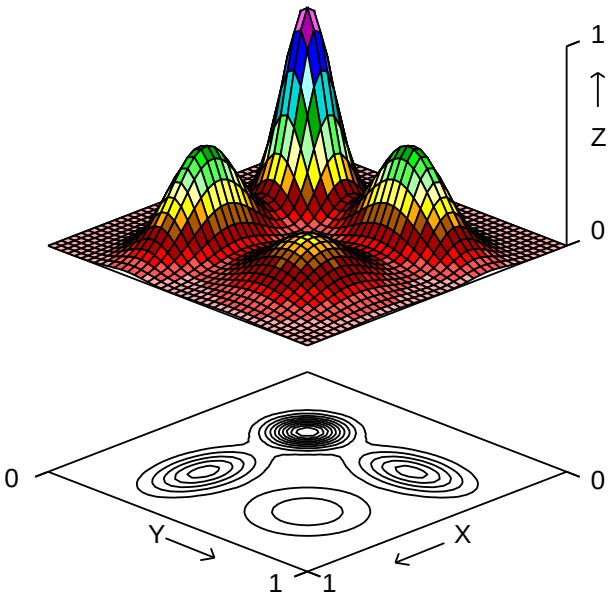
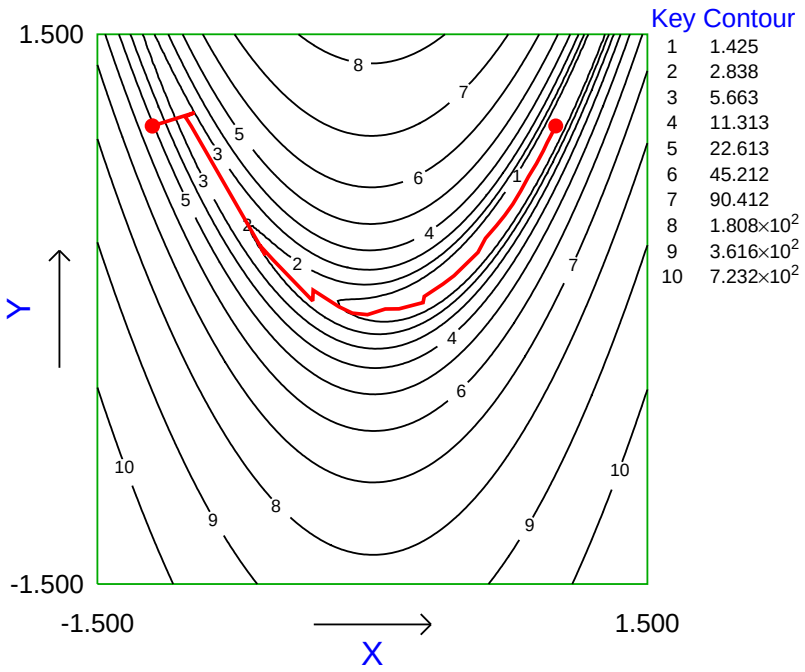
TECHNICAL GUIDE TO COMPILING AND LINKING

SIMFIT AND SIMDEM

SIMFIT 3D plot for z = f(x,y)



Contours for Rosenbrock Optimization Trajectory



Contents

1 Building SIMFIT version 7.4.2 onwards

The procedure has been greatly simplified as will now be described. However, the previous detailed instructions are still included in this document for older versions and to build versions using the NAG library. First of all the latest zip file is unzipped and the main folder `c:\simzip` renamed as `c:\simfit7` (or `c:\simfit8`)

The 32-bit version

Change to the work folder, e.g., `c:\simfit7\work` and simply type

`make_everything` to execute the batch file `make_everything.bat`.

The 64-bit version

Change to the work folder, e.g., `c:\simfit_x64\work` and simply type

`x64_make_everything` to execute the batch file `x64_make_everything.bat`.

2 Building SIMDEM version 7.4.2 onwards

The procedure has been greatly simplified using only two dlls (`simdem32.dll` or `simdem64.dll`) for the FTN95 versions as will now be described. However, the previous detailed instructions are still included in this document for older versions and to build versions using the NAG and gFortran versions. First of all the latest zip file is unzipped and the main folder `c:\demzip` renamed as `c:\simdem`.

The 32-bit version

Change to the folder `c:\simdem\simdem32` and simply type

`make_simdem32` to execute the batch file `make_simdem32.bat`.

The 64-bit version

Change to the folder verb+c:SIMDEMSIMDEM64+ and simply type

`make_simdem64` to execute the batch file `make_simdem64.bat`.

The NAG and gFortran versions

These require the appropriate Clearwin DLLs (`clearwin.dll` or `x64_clearwin.dll`) which can be copied from the latest SIMFIT installation, or built independently using FTN95. The batch files and link files supplied must be edited to correct the paths to the NAGfor or gFortran compilers. For instance, the batch files `nag32.bat` or `nag64.bat` can be used to generate object files using NAGfor while `strict.bat` can be used to generate 64-bit object files using gFortran.

3 The zip files

SimFIT is a large package and, to allow programmers to compile subsections of the package, for instance just the SimFIT GUI, or only one or a selection of the NAG library routines used by SimFIT, the code is available from `https://simfit.org.uk` in several forms.

From versions 7 onwards the SimFIT and SimDEM codes are distributed in the following zip files where `x` is the version (e.g., 7 or 8), `y` is the level and `z` is the release number at level `y`.

```
demzipx_y_z.zip ... the Simdem package
simzipx_y_z.zip ... the Simfit package
manzipx_y_z.zip ... the reference manual
nagzipx_y_z.zip ... the NAG library interfacing code
naglibx_y_z.zip ... The NAG library source code
```

Note that the Fortran source in these zip files has been compiled and run successfully using FTN95, NAGfor, and gFortran under the strictest checking options. Most of the SimFIT codes do not require the Windows API, and those that do are collected together in the `clearwin` folder, and they have a `w_` underscore prefix as in `w_config.for`. The driver programs `w_simfit.exe` and `x64_simfit.exe` also use the Windows API.

These zip files are intended for the use indicated.

- **The demzip package**
  Only provides code necessary to compile and link the SimFIT GUI, that is the SimDEM package.
- **The simzip package**
  This is the complete code for the whole of the SimFIT package.
- **The manzip package**
  All the LaTeX sources and graphics files needed to create the SimFIT reference manual and tutorials.
- **The nagzip package**
  This is provided for users who wish to upgrade SimFIT to use a new release of the NAG library DLLs.
- **The naglib package**
  This code enables users to compile all the NAG library codes used by SimFIT or a subsection of these codes.

3.1 The SimFIT source code tree

The zip files used to distribute the SimFIT codes assume the following code tree structure of folders below the top-level simzip folder. The root `\simzip\` would be changed to `c:\simfit7\` or `c:\simfit8\` as required.

```
\simzip\work
\simzip\dll\simfit
\simzip\dll\clearwin
\simzip\dll\menus
\simzip\dll\graphics
\simzip\dll\maths
\simzip\dll\models
\simzip\dll\nag
\simzip\dll\numbers\...
```

- **The work folder**
  All the source codes, icons, batch files and link scripts used to compile the SIMFIT package driving programs.
  These codes must be linked to codes in the other folders, usually the DLLs

  ```
  w_simfit.dll    (or x64_simfit.dll)
  w_clearwin.dll  (or x64_clearwin.dll)
  w_menus.dll     (or x64_menus.dll)
  w_graphics.dll  (or x64_graphics.dll)
  w_maths.dll     (or x64_maths.dll)
  w_models.dll    (or x64_models.dll)
  w_numbers.dll   (or x64_numbers.dll)
  ```

- **The dll simfit folder**
  Subroutines called from the driving programs. These codes must be linked to codes in the other folders, usually the DLLs

  ```
  w_clearwin.dll  (or x64_clearwin.dll)
  w_menus.dll     (or x64_menus.dll)
  w_graphics.dll  (or x64_graphics.dll)
  w_maths.dll     (or x64_maths.dll)
  w_models.dll    (or x64_models.dll)
  w_numbers.dll   (or x64_numbers.dll)
  ```

- **The dll clearwin folder**
  Part of the SIMFIT GUI.
  The interface to 32-bit Clearwin. Includes *.html and *.jpg codes as well as *.for subroutines.
  This code is free-standing and must not be linked to any of the other SIMFIT folders.

