FORSCHUNGSZENTRUM JÜLICH GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61-6402

Interner Bericht

First Experiences with Fortran 90

Gerd Groten, Heinz W. Homrighausen

KFA-ZAM-IB-9421

September 1994
(Stand 22.09.94)

Proceedings of SHARE Europe Spring Conference, Brussels, April 21, 1994

First Experiences with Fortran 90

Gerd Groten and Heinz W. Homrighausen

Research Centre Jülich (KFA)
Central Institute for Applied Mathematics
D 52425 Jülich

This paper will give an overview on the activities undertaken to introduce the new Fortran 90 compilers into a Fortran-oriented research center. Especially, it will compare the compilers currently available (NAG, IBM AIX, Cray) concerning suitability for program development, portability of user code, and program performance.

As typical for the **Fortran history** in a scientific research center, programming activities at KFA Jülich are oriented to Fortran. Nearly 90 % of program development is done in Fortran; this percentage will even be higher with respect to the executed code. In many technical areas (reactor development, solid state physics, nuclear physics, plasma physics, computational chemistry) millions of lines of code have been developed which form a company-wide knowledge base and which force us to stay with the Fortran programming language history, rather than utilizing other modern programming languages, as C, or to use other program development tools not compatible with Fortran for the majority of numerical calculations.

Thus Fortran 90 will be the base for our numerically intensive applications in the future.

Our historical **computing environment** is described by nearly 30 years of IBM mainframes, now with the operating systems VM/CMS and MVS, and 10 years of high performance vector processing on Cray computers. The importance of high performance processing has grown by the fact that KFA offers computer power to a German supercomputer center (Höchstleistungs-Rechenzentrum, HLRZ) founded by 3 governmentally owned scientific research centers (DESY Hamburg, GMD Birlinghoven, KFA Jülich).

As usual for nearly all comparable institutions, KFA is now going to realize a decentralized computing concept. This concept will provide the scientists with groups of workstations, isolated workstations and PCs according to the interests and budgets of user groups and single users. All these computers are integrated into a local area network which also gives access to the Crays for high performance computing. It is expected that the majority of the tasks done by the mainframe till now will be overtaken by the workstations, so we decided to migrate all mainframe applications to the workstations within the next 3 years and then to abolish the mainframe. For those applications which should be continued to be executed on central computers because of different reasons, an 8 processor SP1 system had been installed recently. The SP1 is not intended to be used as a massively parallel system; in addition to the equipment described till now an Intel Paragon computer is installed to cover massively parallel activities.

The computer center's responsibility for Fortran programming support in this heterogeneous computing environment is to provide a uniform concept for program development and execu-

tion on all platforms. In this paper, considerations for utilizing parallel Fortran dialects (as HPF and other developments) are not included, although extensive research activities exist at our installation, concentrating on shared virtual memory. We will restrict ourselves to sequential Fortran, including vector processing.

Some programmers have already begun to use Fortran 90 in new software projects. But for others, a quick transition to Fortran 90 is not the primary objective. Often they only intend to modify existing programs where they think the Fortran 77 language level to be sufficient. But even in these cases the transition to Fortran 90 will be necessary: first of all, just formally, because the old 77 level compilers will only be supported for the usual midrange period of time; then, because the new compilers will contain features useful for program maintenance, as well as tools and libraries basing on the 90 standard, finally because the users will see that even for small extensions to existing programs the new standard can be helpful.

For the time being, we have three Fortran 90 compilers in use: the NAG compiler (installed at KFA since January 1993), the Cray compiler (installed since January 1994), and the IBM compiler for the RS6000s (installed since March 1994).

The **NAG compiler** [referred below as NAG] claims to be the world's first Fortran 90 compiler. It compiles Fortran 90 to a host-compatible C on all usual UNIX-platforms. This means that the efficiency of program execution is dependent on the optimizing capabilities of the C compiler available on this special platform; generally it will be inferior to the efficiency gained by a proprietary compiler which produces code optimized for the actual platform. The compiler has no platform-specific extensions; so it is able to provide a very good portability. Its standard-conformance, a major design goal, is proven by various validations.

