Migrating from Fortran 77 to Fortran 90 with the NAGWare f77 and f90 Tools

Ian Hounam and Stuart Hemstock, NAG Ltd

Abstract

The investment in code written in the Fortran language over the last four decades means that there is a vast amount of legacy code written to various standards and incorporating numerous vendor extensions to the language. Meanwhile, the new Fortran 90 standard adds many attractive new features to the language while retaining compatibility with Fortran 77. This paper shows how the NAGWare f77 Tools may be used to standardise Fortran code to a form suitable for porting to standard conforming Fortran 90 compilers. Then the paper goes on to discuss how the NAGWare f90 Tools can be used to convert this code to free format Fortran 90 with standardised precision.

1 Introduction

Fortran has been the foremost computing language for science and engineering for nearly 40 years. During this time the language has been standardised in 1966, 1978 (Fortran 77) and 1991 (Fortran 90). At each stage the vast investment in “legacy code” has been recognised and each new standard has been largely backwards compatible with its predecessor.

Fortran 90, the latest standard, is now available. The NAGWare f90 Compiler, which is available on most Unix systems, VMS and the PC, is now a mature and much used tool offering a consistent compiler across this wide range of systems. Native compilers have now been released on a number of systems and more are scheduled for the near future.

Now is the time to move to the new standard. In the future, compiler development will take place on the new, rather than the old, standard. Even in the USA, where the new and old standards are supposed to co-exist, vendors are unlikely to support both languages to the same level. In order to track the latest developments in compiler technology and move to new hardware, it will become necessary to use Fortran 90.

The new language offers many new features which the Fortran community has needed for some time, such as the better looping constructs, dynamic memory, pointers, optional arguments, recursion and the new whole array operations which speed up program development time and improve readability dramatically. So there is an incentive to move to Fortran 90 to make use of these new features.

Initially, new development may take place in Fortran 90 without even recompiling old libraries of Fortran 77 routines. The NAGWare f90 Compiler makes it easy to call old Fortran 77 libraries by providing a compatibility switch. Eventually, though, it is going to be necessary to recompile these “legacy” libraries. For example, a redesign of the calling sequence of old routines to simplify the interface by using optional arguments with default values might be considered. Another reason for recompilation may be that Fortran 77 compilers cease to be fully supported as Fortran 90 takes over. We show here how the NAGWare tools may be used to aid in this process.
Further development of older code in Fortran 77 may be required. In this case we show how the NAGWare tool suites can be used to transform the old code to a form that allows this further development to take place using the new free format. Indeed some transformations from old constructs to newer forms can be made automatically.

Migration from a Fortran 77 compiler to a Fortran 90 compiler may initially consist solely of recompiling old code. As ANSI standard Fortran 77 is a subset of Fortran 90 this should be easy. However, a large number of compiler extensions are available to the Fortran 77 programmer and these may present problems when changing compilers. Some of these extensions are unique to one manufacturer, while others have been adopted by the majority of manufacturers. The former are a definitive hindrance to portability; the latter may also cause problems. Some former extensions like IMPLICIT NONE are now part of the Fortran 90 standard. Any extensions that are not in Fortran 90 need to be identified before or during porting from Fortran 77.

The initial stage of migration will probably consist of recompilation with the minimum of changes possible applied, though this unfortunately means that many of the exciting new language features will not be incorporated, and some features (such as COMMON blocks) will affect any new code which calls old subroutines.

The existing investment in Fortran is such, however, that the old code must be carried forward to the new standard. NAGWare tools can be used to help with this.

1.1 NAGWare Tools

The NAGWare f77 and f90 Tool Suites are general purpose tool suites designed for everyday use, but which contain functionality invaluable to the task of migrating software from Fortran 77 to Fortran 90.

Software tools can ease much of the work in writing software to conform to stylistic and "good practice" standards which make them easier for another member of a work group (or the original author after a break of some time) to read, understand and maintain the code. The advantage of software tools is the ability to perform large, repetitive tasks when working on a project containing possibly thousands of lines over many separate source files.