- **The dll menus folder**
  Part of the SIMFIT GUI.
  These subroutines filter calls from elsewhere into a form suitable for calling routines in the clearwin folders. There are also free-standing items to control input/output.
  This code must be linked to the clearwin code in `w_clearwin.dll` or `x64_clearwin.dll`.

- **The dll graphics folder**
  Part of the SIMFIT GUI.
  Code to prepare graphics calls before calling items in the clearwin folder. These codes must be linked to codes in the DLLs

```
w_clearwin.dll  (or x64_clearwin.dll)
w_menus.dll     (or x64_menus.dll)
```

- **The dll maths folder**
  Subroutines with the same names and calling sequences as the NAG library. This code is linked into the academic version of SIMFIT. It must be linked to `w_numbers.dll` or `x64_numbers.dll`.

- **The dll nag folder**
  Subroutines with the same names as those in the maths folder except that they are *.f rather than *.for. This code is linked to the NAG library DLLs so that the NAG version of SIMFIT calls the NAG library rather than the SIMFIT maths library,

- **The dll models folder**
  Subroutines for user-supplied models.
  These codes must be linked to codes in the other folders, usually the DLLs

```
w_maths.dll   (or x64_maths.dll)
w_menus.dll   (or x64_menus.dll)
w_numbers.dll (or x64_numbers.dll)
```

- **The dll numbers folder**
  Public domain code for numerical analysis called by SIMFIT and subroutines in the maths folder.
  This code is free-standing and must not be linked to any of the codes in the other SIMFIT folders.

3.2 The SIMDEM source code tree

The SIMDEM package is intended to demonstrate to Fortran programmers how to write Fortran programs that use the Windows API to create menus, tables, and graphs without knowing anything about the API. All the subroutine arguments are in standard Fortran and, as it relies on the FTN95 Clearwin functionality and runtime system, it is only useful for Windows programming.

The zip files used to distribute the SIMDEM codes assume the following code tree structure of folders below the top-level `demzip\` folder renamed to `c:\simdem`.

```
\demzip\dll\clearwin
\demzip\dll\menus
\demzip\dll\graphics
\demzip\chm
\demzip\extras
\demzip\for
\demzip\f95
\demzip\nagfor
\demzip\simdem32
\demzip\simdem64
```

The zip files used to distribute the SIMDEM codes assume the following code tree structure of folders below the top-level folder.

- **The chm folder**
  HTML files to create the `simdem.chm` compiled HTML help file.
- **The dll folder**
  The SIMFIT GUI as explained for the SIMFIT zip files.
- **The dem folder**
  Test files.
- **The eps folder**
  Encapsulated PostScript graphics files.
- **The f95 folder**
  Free format code.
- **The for folder**
  Fixed format code.
- **The html folder**
  HTML files for the free-standing SIMDEM help program.
- **The nagfor folder**
  Code and batch files to build the NAGfor SIMDEM package.
- **simdem32**
  Code and batch files to create the Silverfrost 32-bit version of `simdem` linked to the single run-time system `simdem32.dll`.
- **simdem64**
  Code and batch files to create the Silverfrost 64-bit version of `simdem` linked to the single run-time system `simdem64.dll`.

4 Overview

Experienced users can just go to the final section on makefiles where there are the sequences of command lines necessary to compile and link the SIMFIT and SIMDEM packages. These can be used to construct makefiles if makefiles are not distributed with the codes. Otherwise, details and examples follow.

4.1 Websites

The SIMFIT, SIMDEM and LaTeX source codes can be downloaded as zip files from

```
https://simfit.org.uk
```

and they should be unzipped into the tree structures provided. However, it may be necessary to change the logical drives (e.g. C: instead of D:) or edit some paths to get all of the batch files and link scripts to work.

Note that the utility program **for2f95**, distributed with the SIMDEM package, can be used to transform `*.for` files into `*.f95` files, if that is required. It was designed to respect features of the code employed to aid readability and checking and should be used rather than general purpose fixed to free translators which will destroy such carefully designed structures.

There are also two SIMFIT mirror sites as follows.

```
http://simfit.usal.es
http://simfit.silverfrost.com
```

4.2 Summary

SIMFIT and SIMDEM from version 7 onwards can be compiled and linked in such a way that there are no cross compiler problems, and the resulting packages will run in all versions of Windows from XP, as well as Linux under Wine, and Macintosh under VMware or Crossover. Details are given for Silverfrost FTN95, which must be used without the `/f_stdcall` switch for standard cdecl Silverfrost applications, but with the `/f_stdcall` switch for the NAG library versions, and also for NAGfor which must be used with the `-compatible` switch for NAG 32-bit DLL applications.

This is very important, and is mentioned several times in this document for emphasis, as failure to observe the advice about not mixing cdecl and stdcall 32-bit binaries leads to run time crashes that can be very difficult to trace.

To compile and link SIMFIT to only create the Academic 32-bit versions then there is no need to use the STDCALL calling convention, but for 32-bit versions that may be linked to the NAG library DLLs it is vital to use the STDCALL calling convention. It is not possible to mix binaries with both calling conventions in 32-bit applications. If the NAG library is going to be used with FTN95 then all the code must be compiled using the `/f_stdcall`

compiler switch, and with NAGfor the compiler switch `-compatible` must be used everywhere. These complications do not apply to the 64-bit versions.

4.3 The installation folders

The default installation schemes for the packages are as follows.

For SIMDEM

```
C:\Program Files\Simdem\bin    ... binaries
C:\Program Files\Simdem\dem    ... demonstration test data sets
C:\Program Files\Simdem\doc    ... documentation
C:\Program Files\Simdem\f95    ... free format code
C:\program Files\Simdem\for    ... fixed format code
```

Both 32-bit and 64-bit applications are placed into in the same folder

```
C:\Program Files (x86).
```

For SIMFIT

```
C:\Program Files\Simfit\bin ... binaries
C:\Program Files\Simfit\dem ... demonstration test data sets
C:\Program Files\Simfit\doc ... documentation
```

The 32-bit application would be installed in

```
C:\Program Files (x86)\
```

but with 64-bit Windows the installation would be in the following tree

```
C:\Program Files\
```

The source code has been written to be consistent with these structures. Some other features are now considered.

4.4 FTN95 and w_clearwin.dll and x64_clearwin.dll

All the Silverfrost-specific calls are now in just one dynamic link library, namely `w_clearwin.dll` (or `x64_clearwin.dll` in 64-bit versions). This must be compiled using Salford-Silverfrost FTN95, as it uses `winio@` and other calls that rely on the Silverfrost run time system, `salflibc.dll`. From Version 6.8.1 the files
`w_clearwin.dll`,
`run6`, and
`change_simfit_version`
are now the only parts of SIMFIT that are FTN95 specific and rely on the Silverfrost run time system.

