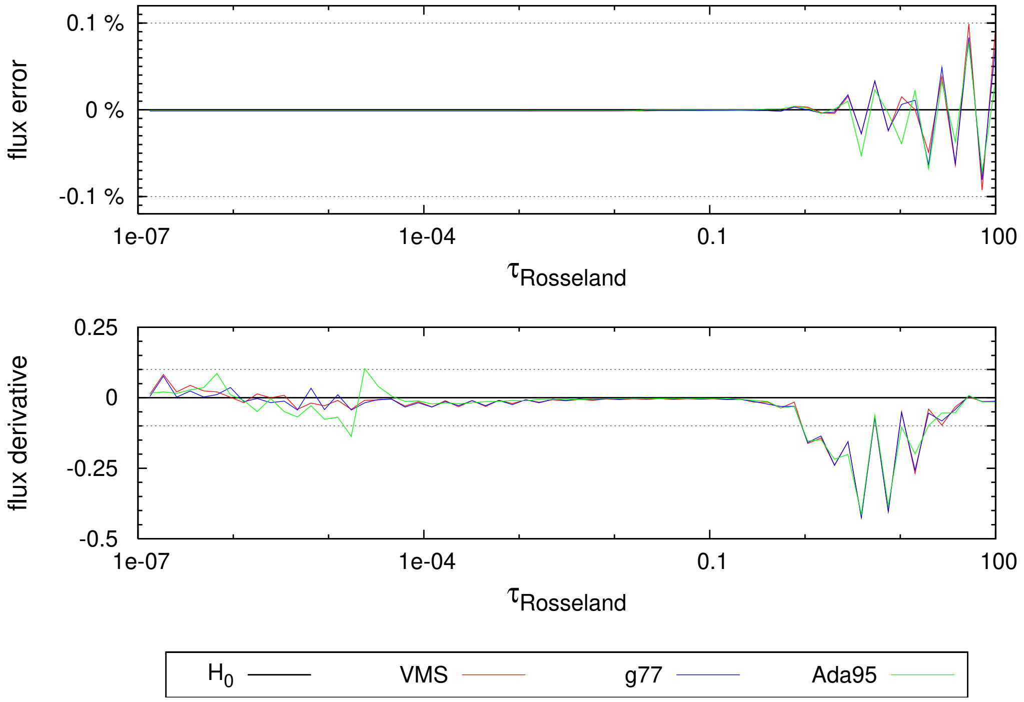

Fig. 8. Comparison of the Ada95 model to the FORTRAN models: results of 25 iterations on a 10 000 K main sequence model with solar abundances. (Analogous to Fig. 1.) The differences between the Ada95 model and the VMS model are larger than the differences between the FORTRAN models. This is probably due to different data types of higher numeric precision declared in Ada95 and the different mathematical libraries.

So far the influence of the magnetic field on the atmospheric structure is not fully taken into account. Magnetic line blanketing is included but Lorentz forces are neglected. The Zeeman Feautrier solver for the polarised radiation transfer equation has been taken from the CARAT code (Alecian & Stift 2002, 2004); the line opacity calculations are based on **Co**mponent by **Co**mponent **S**ampling. CAMAS is producing converged models but it still has to be improved by adding some missing physics (e.g. magnetic pressure) and to be tested thoroughly.

5. Conclusion

Our g77 FORTRAN code produces the same models as the original VAX/VMS FORTRAN code. Therefore it can safely be used for calculating ATLAS12 models on all kinds of Linux/Unix based systems (including Mac OS X). The Ada95 version also offers some additional functionality. It is platform independent, split up in modules and runs in parallel on multi-processor machines. Any limitations as to the maximum number of depth and frequency grid-points and the number of lines that can be treated have been pushed far beyond what is possible with the original version. The incorporation of the continuous opacity routines of ATLAS12 in our new CAMAS code for magnetic atmospheres makes it possible to compare our future models with those of R. L. Kurucz.