Software tools can also be used to identify coding errors that are often undetected by compilers.

In some cases old code can be simply recompiled; when things go wrong the NAGWare tool suites can help you out. As you take advantage of the new facilities of the new language in your old code, the NAGWare f90 Tools will be there to help again.

2 The background to the tools

2.1 The NAGWare f77 Tools

The NAGWare f77 Tools provide advanced analysis and transformation capabilities for developers and maintainers of Fortran 77 software.

These Fortran 77 tools have a strong foundation in over 10 years of work designing, developing and using Fortran 77 processing tools, which itself builds on the efforts which had been applied to Fortran tools before that period.

All the tools use the NAGWare f77 Tools extended Fortran dialect. This consists of ISO/ANSI standard Fortran 77, plus some common extensions that are found in Fortran 77 compilers in general use. The NAGWare f77 Tools language definition includes the
US Military Standard Fortran 77 extensions, e.g. DO WHILE, END DO and IMPLICIT NONE. Other extensions include long variable names, and non-standard data types. This makes it possible to process existing Fortran code which has these extensions. All such extensions are flagged by the analysis tools. With the exception of non-standard data types, e.g. DOUBLE COMPLEX, these extensions to Fortran 77 have been incorporated into Fortran 90.

These tools provide the analysis capabilities necessary to ensure that old Fortran 66 or Fortran 77 code conforms to the standard necessary to port to Fortran 90.

2.2 The NAGWare f90 Tools

The NAGWare f90 Tools are based upon the proven technology of the NAGWare f90 Compiler. They currently provide basic Fortran 90 transformational functions. The tools’ front end is able to analyse both fixed format and free format Fortran programs; the output is always free format. Although there is no separate analyser tool, the tools produce warnings about non-standard and obsolescent features etc as they process the code. The NAGWare f90 Compiler itself provides full ISO/ANSI Fortran 90 standard checking. The Fortran accepted is ISO/ANSI Fortran 90 with no omissions and a small number of optional extensions; all use of extensions produces clear warnings, unless these are deliberately suppressed.

3 The NAGWare f77 Tool Suite

The NAGWare f77 Tools can be divided into two classes: transformers and analysers. The transformers carry out automatic transformations on Fortran 77 code. These can be used to make repetitive changes to code easy and eliminate errors introduced by “hand editing”. The principal analyser gives a more rigorous approach to verification of Fortran 77 code against the ANSI standard than, in general, compilers do. This analysis is also extended to highlight non-portable usage of Fortran 77 features. The main features of the NAGWare f77 Tools are listed below.

All of the NAGWare f77 Tools are useful to a greater or lesser extent in migrating from f77 to f90. Some specific examples of such use are shown below.

TRANSFORMERS

nag_apt

The Arithmetic Precision Transformer converts single precision programs to double precision or vice versa, automating this normally error prone task. It converts declarations (including adding declarations if implicit typing was used), constants, edit descriptors and non-generic intrinsic functions (if appropriate). EQUIVALENCE declarations are checked to see if they are invalidated by the operation.

When migrating, nag_apt can be useful in converting non-standard type declarations accepted by many Fortran 77 compilers (though not part of the standard) but which are not in the new standard: DOUBLE COMPLEX and REAL*8 are two examples. Where this necessitates a change of precision the nag_prest90 tool can convert a program to the precision required in Fortran 90.
Converting DOUBLE COMPLEX to COMPLEX*16, which is acceptable to some Fortran 90 compilers and the NAGWare f90 Tools, can be done by nag Apt using the -byte option, but this generates warning messages if no precision transformation is carried out. This task is usually best done using nag Decs, see below.

**nag chname**

This tool allows the user to change the names of symbols used throughout a program. Changes are selectively done on symbols, strings, Holleriths and comments. Changes can be simple string replacements or complex regular expressions. Character case can be significant or not at the user's discretion.