4.5 Special versions of FTN95 SimDEM

From Version 7.4.0 onwards the three run-time dlls used by the SimFIT package are replaced by just one, i.e. `simdem32.dll` in 32-bit versions and `simdem64.dll` in 64-bit versions. The three separate dlls must still be used by other compilers to avoid cross-compiler problems.

The reason for this is because `w_clearwin.dll` and `x64_clearwin.dll` do not use open, close, inquire, backspace, rewind, read, write, or any actions that would restrict cross-compiler use. The files `w_menus.dll`, `w_graphics.dll`, `x64_menus.dll`, and `x64_graphics.dll` would then be compiled by the native compiler, such as NAGfor, gFortran, etc.

4.6 FTN95 and compiled HTML

All the compiled HTML help for the SimFIT and SimDEM executables is in `w_clearwin.dll` which must be compiled using Silverfost FTN95. A compiled HTML SimDEM help file called `simdem.chm` is now installed by the SimDEM installation package.

4.7 change_simfit_version.exe and NAG DLLs

There is a SimFIT program called **change_simfit_version** that can be compiled using FTN95, or could easily be re-written to be compiled by any compiler. This program can do the following tasks:

```
Overwrite w_maths.dll using academic_maths.dll
Overwrite w_maths.dll using fldll20_maths.dll
Overwrite w_maths.dll using fldll214a_mkl.dll
Overwrite w_maths.dll using fldll214z_mkl.dll
Overwrite w_maths.dll using fldll214a_nag.dll
Overwrite w_maths.dll using fldll214z_nag.dll
Overwrite w_maths.dll using fldll215z_nag.dll
Overwrite w_maths.dll using ************.dll
```

(and corresponding 64-bit dlls) and it is configured by the file `change_simfit_version.config` or `x64_change_simfit_version.config`. This results in a consistent version of SimFIT that is either free standing (Academic) or NAG DLL based. The package can also be distributed without the utilities

change_simfit_version.exe,or **x64_change_simfit_version.exe**

but fixed into one of these configurations. The SimFIT program **change_simfit_version.exe** can be run as administrator, but only when SimFIT is switched off so as to not be linked to `w_maths.dll`. The Academic and NAG versions only differ in the version of `w_maths.dll` that is in the same folder as the rest of the SimFIT binaries.

Program **change_simfit_version** does not use open, close, inquire, backspace, rewind, read, write, or any actions that would restrict cross-compiler use.

4.8 w_simfit.exe

The SIMFIT driver program `run7.exe = w_simfit.exe` must be compiled using FTN95 but could, with some difficulty, be replaced by a new driver written for any compiler, using any language. It links to object code from `dllchk.for` which must be edited for a correct signature.

`run7 = w_simfit.exe` does not use open, close, inquire, backspace, rewind, read, write, or any actions in such a way as to restrict cross-compiler use.

4.9 Cross compiler issues

If the main programs and dynamic link libraries are compiled and linked using the same compiler, e.g. FTN95, NAGfor, etc., there will be no cross compiler problems, as all open, close, read, write, inquire, etc. will be using the same run-time system. The resources can be compiled using the Silverfrost SRC compiler or using other resource compilers, such as `windres` supplied with MinGW `gcc` and NAGfor. The HTML required by `w_clearwin.dll` can only be compiled using SRC.

In the SIMDEM examples documentation it is explained how to use special subroutines and functions to perform, read, write, open, close, inquire, etc. to circumvent the situation where code calling the SIMDEM GUI is not compiled by the same compiler as the GUI.

4.10 File extensions

```
*.f95  ... Fortran file in free format
*.for  ... Fortran file in fixed format(main programs and dll)
           Some are single routines but many are composite.
           Some use long names and allocate/deallocate.
*.ins  ... Fortran file in fixed format(included routines)
           Some are single routines but many are composite.
           Some are .ins files defining common blocks etc.
           These are being phased out in favor of modules.
*.f    ... Front end code for the NAG library calls
*.rc   ... Resource script for SRC (the Salford resource compiler)
           These can also be compiled using windres.
*.ico  ... Icon (for *.rc scripts)
*.htm  ... HTML script (for *.rc script)
*.link ... Link script for SLINK or SLINK64 (or NAGfor)
*.bat  ... MS DOS batch file
*.tex  ... LaTeX script
*.wgb  ... EPS file minus the prolog (prolog.wgb)
*.eps  ... EPS file
*.cpp  ... C code
```

4.11 Scripts

The source codes, when unzipped, contain batch files and link scripts, so that the process is extremely simple. The batch files all suppose that Silverfrost FTN95 is on the path, but this is only strictly necessary for three items:

```
w_clearwin.dll             ... Simfit and Simdem
run7.exe = w_simfit.exe    ... Simfit only
change_simfit_version.exe  ... Simfit only
```

Otherwise, by making appropriate replicas of the batch files and link scripts, any Fortran compiler can be used.

It is also assumed that the source codes for

```
w_clearwin.dll (and x64_clearwin.dll)
w_menus.dll (and x64_menus.dll)
w_graphics.dll (and x64_menus.dll)
```

are identical in the SIMDEM and SIMFIT packages. In the event of dedicated NAG and Silverfrost versions, in future this may not always be the case.

5 Source codes

Download and unzip the latest `versionx_y_z` zip files as follows:

```
demzipx_y_z.zip ... the Simdem package
simzipx_y_z.zip ... the Simfit package
manzipx_y_z.zip ... the reference manual
nagzipx_y_z.zip ... the NAG library interfacing code
naglibx_y_z.zip ... The NAG library source code
```

The SIMDEM package will be unzipped into `C:\demzip`
The SIMFIT package will be unzipped into `C:\simzip`
The reference manual will be unzipped into `C:\manzip`
The NAG library interfacing code will be unzipped into `C:\nagzip`
The NAG library source code will be unzipped into `C:\naglib`

After unzipping, the source codes can be used to update existing installations.

If you decide to unzip elsewhere it will all be very much harder

Note that the source codes for

```
w_clearwin.dll
w_menus.dll
w_graphics.dll
```

in `demzipx_y_z.zip`, `simzipx_y_z.zip`, and `nagzipx_y_z.zip` may not always be identical.

5.1 Code style

The SIMFIT code does contain some obsolescent features, e.g. `COMMON` blocks and `GOTO`s, but I am steadily replacing these. There are no equivalences, entries, Holleriths, subroutine calls creating side effects, or any of the well known howlers that Fortran allows.

All subroutines are heavily commented, but observers will note how the style has changed progressively from the days when we had to trap errors using things like

```
READ (NIN,100,END=20,ERR=40)
```

so that, in general, routines in upper case with labels and GOTOs will tend to be older than code in lower case with things like

```
read (nin,100,iostat=ios)
if (ios.ne.0) then...
```

At one stage the code never used things like

```
DO I = 1, N
   K(I) = L(I + 1) + 2
ENDDO
```

because of confusion between INTEGER*1, INTEGER*2, and INTEGER*4, and there are many integers defined in parameter statements because of this, as in

```
INTEGER    N1, N2
PARAMETER (N1 = 1, N2 = 2)
...
DO I = N1, N
   K(I) = L(I + N1) + N2
ENDDO
```

Subsequently, I did maintain this feature so that integers used explicitly in a subroutine were all declared and could be easily traced.

Another feature is that I tend to use argument lists like this

```
CALL SOME_THING (I, J, K,
                 A, B, C,
                 XTITLE, YTITLE, ZTITLE,
                 ABORT, OK, QUIT)
```

with integers, then double precisions, then characters, then logicals, all in alphabetical order within their type. This helps type checking but was not always done with older code.