The **Cray compiler** and its compiling system cf90 [referred below as cf90] is a successor of the Cray vectorizing compilers: its main goal is to provide vector processing performance for the Cray architecture. It includes the IBM language extensions, especially the byte oriented type definitions, as `REAL*4.` Is has pragmatics to control vectorization and language extensions primarily designed to control storage in multitasking applications (`AUTOMATIC, POINTER`).

The **IBM compiler** comes with a full implementation of Fortran 90, modestly offered as the new version of the xlf compiler [thus referred here as xlf3]. It includes, in addition to AIX-specific extensions, most of the VS FORTRAN extensions as well as the Cray extensions mentioned above, thus including the typical industry standard extensions. It produces optimized code for the RS6000 architecture. Enhanced optimization, especially parallelization can be supported by means of a pre-processor (not installed at KFA).

KFA sees the necessity to offer all these three compilers to the users: cf90 will become the workhorse for numerically intensive computing on the Cray machines; in addition to this we have to offer it for the users of the supercomputer center. xlf3 will be necessary for the general work to be done on the SP1 and its successor and on many RS6000 workstations.

KFA has many reasons to offer the NAG compiler to the users also in the future, even if proprietary software is available meanwhile:

- Because of its standard-conformance, it is a good reference compiler and can be used for the development of portable, standard-conforming Fortran 90 software, as well as for educational purposes.

- KFA has some experience with Fortran tools in the past (TOOLPACK, NAG tools). So our plans are also to install the NAG Fortran 90 tools as soon as possible, and we expect that these tools will cooperate with the NAG compiler in a very natural way, thus establishing a good base for program development.

- Because of its portability (both of compiler and of produced code) the compiler can be installed on all our UNIX platforms; in addition to that a PC version is available. So Fortran 90 can be used from all platforms (of course except from IBM mainframes).

Besides the advantage of producing optimized code for a given architecture, cf90 and xlf3 will be preferred as proprietary software if a good interface to the respective operating system is used.

Fortran education: In our Fortran programming courses, only Fortran 90 is taught, first on base of the NAG compiler since it was foreseeable that cf90 and xlf3 would be available in the beginning of this year. We have two types of courses: a one week course for beginners (typically university graduates who have learnt other programming languages, as Pascal or C, and will now learn the language needed for their daily work. For the "old" Fortran programmers we offer a course consisting of 3 lessons (each 2 hours) where only the updates compared to Fortran 77 are taught. We prepared some material for these courses which consists of a language overview, an overview on the features that are new compared to Fortran 77, and a collection of simple (mostly numerical) algorithms which will demonstrate the new programming style.

Concerning **style**, emphasis is put on the array handling features, as well as the module concept. Of course we try to teach Fortran 90 as a modern language for which concepts as defined types or operator overloading are as obvious as they have become meanwhile e. g. in C++. We try to prevent our users from using Fortran 77 constructs, and we succeed with this at least for new programming projects. But we must admit that most of our daily Fortran programming business consists of modification and extension of old Fortran 77 codes, so that modern style is accepted as a matter of principle but will show practical consequences only in some time.

Performance considerations with respect to Fortran 90 can have the following objectives:

- to compare the performance of a Fortran 90 compiler with the performance of its Fortran 77 predecessor on the same system,
- to compare different compilers on the same system,
- to compare the performance of different programming styles in the same compiler and on the same system.

For absolute performance data, we will of course be primarily interested in the cf90 performance, because in our computing environment numerically intensive computing will usually be done on the Cray computers.