When migrating: nag chname can be used to rename procedures across a whole project where a procedure name used conflicts with one of the new Fortran intrinsics. Also long names may be substituted for meaningless Fortran 77 short names; note that this may also be done by the NAGWare f90 tool nag chname90.

**nag decs**

The declaration standardiser standardises the declaration of variables and external procedures. The tool can be used to add declarations for implicitly typed items, to remove declarations for unused items or simply to ‘tidy up’ the declarations. This tool can improve dramatically the readability of old code. nag decs optionally changes all Fortran 77 specific intrinsic functions to the generic form; for example, DMAX1 can be changed to MAX. The section headers added by nag decs are totally compatible with nag decs90.

When migrating: Start as you mean to go on! Convert your program to 100% explicit declarations. This can also be done by nag decs90, which can add an IMPLICIT NONE statement to each program unit. The main advantage of nag decs over nag decs90 in porting is to convert DOUBLE COMPLEX declarations to COMPLEX*16 (using the -no_double_complex option). DOUBLE COMPLEX is not legal input to the NAGWare f90 Compiler and cannot be input to the f90 Tools and so must be converted.

**nag lvi**

The Local Variable Initialiser adds code to each program unit to initialise local variables to a given value. This value may be optionally provided by the user. This tool may be used as a debugging aid where a problem of uninitialised variables is suspected.

When migrating: nag lvi will enable users whose compilers set all uninitialised data to a predetermined value, typically zero, to set that value when moving to a Fortran 90 compiler.

**nag polish**

The Fortran 77 “pretty printer” tidies up source code. This tool can be used to standardise the layout of the code. It is configurable and almost every aspect of the layout of the code may be changed: spacing within and between the lines, case, labelling, etc.
nag_profile

The profiler creates a modified form of the input source that has code automatically added
to count at run time the execution frequency of segments of the program. The modified
program may be run many times, building up a history of the execution frequencies over
multiple test data sets.

nag_struct

The Restructurer rebuilds the control flow of a Fortran 77 program unit creating block IF
constructs wherever possible. It may also optionally create DO WHILE constructs. This
tool makes Fortran 66 spaghetti code look more like Fortran 77.

When migrating: nag_struct will update (where it can) old control structures,
some of which have now been declared obsolescent and will be removed in a
future Fortran standard. Use the -while and -enddo options to generate DO
WHILE loops and terminate DO loops with END DO.

ANALYSERS

In addition to the sections highlighted below for use in migration, all of the analysers
produce information which is very useful to the programmer in performing the modification
tasks which must be performed by hand.

nag_pfort

The Portability Verifier checks Fortran 77 source code against the ANSI language standard
and reports on non-portable usage of Fortran 77 features. The dialect of Fortran 77
supported by the tool suite is ANSI, but nag_pfort issues warnings for non-compliance
with a subset of the language known to be portable. There are many extra checks; for
example, a common block may not appear in more than one BLOCK DATA; a DO loop
must have an integer control variable; only standard intrinsic functions may be used.

When migrating: nag_pfort will highlight any non-standard Fortran 77 and
show up many errors in the original program code which Fortran 77 compilers
may ignore. Many of these extensions will be unacceptable to your Fortran 90
compiler.

A new nag_pfort option has recently been added to aid in porting to Fortran
90. Many users will have Fortran 77 code that uses long variable names and US
Military Standard extensions. By default nag_pfort will flag these legal Fortran
90 features as non-standard Fortran 77. The -port90 option allows long names
up to 31 characters, IMPLICIT NONE, INCLUDE and END DO\textsuperscript{1}.

nag_fcalls

Generates a call tree, which is a listing of what routines are called by a routine. This tool
also optionally lists which routines call each routine.

\textsuperscript{1}The -port90 option is currently available only as an extension to the standard nag_pfort tool which is
supplied on request to customers with the optional support service.
When migrating: It often useful to know the call tree of your program. Indeed you may discover routines that are no longer called and can be eliminated. It would indeed be a waste of time porting a difficult routine that had been superseded!