Note that using code with unnecessary continuation lines like

```
      call putadv (
     +'Input a file like manova1.tf1')
```

instead of just

```
      call putadv ('Input a file like manova1.tf1')
```

was adopted to make the work of the Spanish translators easier

5.2 Signatures

All SIMFIT programs have signatures to identify the version and release numbers, and these are constantly checked during normal operation so that users can be warned of any inconsistencies. All binaries in a SIMFIT installation must have the same signature, so you must edit the signature codes for version and release numbers as follows:

For the SIMDEM package:

```
C:\simfit7\dll\menus\dllmen.for
C:\simfit7\dll\graphics\dllgra.for
C:\simfit7\dll\clearwin\dllclr.for
C:\simdem\simdem.for
C:\simdem\for\simdem.for
C:\simdem\f95\simdem.f95
```

For the SIMFIT package:

```
C:\simfit7\work\dllchk.for (and x64_dllchk.for)
C:\simfit7\dll\simfit\dllsim.for
C:\simfit7\dll\menus\dllmen.for
C:\simfit7\dll\graphics\dllgra.for
C:\simfit7\dll\models\dllmod.for
C:\simfit7\dll\numbers\dllnum.for
C:\simfit7\dll\clearwin\dllclr.for
C:\simfit7\dll\maths\dllmat.for
C:\simfit7\dll\nag\dllmat_mark20.f  ... now done by makenag.bat
C:\simfit7\dll\nag\dllmat_mkl214a.f ... now done by makenag.bat
C:\simfit7\dll\nag\dllmat_mkl214z.f ... now done by makenag.bat
C:\simfit7\dll\nag\dllmat_nag214a.f ... now done by makenag.bat
C:\simfit7\dll\nag\dllmat_nag214z.f ... now done by makenag.bat
```

For **change_simfit_version.exe** in the SIMFIT package edit the file `change_simfit_version.config` stored in the `C:\setup\programs` folder.

For the reference manual version and release numbers:

```
C:\manuals\manual0\color.tex
C:\manuals|manual0\mono.tex
```

6 Compilers

Examples are given for Silverfrost FTN95 and NAGfor but, except for one essential item and three nonessential auxiliary items for which FTN95 must be used, any Fortran compiler can be used. Note that most compilers can create binaries consistent with either the cdecl calling convention, or the stdcall calling convention. It is possible to link executables to DLLs built using either convention but, in general, it is best to use just one of these conventions, e.g. stdcall for Excel, Visual Basic, NAG library DLLs, etc. 64-bit versions can also be compiled using NAGfor or gFortran.

6.1 Example 1: FTN95 and w_clearwin.dll

As an example of how to use FTN95, the complete procedure for creating `w_clearwin.dll` will be described. This DLL is an essential part of SIMFIT and SIMDEM and must be compiled using the Silverfrost FTN95 compiler.

6.1.1 Configuring FTN95

First of all, the command

```
ftn95 /config
```

must be used to configure the compiler for either
a) cdecl (default) for some C programs, or

b) stdcall (for VB, Excel, NAG DLLs, Windows API, etc.)
Note that `/f_stdcall` compromises some `/checkmate` functionality.

6.1.2 Compiling the resources

Icons and HTML source code must be compiled into object code using the resource compiler SRC where necessary (for the one essential item and the three FTN95-specific auxiliary items).

For example, this command issued from the `C:\simfit7\dll\clearwin` folder

```
src ico_clr
```

will use the script file `ico_clr.rc` to compile the `*.ico`, `*.htm`, and `*.jpg` files into an object file for loading into `w_clearwin.dll`.

6.1.3 Compiling the source code

It may be advisable to edit the format statement in `w_config.for` to upgrade defaults for the SIMFIT auxiliaries, or even alter this code to specify completely new defaults. After that, this command issued from the `C:\simfit7\dll\clearwin` folder

```
ftn95 *.for
```

will create *.obj files from all the *.for files in that local folder. Note that batch files `f.bat` are provided where compiler directives can be added if required to override the defaults placed by the command

```
ftn95 /config
```

into the file `ftn95.cfg`. In that case, the simple command

```
f *
```

can be used to create the `*.obj` files.

6.1.4 Linking the object code

This uses the Silverfrost linker SLINK.

To illustrate, if you issue the command

```
slink clearwin.link
```

from within `C:\simfit7\dll\clearwin`, then SLINK will use the link script `clearwin.link` to create `w_clearwin.dll`. A batch file `makeclr.bat` is provided to create `w_clearwin.dll`, and this can be edited to include the compilation phase as well if required.

You should not try to build the SimFIT or SimDEM packages using the Plato IDE, as it is infinitely better to use the batch and link files supplied with the source code to do this.

6.2 Example 2: NAGfor and w_menus.dll

As an example, the complete procedure for using NAGfor to create `w_menus.dll` will be described.

NAGfor creates intermediate C code that is passed to the `gcc` compiler for creating object code `*.o`, and also for linking. The `gcc` auxiliary program `windres` can be used to compile resources, and the `-compatible` compiler switch (formerly `-f77`) creates code according to the stdcall convention.

6.2.1 Compiling the source code

For instance, the command

```
nagfor -compatible -c *.for
```

issued from within `C:\simfit7\dll\menus` will create *.o files from all the *.for files in that folder.

6.2.2 Linking the object code

This uses NAGfor to pass link instructions on to `gcc`, and it will only work if there is an existing copy of `C:\simfit7\dll\clearwin\w_clearwin.dll`. This is only needed so the export table can be scanned to satisfy all the references.

For example, the command

```
nagfor @nagfor_menus.link
```

will create `w_menus.dll` using the link script `nagfor_menus.link`.

You should not try to build the SimFIT or SimDEM packages using the NAG Fortran Builder IDE, as it is infinitely better to use the batch and link files supplied with the source code to do this.

7 SimDEM GUI

This consists of three DLLs.

```
w_clearwin.dll (or x64_clearwin.dll)
w_menus.dll (or x64_menus.dll)
w_graphics.dll (or x64_graphics.dll)
```

The silverfrost release versions from 7.4.2 only use the dll `\simdem32.dll` in 32-bit applications or `simdem64.dll` in 64-bit applications.

7.1 w_clearwin.dll

This must be compiled and linked using Silverfrost FTN95.

Do not use `/f_stdcall` for the standard Silverfrost version.
Use `/f_stdcall` for the NAG version.

Procedure A.

```
Change to C:\simfit7\dll\clearwin
Type src ico_clr to compile the HTML code
Type scc *.cpp to compile C codes
Type f w_editor to create the module rp_editor_module
Type f module_clearwin to create the module module_clearwin
Type f * to cause the f.bat program to compile the object code
Type makeclr to activate makeclr.bat
```

7.2 w_menus.dll

Procedure B.

```
Change to C:\simfit7\dll\menus
Type f * to cause the f.bat program to compile the object code
Type makemen to activate makemen.bat
```

The linker SLINK will report unsatisfied references if it cannot find `C:\simfit7\dll\w_clearwin.dll`.

7.3 w_graphics.dll

Procedure C.

```
Change to C:\simfit7\dll\graphics
Type f module_savegks to compile the module_savegks
Type f * to cause the f.bat program to compile the object code
Type makegra to activate makegra.bat and
link to w_clearwin.dll
```

The linker SLINK will report unsatisfied references if it cannot find `C:\simfit7\dll\w_clearwin.dll`

Repeat procedures A, B, and C (if SLINK reports unresolved references) until `w_clearwin.dll` and `w_graphics.dll` and `w_menus.dll` are all consistent.