To get a first impression on the performance behaviour of the different compilers, we made some **comparisons** with a relatively simple program which we also used in our Fortran education as an example for the module concept. It computes the extreme eigenvalue of a matrix by means of the *von Mises* iteration, a numerically antiquated algorithm which, because of its simplicity, may nevertheless show quite well the principles of programming techniques concerning array handling. We started with a Fortran 77 version, as it typically can be found in

collections of mathematical algorithms[1] . We confronted this program with an algorithmically equivalent Fortran 90 version utilizing the concepts of modules and operator definition (an outline of this program may be found in the appendix).

The **results** of different tests according to our test objective scheme can be described as follows: (cf. table in appendix)

- xlf3 and cf90 perform as good as their predecessors, applied to pure Fortran 77 programs,
- NAG has no dramatic performance decrease compared to xlf3, as would have been expected from the different concepts, as far as usual code is concerned. The decrease is more striking if, as for matrix operations, optimized library routines are used.
- Utilization of Fortran 90 style will lead to a better readability and modularity concerning matrix / vector operations. This does not lead to a significant performance degradation for large matrices, but for many operator executions with small matrices, the overhead of the module concept becomes visible.
- For small matrices *inlining* may improve performance to some degree for NAG and xlf3. Better techniques are realized in the Cray compiler where small matrices do not show a significant performance degradation.

We would have liked to give more details on xlf3 here, but because we had some troubles with the new licence managing software coming with this compiler, it has been available for us since two months. With the relatively small test base we could use (Fortran 90 course material, a few medium sized Fortran 77 programs) we did not notice any significant **reliability** problems.

Although Fortran 77 **numerical libraries** are callable from Fortran 90 programs, leading suppliers will offer special Fortran 90 libraries in the near future. These libraries will consequently use the Fortran 90 style, while preserving and to some degree improving the high quality standard of numerical algorithms. Especially with respect to matrix computations, IMSL provides a concept of using defined types and operators which allows to express matrix formula in a most natural way, thus giving even the casual user a good chance for rapid prototyping. Operator overloading allows polymorphism by choosing the appropriate library dependent on the type of the operands. The NAG library will follow similar concepts.

We intend to offer these libraries to our users as soon as they are available: We think that it will be helpful at least for the casual user that the special library calls are hidden from him. (Of course the professional user should retain the opportunity to choose the routine with the specific properties he needs to solve his problem.)

[1] e.g. G. Engeln-Müllges, F. Reutter: Numerik-Algorithmen mit Fortran 77-Programmen, Mannheim 1993

Appendix

Example: *von Mises* iteration

```
MODULE matvec
     INTERFACE OPERATOR (.mv.)
          MODULE PROCEDURE mv
     END INTERFACE
     ....
CONTAINS
FUNCTION mv (A,X)
     REAL, DIMENSION (:,:), INTENT(IN):: A
     REAL, DIMENSION (:  ), INTENT(IN):: X
     REAL, DIMENSION (UBOUND(A,1))    :: mv
     mv = MATMUL(A,X)
END FUNCTION mv
```

```
X = NORMALIZE (START-VECTOR)
DO iter = 1, max_iter
     Y = A .mv. X
     eigenvalue = MEAN(Y/X)
     IF (CONVERGENT (X,Y)) EXIT
     X = NORMALIZE (Y)
END DO
```

Test results: *von Mises* iteration using different styles

f 77:	Fortran 77
f 90:	Fortran 90 without using module concept
f 90 M:	Fortran 90, using module concept
n	size of matrix

n	10	100	300
f 77	1.0	1.0	1.0
f 90	3.1	1.2	0.7
f 90 M	4.5	1.5	0.6

using xlf3; f77 values scaled to 1

10	100	300
2.1	1.5	6.2
3.6	1.2	0.7
5.5	1.5	6.2

using NAG

	xlf3 inline	xlf3	NAG		Cray
f 77	3.2	3.9	17.8		25.4
f 90	21.7	23.1	29.9		33.4
f 90 M	26.7	32.2	53.8		38.5

different styles, different compilers, small matrices
(xlf3 inline, xlf3, NAG same scale, Cray different scale)