**nag_fxfref**

The cross referencer tool creates a table showing the line numbers of declaration, definition and use of all variable names in the source code.

When migrating: This tool can provide useful information on the use of variables.

**nag_libdoc**

This tool analyses the source code for a subroutine or subroutine library and generates the information required to document the routine’s interface. The output shows, if a full call tree is provided, those routine arguments that are input only. An example program is provided showing how to read the output of this tool and generate a skeleton document in a form similar to the NAG Fortran Library documentation. This example may be copied and modified by the user to produce customised subroutine library documentation.

When migrating: This tool can provide useful documentation for porting.

**nag_metrics**

The metrics tool analyses Fortran source code and calculates a variety of software metrics. These include simple counts of the size of the program like the number of executable statements. Complexity metrics including McCabe’s cyclometric number, Knots and the Halstead’s metrics can also be calculated.

When migrating: This tool can provide basic useful statistics on the size of the porting task.

4 The NAGWare f90 Tools

**nag_chname90**

This tool systematically replaces names in Fortran 90 source code and optionally in comments and/or strings.

When migrating: nag_chname90 will allow you to change cryptic six-letter names accepted by Fortran 77 to meaningful names for Fortran 90. It can also rename procedures whose names clash with new Fortran intrinsic functions.

**nag_decs90**

The Declaration Standardiser has two modes.

1. It rebuilds the complete declaration section of all program units, declaring all names whether explicitly or implicitly typed. These new declarations are divided into logical sections by comments inserted by the tool.
2. It can declare all implicitly typed names leaving existing declarations unchanged.

When migrating: Convert your declarations to the new Fortran 90 style with nag_decs90. This will give the program the correct Fortran 90 look and ensure everything is declared explicitly, and by continuing to use this tool you can ensure that everything stays declared as the software develops over its lifetime.

nag_decs90 will convert old-style declarations like INTEGER*2 to INTEGER(KIND=2), and COMPLEX*16 to COMPLEX(KIND=2). Hence nag_decs and nag_decs90 may be used together to convert DOUBLE COMPLEX Fortran 77 code.

nag_polish90

The Fortran 90 “pretty printer” tidies up source code. It can be used to standardise the layout of Fortran 90 code and has many user-configurable options. As with the f77 Tools, the polisher is incorporated into the other tools; hence all output from the NAGWare f90 Tools is polished. The polisher is driven by an options file for which there is a separate editor.

When migrating: This tool, in common with all the others, will convert fixed format to the new free format style.

nag_polopt90

nag_polopt90 is provided to create or modify options files for all of the transformational tools. A large number of parameters are available to allow an acceptable style to be created.

nag_prest90

This tool changes source code to a standard arithmetic precision which is specified through a parameterised KIND number. The output from this tool may be converted from single to double precision, or vice versa, by a small modification to one statement in each program unit.

When migrating: Parameterise the KIND of all REAL and COMPLEX data types.

4.1 Future Enhancements

Release 2 of the NAGWare f90 Tools is only a start. NAG is committed to extending this tool suite. In the future further transformational tools will be added with an emphasis on tools to “modernise” old Fortran constructs.

There are no analyser tools in the initial releases because the NAGWare f90 Compiler itself provides a complete check of standard conformance. As other Fortran 90 compilers become available, a standards conformance and portability verifier will be developed, if required. For NAGWare f90 Tools users who do not have access to the NAGWare f90 Compiler, the tool nag_modules90 (the module builder) provides some of the analysis capabilities of the NAGWare f90 Compiler.
Some likely tools for the near future are:

**A Restructurer.** This will restructure old code adding in new control-flow command etc; this will be similar to the f77 tool nag_struct but will support the new language constructs.

**Profiler.** This will add code to count execution frequencies; similar to the f77 tool nag_profile.

**COMMON BLOCK to MODULE converter.** This tool will replace (where possible) COMMON blocks, creating MODULEs to allow the data to be shared in a more controlled way.