8 SIMDEM executables

This is done in `C:\simdem` and requires local copies of `w_clearwin.dll`, `w_menus.dll`, and `w_graphics.dll`.

- To make the standard non `/f_stdcall` Silverfrost version

```
Use ftn95 /config to make sure /f_stdcall is switched off
Type make_SILVERFROST_simdem to activate make_SILVERFROST_simdem.ba
```

- To make the `/f_stdcall` Silverfrost version

```
Use ftn95 /config to make sure /f_stdcall is switched on
Type make_SILVERFROST_simdem to activate make_SILVERFROST_simdem.ba
```

- To make the NAGfor -compatible version

```
Type make_NAG_simdem to activate make_NAG_simdem.bat
```

9 FTN95 auxiliary items

For SIMFIT only, not SIMDEM you must first edit then compile `dllchk.for`.

The two auxiliary items are

1. The driver `run6.exe = w_simfit.exe`, and
2. `change_simfit_version.exe`.

If Silverfrost FTN95 is not going to be used then it would be easier to build a new `w_simfit.exe` driver from scratch.

9.1 w_simfit.exe

Change to `C:\simfit7\work`
Type `getdll` to make local copies of the SIMFIT DLLs available
Type `f run6` to activate `f.bat` to create `run6.obj`
Type `slink run6.link` to create `run6.exe`
Type copy `run6.exe` to `w_simfit.exe` to create the SIMFIT driver

9.2 change_simfit_version.exe

Change to `C:\simfit7\work`
Type `f change_simfit_version` then `slink change_simfit_version.link`

10 Numerical analysis

The files concerned are

`w_maths.dll and`
`w_numbers.dll`

but there are several variants due to the fact that there are academic versions as well as NAG versions.

This is how the system works.

- Every installation of SIMFIT requires `w_maths.dll` and `w_numbers.dll`
- This pair must be consistent in any installation
- The only difference between versions of SIMFIT is in the pair of DLLs that are linked in
- In all versions: `w_numbers.dll` is completely free standing and includes BLAS and LAPACK
  SIMFIT is dependent on this `w_numbers.dll`
- In the Academic version `w_maths.dll` is linked to `w_numbers.dll`
- Instead, in the NAG versions `w_maths.dll` is linked to the NAG DLLs.

This is how to prepare the DLLs

10.1 w_numbers.dll

```
Change to C:\simfit7\dll\numbers and type compile to activate
compile.bat then makenum to make w_numbers.dll
```

10.2 w_maths.dll

```
Change to C:\simfit7\dll\maths and type f* to activate f.bat,
then type makemat to make w_maths.dll and academic_maths.dll
Change to C:\simfit7\dll\nag and type make_all_nag to make the
NAG library linked versions. It will be necessary to study
and possibly edit make_all_nag.bat and the link files it calls.
It may be necessary to edit change_simfit_version.config if links to
the NAG library DLLs are required.
```

11 w_models.dll

Change to `C:\simfit7\dll\models`

```
Type f * to activate f.bat
Type makemod to activate makemod.bat
```

12 w_simfit.dll

Change to `C:\simfit7\dll\simfit`

```
Type f * to activate f.bat
Type makesim to activate makesim.bat
```

13 SIMFIT executables

Change to `C:\simfit7\work`

```
Type f * to activate f.bat
Type linkall to activate linkall.bat
Type makew to activate makew.bat
```

14 NAG library details

It should be noted that some of the information in this section refers to NAG routines that are no longer extant, because they have been deleted from the library. For example, `j06sbf` was in the obsolete NAG graphics library. However most of the functionality that was available in the former NAG graphics library is still available using the SIMFIT graphics procedures. Again, the old G05 routines for random number generators, and some other obsolete routines, are still referenced due to their extremely widespread use in SIMFIT but what happens in such cases is that there is extra code to call the newer replacement routines. When NAG routines are called, users can interactively edit all the control parameters described in the NAG documentation, but in some cases the SIMFIT routines have extra functionality and can call the routines with additional parameters, which is done by planting code that is activated when additional arguments are required.

14.1 NAG data files and models

The following SIMFIT test files are data sets and model equations taken from the NAG documentation that are used in SIMFIT to demonstrate the NAG library routines. These files are all available after using the[NAG] button of the SIMFIT files Open control, but in most cases they are presented as defaults anyway when the routine is called. The list of files is maintained in the file `list.nag`, and all that is required to add further files is to edit `list.nag` and place the new files in the SIMFIT file store, as `list.nag` is scanned for this list each time the [NAG] button is activated.

Models

c05adf.mod	1 function of 1 variable
c05nbf.mod	9 functions of 9 variables
d01ajf.mod	1 function of 1 variable
d01eaf.mod	1 function of 4 variables
d01fcf.mod	1 function of 4 variables
e04fyf.mod	1 function of 3 variables

Data

c02agf.tf1	Zeros of a polynomial
e02adf.tf1	Polynomial data
e02baf.tf1	Data for fixed knot spline fitting
e02baf.tf2	Spline knots and coefficients
e02bef.tf1	Data for automatic knot spline fitting
e04fyf.tf1	Data for curve fitting using e04fyf.mod
f01abf.tf1	Inverse: symposdef matrix
f02fdf.tf1	A for Ax = (lambda)Bx
f02fdf.tf2	B for Ax = (lambda)Bx
f02wef.tf1	Singular value decomposition
f02wef.tf2	Singular value decomposition
f03aaf.tf1	Determinant by LU
f03aef.tf1	Determinant by Cholesky
f07fdf.tf1	Cholesky factorisation
f08kff.tf1	Singular value decomposition
f08kff.tf2	Singular value decomposition
g02baf.tf1	Correlation: Pearson
g02bnf.tf1	Correlation: Kendall/Spearman
g02bny.tf1	Partial correlation matrix
g02daf.tf1	Multiple linear regression
g02gaf.tf1	GLM normal errors
g02gbf.tf1	GLM binomial errors
g02gcf.tf1	GLM Poisson errors
g02gdf.tf1	GLM gamma errors
g02haf.tf1	Robust regression (M-estimates)
g02laf.tf1	Partial Least squares X-predictor data
g02laf.tf2	Partial Least Squares Y-response data
g02laf.tf3	Partial Least Squares Z-predictor data
g02wef.tf1	Singular value decomposition
g02wef.tf2	Singular value decomposition
g03aaf.tf1	Principal components
g03acf.tf1	Canonical variates
g03adf.tf1	Canonical correlation
g03baf.tf1	Matrix for Orthomax/Varimax rotation
g03bcf.tf1	X-matrix for procrustes analysis
g03bcf.tf2	Y-matrix for procrustes analysis
g03caf.tf1	Correlation matrix for factor analysis
g03ccf.tf1	Correlation matrix for factor analysis
g03daf.tf1	Discriminant analysis
g03dbf.tf1	Discriminant analysis
g03dcf.tf1	Discriminant analysis
g03eaf.tf1	Data for distance matrix: calculation
g03ecf.tf1	Data for distance matrix: clustering
g03eff.tf1	K-means clustering
g03eff.tf2	K-means clustering
g03faf.tf1	Distance matrix for classical metric scaling

g03ehf.tf1	Data for distance matrix: dendrogram plot
g03ejf.tf1	Data for distance matrix: cluster indicators
g04adf.tf1	ANOVA
g04aef.tfl	ANOVA library file
g04caf.tf1	ANOVA (factorial)
g07bef.tf1	Weibull fitting
g08aef.tf1	ANOVA (Friedman)
g08aff.tfl	ANOVA (Kruskall-Wallis)
g08agf.tf1	Wilcoxon signed ranks test
g08agf.tf2	Wilcoxon signed ranks test
g08ahf.tf1	Mann-Whitney U test
g08ahf.tf2	Mann-Whitney U test
g08cbf.tf1	Kolmogorov-Smirnov 1-sample test
g08daf.tf1	Kendall coefficient of concordance
g08raf.tf1	Regression on ranks
g08rbf.tf1	Regression on ranks
g10abf.tf1	Data for cross validation spline fitting
g11caf.tf1	Stratified logistic regression
g12aaf.tf1	Survival analysis
g12aaf.tf2	Survival analysis
g12baf.tf1	Cox regression
g13dmf.tf1	Auto- and cross-correlation matrices
j06sbf.tf1	Time series

