



NAMMU

Release 7.2

Installation and Running Guide

Serco Reference: SA/ENV-0630

Date of Issue: December 2003

Title	NAMMU Release 7.2 Installation and Running Guide
Customer	
Customer reference	
Confidentiality, copyright and reproduction	
File reference	doc / NAMMU_Installation
Report number	SA/ENV-0630
Report status	Issue 1

Serco Assurance
150 Harwell IBC
Didcot
Oxfordshire OX11 0QJ
United Kingdom

Telephone: +44 (0) 1635 280415
Facsimile: +44 (0) 1635 280305
E-mail: gw.support@sercoassurance.com

Serco Assurance is a division of Serco Ltd.
Serco Assurance is certificated to BS EN ISO9001:(1994)

	Name	Signature	Date
Author	AR Hoch		5/12/03
Reviewed by	D Holton		5/12/03
Approved by	DA Lever		5/12/03

Preface

NAMMU is a software package for modelling groundwater flow and transport in porous media. The package can be used to model steady state and time-dependent behaviour, including unsaturated flow and the transport of mass and heat. An option is available for modelling radioactive decay and the transport of chains of radionuclides. The software is based on an efficient implementation of the finite-element method that provides many options for modelling complex geological regions.

The following documentation is available for Release 7.2 of NAMMU:

- NAMMU (Release 7.2) Technical Summary Document;
- NAMMU (Release 7.2) User Guide;
- NAMMU (Release 7.2) Command Reference Manual;
- NAMMU (Release 7.2) Verification Document;
- NAMMU (Release 7.2) Installation and Running Guide.

This document, the Installation and Running Guide, describes how to install and run NAMMU using its job submission program. It also describes how to run the test cases for the program.

COPYRIGHT AND OWNERSHIP OF NAMMU

The NAMMU program makes use of the TGSL subroutine library.
All rights to the TGSL subroutine library are owned by Serco Assurance.

All documents describing the NAMMU program and TGSL subroutine library are protected by copyright and should not be reproduced in whole, or in part, without the permission of Serco Assurance.

NAMMU also makes use of the freely available LAPACK linear algebra library.

Additional information about the capabilities and the potential applications of NAMMU is available on request from Serco Assurance.

Capabilities of NAMMU

NAMMU has a wide range of facilities for specifying a model region, the properties of rocks, fluids and solutes within the region, the equations to be solved, and the output options required. In addition to the standard facilities, many options are available that allow the user to customise the functionality of NAMMU for a particular project. The advanced 3D-visualisation package, GeoVisage, is available for NAMMU.

NAMMU can be used to model the following geometries and physics:

- Flow and transport in 3D Cartesian, 2D vertical and plan section, and 2D radial geometries;
- Deterministic and stochastic continuum modelling;
- Steady state and transient behaviour;
- Groundwater flow in saturated and unsaturated conditions;
- Saline groundwater flow with the density dependent on concentration;
- Coupled groundwater flow and heat transport with the density dependent on temperature;
- Saline groundwater flow and heat transport with the density dependent on concentration and temperature;
- Groundwater flow in a dual porosity system based on the Warren and Root model;
- Transport of contaminants, including the effects of advection, dispersion, and sorption, with solubility limitation;
- Transport of radioactive decay chains, allowing for interacting chains to be linked by solubility limitation of a common radionuclide.

NAMMU can be used to model the following features:

- Complex distributions of lithology;
- 3D volumes of enhanced or reduced permeability;
- Conductive or semi-impermeable fracture zones;
- Stochastic models of permeability and porosity;
- Boreholes, tunnels and shafts;
- Specified value (Dirichlet) and specified flux (Neumann) boundary conditions;
- Infiltration boundary conditions for surface recharge/discharge areas;
- Hydrostatic and outflow boundary conditions for vertical boundaries;
- Time-varying boundary conditions (e.g. used to model land uplift, or time-dependent contaminant discharge);
- Sources of contaminants, salinity or heat.

NAMMU models and results can be displayed by:

- A 3D visualisation system, GeoVisage for Nammu, that allows 3D rendering of finite-elements, rock types, permeability, fracture zones, variables, flow vectors, and pathlines;
- 2D plot and numerical output that includes:
 - Plots of the finite-element mesh and its boundaries;
 - Plots of contours of a variable on a surface;
 - Plots of contours of a variable on a 2D slice;
 - Plots of velocity arrows, showing direction and magnitude of the groundwater flow;
 - Plots of pathlines either for steady state or for transient groundwater flows;
 - Plots of backward pathlines, showing the region of influence of a borehole;
 - Graphs of variables along a line;
 - Graphs of the evolution of variables at a point;
 - Graphs of data;
 - Integrals (e.g. flux of groundwater across a plane).

NAMMU models have been used in the following applications:

- Calculations in support of safety assessments for radioactive waste disposal programmes:
 - Regional groundwater flow;
 - Site investigation;
 - Pump test simulation;
 - Tracer test.
- Modelling for groundwater protection schemes:
 - Aquifer;
 - Saline intrusion.
- Modelling to design and evaluate remediation strategies;
 - Aquifer contamination;
 - Landfill site.