**Interface block builder.** Will build interface declarations automatically.

5 Migrating from Fortran 77 to Fortran 90 with the NAGWare Tools

Moving from Fortran 77 to Fortran 90 should be just a matter of recompiling. This is not always so and there are two reasons for that: firstly, very little Fortran 77 code in existence is actually completely standard conforming with no extensions; and secondly there are some differences between Fortran 77 and Fortran 90 though these are minor and unlikely to cause problems as they consist of “constraints” to existing code.

**Fortran 77 Tools**

The Fortran 77 tools are an excellent first step in migrating software as they were developed to enable portable software to be produced.

The first tool use should be nag_pfort. This performs rigorous checks ensuring that the code conforms to the strict ISO/ANSI standard and warns where implementors are known to have varied in their interpretation. This allows potential problems to be found and corrected easily before porting to any other Fortran compiler. It can often find previously undiscovered ‘bugs’ in the Fortran 77 code such as argument list mismatches, even in code that has been in use for some time. Use this tool, with the -port90 option to suppress warnings related to legal Fortran 90, to highlight any non-standard features that will hinder porting to Fortran 90. Correct these and your code is in a much better state for porting to any Fortran compiler, including Fortran 90.

Much Fortran 77 code in use for science and engineering may make use of the double precision complex data type. Unfortunately, the Fortran 77 standard did not provide this useful type. Most compilers do include DOUBLE COMPLEX, often with the alternative representation COMPLEX*16, and some compilers only have COMPLEX*16. The Fortran 90 standard does have the double precision complex data type which can be represented as, for example, COMPLEX(KIND=KIND(0.0D0)). Although Fortran 90 compilers may allow the older representation of this data type, it is a good idea to convert this. The first step is to use nag_decs with the -no_double_complex option to turn DOUBLE COMPLEX to COMPLEX*16 which is acceptable to the NAGWare f90 Tools. This can then be converted to the new Fortran 90 forms; see below.

Some procedure names used in Fortran 77 programs may clash with one of the new Fortran 90 intrinsics (there are 75 new intrinsics). nag_chname can be used to change these.
Also short names may be converted to longer more meaningful name. The advantage of a tool such as nag_chname over, say, a text editor is that it understands more about the text it is reading and so can avoid, if selected, changing strings or comments.

The restructurer nag_struc may be used to improve the structure of really old code, or to add the DO WHILE construct (with -while) and ENDDO loop terminators (with -enddo).

The NAGWare f77 Tools are useful both in the initial push to make Fortran 77 code compilable with a Fortran 90 compiler, and also if it should be felt that the migration is a good time to “tidy up” old code.

Summary

- Use nag_pfort -port90 to highlight non-standard Fortran features.
- Eliminate non-standard Fortran features.
- Transform DOUBLE COMPLEX to COMPLEX*16 using nag_decs.
- Use nag_chname to replace short names and avoid name clashes.
- Use nag_struc with -while and -enddo to modernise loops.

Fortran 90 Tools

It is no longer necessary to code Fortran in the old card image format; this is the time to convert to the more modern free format. Any of the NAGWare f90 Tools carries out this transformation, but if none of the other f90 tools is used, nag_polish90 will make this conversion.

The process of converting from DOUBLE COMPLEX may be continued by processing with nag_prest90, which can convert all REAL and COMPLEX declarations to use a parameterised KIND number.

Summary

- Use nag_prest90 to parameterise REAL and COMPLEX data types.
- Use nag_decs90 to convert non-standard data types.
- Use nag_chname90 to avoid name clashes and introduce long meaningful names (if nag_chname was not used previously).
- If no other f90 tool is used, use nag_polish90 to convert to free format.

6 Conclusion

Moving from Fortran 77 to Fortran 90 should be a trivial task, but problems may occur; when this happens a set of good software tools can move you on quickly past these hurdles quickly and painlessly, even for large projects. The NAGWare tool suites are designed for everyday use, not solely crossing the standards barrier, and they can help maintain programs and libraries before, during, and after the migration phase.