14.2 NAG procedures

- a00acf, a00adf
- c02agf
- c05adf, c05azf, c05nbf
- d01ajf, d01eaf
- d02cjf, d02ejf
- e02adf, e02akf, e02baf, e02bbf, e02bcf, e02bdf, e02bef, e02gbf, e02gcf
- e04jyf, e04kzf, e04uef, e04uff
- f01abf, f01acf, f01adf
- f02aaf, f02aff, f02ebf, f02fdf
- f03aaf, f03abf, f03aef, f03aff
- f04aff, f04agf, f04ajf, f04asf, f04atf
- f06eaf, f06ejf, f06qff, f06yaf, f06raf
- f07adf, f07aef, f07agf, f07ajf, f07fdf

- f08aef, f08aff, f08faf, f08kaf, f08kef, f08kff, f08mef, f08naf, f08saf
- fz1caf, fz1clf
- g01aff, g01bjf, g01bkf, g01cef, g01dbf, g01ddf, g01eaf, g01ebf, g01ecf, g01edf, g01eef, g01eff, g01emf, g01faf, g01fbf, g01fcf, g01fdf, g01fef, g01fff, g01fmf, g01gbf, g01gcf, g01gdf, g01gef
- g02baf, g02bnf, g02byf, g02caf, g02gaf, g02gbf, g02gcf, g02gdf, g02gkf, g02haf, g02laf, g02lcf, g02ldf
- g03aaf, g03acf, g03adf, g03baf, g03bcf, g03caf, g03ccf, g03daf, g03dbf, g03dcf, g03eaf, g03ecf, g03eff, g03ejf, g03faf, g03fcf
- g04adf, g04aef, g04agf, g04caf
- g05cbf, g05ccf, g05daf, g05dbf, g05dcf, g05ddf, g05def, g05dff, g05dhf, g05dpf, g05dyf, g05ecf, g05edf, g05ehf, g05eyf, g05fff, g05kff, g05kgf, g05ncf, g05saf, g05scf, g05sdf, g05sff, g05sjf, g05skf, g05slf, g05smf, g05snf, g05sqf, g05ssf, g05taf, g05tdf, g05tjf, g05tlf
- g07aaf, g07abf, g07bef, g07daf, g07ddf, g07eaf, g07ebf
- g08aaf, g08aef, g08acf, g08aff, g08agf, g08ahf, g08ajf, g08akf, g08baf, g08cbf, g08cdf, g08daf, g08eaf, g08raf, g08rbf
- g10abf, g10acf, g10baf, g10zaf
- g11caf
- g12aaf, g12baf, g12zaf
- g13aaf, g13abf, g13acf, g13adf, g13aef, g13ahf
- s01baf
- s11aaf, s11abf, s11acf
- s13aaf, s13acf, s13adf
- s14aaf, s14abf, s14acf, s14adf, s14baf
- s15abf, s15acf, s15adf, s15aef, s15aff
- s17acf, s17adf, s17aef, s17aff, s17agf, s17ahf, s17ajf, s17akf
- s18acf, s18adf, s18aef, s18aff
- s19aaf, s19abf, s19acf, s19adf
- s20acf, s20adf
- s21baf, s21bbf, s21bcf, s21bdf, s21caf
- x01aaf, x02ajf, x02alf, x02amf, x03aaf

14.3 NAG DLL interface

In order for SIMFIT to run with any version of the NAG library, and to have additional functionality, like extra arguments, or calling obsolete routines, the named procedures just listed are not called directly from SIMFIT. What happens is that there is a set of dummy procedures with exactly the same argument lists as required by the NAG library, but they all have an additional dollar sign at the end of the named procedure. Inside the source code of such dummy procedures is a call to SIMFIT subroutine `putifa` so SIMFIT will always run with `IFAIL = -1`, but then write out NAG messages for nonzero `IFAIL` values, or results from iterative procedures, to a file called `nagifail.txt`. Some dummy procedures, of course, will also have the code for extra functionality referred to previously.

As an example, consider the subroutine `D01AJF` for quadrature. This would be accessed by a call as follows

```
CALL D01AJF$(F, A, B, EPSABS, EPSREL, RESUL, ABSERR, W, LW,
     +       IW, LIW, IFAIL)
```

but this would be included in a version of `w_maths.dll` which linked in to the object code from compiling the subroutine `D01AJF$.F` coded as follows.

```
C
C
      SUBROUTINE D01AJF$(F, A, B, EPSABS, EPSREL, RESUL, ABSERR, W, LW,
     +                   IW, LIW, IFAIL)
C
      IMPLICIT   NONE
      INTEGER    IFAIL, LIW, LW, IW(LIW)
      DOUBLE PRECISION F, A, B, EPSABS, EPSREL, RESUL, ABSERR, W(LW)
      EXTERNAL   D01AJF, F, GETIFA
      CALL GETIFA (IFAIL)
      CALL D01AJF (F, A, B, EPSABS, EPSREL, RESUL, ABSERR, W, LW,
     +             IW, LIW, IFAIL)
      END
C
C
```

This mode of operation has several very considerable advantages.

- It is a trivial matter to update SIMFIT to use future versions of the NAG library, without having to change the SIMFIT source code.
- It is simple to shunt calls to obsolete routines into calls to newer procedures without needing to change the source code.
- The behavior of the NAG `IFAIL` mechanism can be changed by a one line edit.
- It is easy to create modules to run from within the SIMFIT environment that could link directly to the NAG DLLs, and so bypass the SIMFIT dollar sign mechanism if required.

It should be indicated that any executable made using the NAG Fortran Builder that is linked in to the SIMDEM GUI and calls the NAG library DLLs can be used as a module from within the SIMFIT environment.

14.4 NAG library updates

The only difference between alternative versions of SIMFIT is the file `w_maths.dll`. This is either linked to the SIMFIT numerical libraries, or one of the NAG library DLLs. The usual procedure would be to make a SIMFIT DLL stub, so that SIMFIT can be used with a new version of a NAG DLL that is not covered by the current SIMFIT distribution. This stub is then used by **change_simfit_version.exe** to overwrite the current version of `w_maths.dll` so that SIMFIT links to the NAG library.

The recommended procedure is first summarized, details are given, then a worked example is provided.

- Download and unzip `nagzip***.zip` from `www.simfit.org.uk`.
- Study a typical batch file such as `makenag_markxy.bat` which is for Mark xy.
- Make a copy of this file that just adds the new NAG DLLs to the SIMFIT repertoire.
- It may be necessary to edit a couple of other files referenced by this batch file as described below
- Run `makenag_markxy.bat` to create the new SIMFIT DLL linked to the NAG Mark xy DLL
- Add this new SIMFIT DLL to the SIMFIT distribution

The following details give a description of exactly what to do to to take an existing compiled version of SIMFIT and make it link to a new version of the NAG DLLs.