Acknowledgements. Fruitful discussions with M. J. Stift and R. L. Kurucz are gratefully acknowledged. The VAX/VMS models were computed with the help of E. A. Dorfi and his working group. This work has been supported by the *Austrian Science*

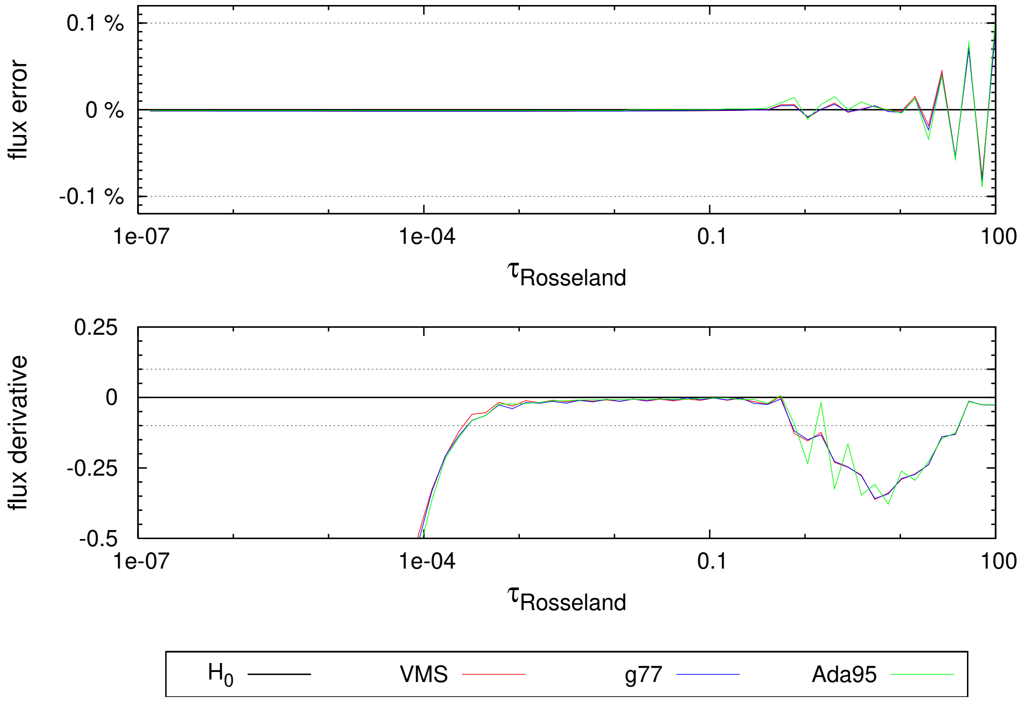

Fig. 9. Comparison of the Ada95 model to the FORTRAN models: results of 25 iterations on a 10 000 K main sequence model with peculiar abundances (C, N, O: −1 dex; Cr, Fe, Ni: +1 dex). One can see that the differences between the results of the codes do not get larger than one would expect due to the different data types and mathematical libraries.

Fund (FWF), project P16003 "Radiative diffusion in magnetic stellar atmospheres".

These programs can be downloaded from: http://fedelma.astro.univie.ac.at/web/bischof/

References

Alecian, G., & Stift, M. J. 2002, A&A, 387, 271

Alecian, G., & Stift, M. J. 2004, A&A, 416, 703

Castelli, F. 1988, "Kurucz's models, Kurucz's fluxes and the ATLAS code", O.A.T. Pub. No. 1164 http://wwwuser.oat.ts.astro.it/atmos/Castelli0607.pdf

Dreizler, S. 2003, ASP Conf. Ser. 288: Stellar Atmosphere Modeling, 288, 69

Feautrier, P. 1964, SAO Special Report, 167, 80

Kupka, F. G., Ryabchikova, T. A., Piskunov, N. E., Stempels, H. C., & Weiss, W. W. 2000, Baltic Astronomy, 9, 590

Kurucz, R. L. 1970, SAO Special Report, 309,

Mihalas, D. 1967, Methods in Computational Physics, 7

Ng, K.-C. 1974, J. Chem. Phys., 61, 2680

Sbordone, L., MSAIS, 8, 61

Stift, M. J. 1998, Contributions of the Astronomical Observatory Skalnate Pleso, 27, 390

Stift, M. J. 1998, Lecture Notes in Computer Science, 1411, 128 http://www.springerlink.com

Stift, M. J., & Dubois, P. F. 1998, Computers in Physics, 12, 150

Stift, M. J. 2000, A peculiar Newsletter, 33 http://www.astro.uwo.ca/apn/Archive/

Wade, G. A., Bagnulo, S., Kochukhov, O., Landstreet, J. D., Piskunov, N., & Stift, M. J. 2001, A&A, 374, 265