It will be assumed that the Silverfrost-Salford FTN95 or NAG NAGfor compiler is going to be used and that the SIMFIT code has been unzipped into the folder `c:\simfit7\dll\nag` using the zip file `nagzip***.zip` distributed with the SIMFIT package. Once a certain amount of limited coding has been completed it is then only necessary to run the batch file `makenag_markxy.bat`, which compiles and links everything. To use different paths or alternative compilers a certain amount of extra editing would be necessary. In order to perform the upgrade it will be necessary to look at the file system defined in the next section, identify the extremely simple codes that are needed, act accordingly, then simply type

```
makenag_markx
```

to use FTN95 or, if NAGfor is to be used, type

```
makenag_markxy_nagfor
```

to create the upgrade to the NAG library at Mark xy.

Files needed to build the NAG DLL interface

1. **Link scripts for the compiler**

   The files below are completed and only need to be edited if the paths to the NAG library DLLs have been changed.
   One file is needed for each DLL to be created.

```
nag_mark20.link
mkl_mark21a.link
mkl_mark21z.link
mkl_mark22m.link
mkl_mark23m.link
mkl_mark24m.link
mkl_mark25m.link
nag_mark21a.link
nag_mark21z.link
nag_mark22m.link
nag_mark23m.link
nag_mark24m.link
nag_mark25m.link
x64_mkl_mark24.link
x64_nag_mark24.link
x64_mkl_mark25.link
x64_nag_mark25.link
```

2. **The DLLs to be created**

   All of these DLL stubs can be created at each new release if required, which can be done by the makefiles `makenag_xy.bat` files. However, this requires archived copies of all previous DLLs and should not normally be used. It would be usual to make an edited copy of e.g. `makenag_23m.bat` to only create just one new version.

```
fldll20_maths.dll
fldll214a_mkl_maths.dll
fldll214z_mkl_maths.dll
fldll224m_mkl_maths.dll
fldll234m_mkl_maths.dll
fldll244m_mkl_maths.dll
fldll254m_mkl_maths.dll
fldll214a_nag_maths.dll
fldll214z_nag_maths.dll
fldll224m_nag_maths.dll
fldll234m_nag_maths.dll
fldll244m_nag_maths.dll
fldll254m_nag_maths.dll
FLW6I24DC_mkl_maths.dll
FLW6I25DC_mkl_maths.dll
FLW6I24DC_nag_maths.dll
FLW6I25DC_nag_maths.dll
```

3. **The makefile**

   This is, for example, `makenag_mark23.bat` which does the following:

   ```
   a. Compile using FTN95
   b. Link
   c. Create the DLLs
   ```

   Browsing `makenag_mark23.bat`, for example, will make all the above perfectly clear. It is only possible to make a DLL if the path to the NAG DLL in the link script points to an existing NAG DLL.

4. **Other action required**

   Edit `change_simfit_version.config` and make sure this file, and the file `change_simfit_version.exe`, and the dummy DLLs described above are distributed with the package.

   Note that no action is required that involves the rest of the SIMFIT package. All that is needed to upgrade the SIMFIT package to use a new version of a NAG DLL is to make sure that the SIMFIT binary folder contains a copy of the new SIMFIT DLL linked to the new NAG DLL, and that the edited version of `change_simfit_version.config` has been used to overwrite the existing file `w_maths.dll`.

14.5 Example: Upgrading from Mark 22 to Mark23

This example should be imitated so that SIMFIT can be made link to future releases of the NAG library DLLs. It is important to note that any compiler can be used, not just FTN95 or NAGfor, and SIMFIT can be used with any version of the NAG library without any recompilation of the SIMFIT code: all that is required is simple editing of some text files and the creation of a new stub linking SIMFIT to the new NAG DLLs.

At Mark 23 some of the routines used by SIMFIT from the F02 and G05 chapters were deleted. Now it would be extremely difficult to edit the SIMFIT code every time a routine is deleted. Instead, SIMFIT uses a dummy name so that the code can be called from the Academic maths library or any past, present, or future release of the NAG library. To understand how this is done please inspect the following files:

`f02_mark23.f`

for the F02 update and the file

`g05_mark23.f`

for the G05 update. Such a large redirection is not usually required, but was necessary at Mark 23 because some LAPACK routines had been omitted at Mark 22 and a wholesale upgrade to the random number generators was made available.

The steps required were as follows.

1. Copy `mkl_mark22m.link` to `mkl_mark23m.link` then edit.

2. Copy `nag_mark22m.link` to `nag_mark23m.link` then edit.
3. Copy `makenag_mark22.bat` to `makenag_mark23.bat` then edit.
4. Type `makenag_mark23` to create the new DLL stubs.
5. Check that the following new DLLs have been created
   `fldll234m_mkl_maths.dll` and
   `fldll234m_nag_maths.dll.`
6. Edit `change_simfit_version.config` to reference the Mark 23 DLLs.
7. Add the following files to the SIMFIT program folder
   `change_simfit_version.config`
   `fldll234m_mkl_maths.dll` and
   `fldll234m_nag_maths.dll.`
8. As administrator, run the executable
   `change_simfit_version.exe` in the SIMFIT folder.

14.6 Example: Upgrading from Mark 23 to Mark24

This is particularly easy as there were no routines used by SIMFIT that became obsolete. Here is an abbreviated form of `makenag_mark24.bat` which creates the dummy DLLs.

```
echo Step 1: Compile all the *.f source code
ftn95 /f_stdcall getifa_ftn95.f95
ftn95 /f_stdcall *.f

echo Step 2: Create the new nag dll linked to the nag mark24m NAG DLL
slink nag_mark24m.link

echo Step 3: Create the new mkl dll linked to the mkl mark24m NAG DLL
slink mkl_mark24m.link
```

The corresponding 64-bit batch file is `x64_makenag_mark24.bat`.

14.7 Example: Upgrading from Mark 24 to Mark25

This is fairly easy but there were some routines used by SIMFIT that became obsolete. Here is an abbreviated form of `makenag_mark25.bat` which creates the dummy DLLs.

```
echo Step 1: Compile all the *.f source code
ftn95 /f_stdcall getifa_ftn95.f95
ftn95 /f_stdcall *.f

echo Step 2: Create the new nag dll linked to the nag mark25m NAG DLL
slink nag_mark25m.link

echo Step 3: Create the new mkl dll linked to the mkl mark25m NAG DLL
slink mkl_mark25m.link
```

The corresponding 64-bit batch file is `x64_makenag_mark25.bat`.

14.8 Compiling the NAG library source codes

This section adds additional information to the previous section on numerical analysis (page 20) so that users can appreciate how to compile selected routines instead of the whole NAG library replacement DLLs. The naglib zip files unzip into a `maths` folder containing the source codes for the NAG routines, and a `numbers` folder with subfolders containing auxiliary routines. A list of public domain software and acknowledgement of the programmers involved will be found in the SimFIT reference manual `w_manual.pdf`.

The source codes used to replace some 215 library routines called by SimFIT are a mixture of public domain subroutines, some edited to conform to the NAG library calling sequences, but with some subroutines created from scratch. This code only contains standard Fortran constructs and can be compiled using any Fortran compiler. Nevertheless, several things should be noted.

1. Some of the subroutines in the `maths` folders are dummy stubs for subroutines that are called by the NAG version of SimFIT but are not called by the academic version of SimFIT and they just return `IFAIL = -399`. Also many of the routines in the `numbers` subfolders are not called by the NAG library routines but are called from elsewhere in SimFIT so, to avoid compiling the whole of the maths and numbers subroutines and just compile a particular NAG routine, it will be necessary to check for dependencies within the `numbers` subfolders and simply extract the code required.

2. The routines treat `IFAIL` as an intent (out) variable that is zeroized on entry to the routines. So the input `IFAIL` value is not used. However, as far as possible, the exit `IFAIL` values correspond to the NAG documentation, but the error trapping must be done by users supplying their own checking code for nonzero `IFAIL` exits, as I have done in the SimFIT package.

3. The routines have exactly the same names as the NAG ones except for an added dollar character to the routine name. However the arguments are exactly the same.

4. Some of the routines use the workspaces dimensioned as for the NAG routines but some use additional workspaces, mostly created as temporary workspaces using allocate.

5. Some routines are as good, or even better, than the NAG routines, but some were thrown together in a hurry and are not so polished. I never got round to optimizing some code, particularly searching, sorting, selecting between accuracy and speed, avoiding repetition, or economizing on storage, and this is often indicated in the comments.

6. Users may wish to use their own implementations of packages like BLAS, LAPACK, and SLATEC.

7. The codes are nearly all in fixed format `*.for` style and, if free format `*.f95` code is preferred, you should use my SimFIT program **for2f95**, as this is designed to maintain the readability built into the original code that will be destroyed by general purpose fixed to free translators.

15 Manual

Translating or extending the manuals will be very easy, since a very strict LaTeX style has been used. Programmers will observe that at one or two points handcrafting has been used (e.g.`\newpage`), and this will have to be edited. Note also that most of the diagrams are included as `*.wgb` files. The file `prolog.wgb` contains the PostScript header that has been cut out of the individual PostScript files to save space. By pasting `prolog.wgb` back into the `*.wgb` files they become `*.eps` files. Of course `dvips` only needs `prolog.wgb` once as a special. Note that `makeindex` is required to create the index. As `hyperef` is used, a call to `dvips` then `ps2pdf` converts the `*.dvi` file into `*.ps` and `*.pdf` with hyperlinks. By obvious editing in `w_manual.tex`, as in `mono_manual.tex`, a monochrome manual can be produced. Usually the package is distributed with `w_manual.pdf` in color with hyperlinks, but `mono_manual.pdf`, and `w_manual.ps` in monochrome for high resolution monochrome printing.

Programmers should definitely use the default folders otherwise it will be necessary to edit every call to included graphics files throughout the whole document.

```
C:\manuals                  ...LaTeX w_manual       [1st pass]
                               LaTeX w_manual       [2nd pass]
                               Makeindex w_manual   [1st pass]
                               LaTeX w_manual       [3rd pass]
                              (Makeindex w_manual)  [2nd pass ?]
                              (LaTeX w_manual)      [4th pass ?]
                               dvips w_manual       [w_manual.ps]
                               ps2pdf w_manual      [w_manual.pdf]
C:\manuals\promote             LaTeX promote
                               dvips promote        [promote.ps]
                               ps2pdf promote.ps    [promote.pdf]
C:\manuals\ms_office           LaTeX ms_office
                               dvips ms_office      [ms_office.ps]
                               ps2pdf ms_office.ps  [ms_office.pdf]
C:\manuals\pscodes             LaTeX pscodes
                               dvips pscodes        [pscodes.ps]
                               ps2pdf pscodes.ps    [pscodes.pdf]
C:\manuals\source              LaTeX source
                               dvips source         [source.ps]
                               ps2pdf source.ps     [source.eps]
```

16 Distribution

Before making a distribution a package must be compiled, but it will be necessary to refresh the binaries. For instance, binaries to build SimFIT are stored in `c:\setup\programs` and the batch files `update.bat`, and `x64_update.bat` should be run to make sure that only the recently compiled binaries are loaded into the distribution executable.

To make the SIMFIT self-extracting installation programs, use edited versions of the scripts `simfit.iss`, and `x64_simfit.iss`, together with text files `infobefo.txt` and `x64_infobefo.txt` for Inno Setup from

```
http://www.jordanr.cjb.net/
or
http://www.jordanr.dhs.org/.
```

In the case of SIMDEM the files are `simdem.iss`, `x64_simdem.iss`, `demobefo.txt`, and `x64_demobefo.txt`.

However, by editing the information files `infobefo.txt` and `demobefo.txt` if required, and analyzing the compilation scripts `simfit.iss` and `simdem.iss` to appreciate what paths are involved, any program can be used to distribute the packages.

17 Makefiles

It is important to note that if frequent changes of compiler are made then modules can become inconsistent. For this reason the object code generated for the SIMFIT package program files and the GUI DLLs
`w_simfit.dll`
`w_graphics.dll`
should be compiled twice in succession to make sure the correct modules are linked in.

The procedure with dedicated FTN95 scripts is described for SIMFIT while for SIMDEM using NAGfor is also illustrated with dedicated NAGfor commands. Check that all the batch files and link scripts have correct paths and that all subfolders exist and contain the necessary files. Also, make sure all signatures are updated and that SRC has been used to create objects from the icon `*.ico` and `*.rc` files then proceed as follows.

For FTN95 and the SIMFIT package the sequence of commands is:

```
ftn95 /config
cd c:\simfit7\dll\numbers
compile
makenum
cd c:\simfit7\dll\maths
f *
makemat
cd c:\simfit7\dll\clearwin
src ico_clr
scc scroll_kludge
f w_editor
f module_clearwin
f *
makeclr
cd c:\simfit7\dll\menus
```

```
f *
makemen
cd c:\simfit7\dll\graphics
f module_savegks
f*
makegra
cd c:\simfit7\dll\models
f *
makemod
cd c:\simfit7\dll\simfit
f orthog
f *
makesim
cd c:\simfit7\dll\help
makehlp
cd c:\simfit7\dll\nag
make_all_nag
cd c:\simfit7\work
getdll
src ico_sim6
src ico_run6
f *
linkall
makew
cd c:\setup\programs
update
cd ..
notepad infobefo.txt
```

Now run the Inno-setup compiler using `simfit.iss`, rename the `C:\setup\output\setup.exe` file appropriately and zip up.

For FTN95 and the SIMDEM package the sequence of commands is:

```
ftn95 /config
cd c:\simfit7\dll\clearwin
src ico_clr
scc scroll_kludge
f w_editor
f module_clearwin
f *
makeclr
cd c:\simfit7\dll\menus
f *
makemen
cd c:\simfit7\dll\graphics
f module_savegks
f *
```

```
makegra
cd c:\simdem
getdll
make_SILVERFROST_simdem
notepad demobefo.txt
cd c:\simdem\output
```

For NAGfor and the SIMDEM package the sequence of commands is to first use FTN95 as follows:

```
ftn95 /config
cd c:\simfit7\dll\clearwin
src ico_clr
scc scroll_kludge
f w_editor
f module_clearwin
f *
makeclr
```

which creates `w_clearwin.dll`. Then use

```
cd c:\nagfor\dll\menus
nagfor -compatible -c -w=x77 -f2003 *.for
nagfor @nagfor_makemen.link
cd c:\nagfor\dll\graphics
nagfor -compatible -c -w=x77 -f2003 module_savegks.for
nagfor -compatible -c -w=x77 -f2003 *.for
nagfor @nagfor_makegra.link
cd c:\simdem
get_nagdll
make_NAG_simdem
notepad demobefo.txt
cd c:\simdem\output
```

Now run the Inno-setup compiler using `simdem.iss`, rename the `c:\simdem\output\setup.exe` file appropriately and zip up. Single makefiles calling batch files can be used to compile and link these packages, but these may not be distributed with the source codes to avoid confusion. Following the above sequence of command lines should allow anybody to create their own makefiles.

Comments and requests for help to `bill.bardsley@simfit.org.uk`