NAMMU is used in support of the radioactive waste disposal programmes of many countries throughout the world, both by nuclear regulators and by national disposal organisations, and by consultants working for those organisations.

NAMMU has been developed by Serco Assurance (formerly UKAEA) over a period of more than 15 years and has been verified extensively in international comparison exercises. It is developed under a rigorous quality system that conforms to the international standards ISO 9001 and TickIT.

Contents

1	INTRODUCTION.....	1
2	INSTALLATION.....	2
2.1	Platforms Supported	2
2.2	Reading the Tape.....	2
2.3	Compilation	3
2.4	Initialisation	3
3	RUNNING NAMMU.....	4
3.1	Introduction	4
3.2	Job Submission.....	4
3.3	Running the Test Cases	4
4	THE INPUT DATA FILE	5
4.1	Job Information	5
4.1.1	Space Allocation	6
4.1.2	Source Files	6
4.1.3	Compiled Files	7
4.1.4	Library Files.....	8
4.1.5	Inline FORTRAN	8
4.1.6	Input Data Files.....	8
4.1.7	Output Data Files	9
5	ADVANCED FEATURES.....	10
5.1	The <code>.namhosts</code> File.....	10
5.2	The <code>nammu</code> Command	11
5.3	Temporary Directories Used.....	11

1 INTRODUCTION

NAMMU is a software package for modelling groundwater flow and transport in porous media. The package can be used to model steady state and time-dependent behaviour, including unsaturated flow and the transport of heat and mass. An option is available for modelling radioactive decay and transport of chains of radionuclides. The software is based on an efficient implementation of the finite-element method that provides many options for modelling complex geological regions.

This document, the Installation and Running Guide, describes how to install and run Release 7.2 of NAMMU using its job submission program. It also describes how to run the test cases for the program.

2 INSTALLATION

This section describes how to install NAMMU. Usually you will have received a compiled version of the program. However, in certain circumstances you may have been supplied with source code and this section also includes a description of how to compile the code.

2.1 Platforms Supported

The following machine configurations are currently supported:

- Cygwin tools (a port of GNU tools for Windows NT and 9x);
- Linux with the GNU g77 compiler;
- Linux with the Portland Group pgf90 compiler;
- Linux with the Intel ifc compiler;
- IBM rs/6000 workstations running AIX version 4.1 with the IBM xlf compiler;
- Silicon Graphics workstations running IRIX version 6.5 with the Silicon Graphics f77 compiler;
- Sun workstations running SunOS version 5.6 with the Sun f77 compiler;
- Windows NT / XP with the Compaq Visual Fortran 6.1 compiler. (This platform is discussed no further in this document.)

2.2 Reading the Tape

Before you install NAMMU you should first create a directory to hold the software. We shall refer to the full pathname of this directory as `NAMMUSYS` (note that `NAMMUSYS` is not a UNIX system variable, just a shorthand notation used in this manual). For example you might want to keep the software in a subdirectory of your `/home` directory called `nammu`. In this case you would type:

```
cd /home
mkdir nammu
```

and `NAMMUSYS` would be `/home/nammu`.

Now change to the `NAMMUSYS` directory and read the tape:

```
cd /home/nammu
tar xvf /dev/rmt0
```

The name of your local tape drive may differ from `/dev/rmt0`, in which case you should enter the appropriate name.

After reading the tape, `NAMMUSYS` should contain five subdirectories (six if the source code has been supplied). The subdirectories are:

sub contains the code used by the installation procedure to produce the NAMMU job submission program `nammu`. It also contains the compilation scripts.

libnam	(if present) contains the binary libraries needed to run NAMMU.
testdata	contains the input files for the test cases.
testdata_out	contains examples of the output which should be produced by the test cases. This directory may be deleted after installation to save space.
doc	contains the HTML files for the latest version of the Command Reference Manual.
source	(if present) contains a number of subdirectories that contain the source code for NAMMU. The source is not used after installation, so this directory may then be deleted to save space.

2.3 Compilation

If you have not been supplied with source code go to section 2.4. If you have been supplied with source code and you do not have the same versions of the operating system or compiler as those listed in section 2.1 then the compiled libraries supplied may not work on your machine. If this is the case you should first compile the code. This is very straightforward. In the `NAMMUSYS` directory execute the following command:

```
sub/tgsl.compile_source machine_type nammu 7.2
```

where *machine_type* is the type of machine on which the program is to be compiled. The following machine types are valid:

- cygwin, Cygwin tools;
- linux-gnu, Linux with the GNU g77 compiler;
- linux-intel, Linux with the Intel ifc compiler;
- linux-pgi, Linux with the Portland Group pgf90 compiler;
- rs, an IBM rs/6000 workstation;
- sg-r8000 or sg-r10000, a Silicon Graphics workstation with either sg-r8000 or sg-r10000 processor;
- sun, a Sun workstation.

The above command compiles all the source code and makes the binary libraries needed to run the code.

2.4 Initialisation

The final step of the installation is to run the `nammu_install` script:

```
sub/nammu_install machine_type 7.2
```

where *machine_type* is the type of machine on which the program is to be installed. The machine types described in section 2.3 are valid.

This script produces the NAMMU job submission program, `nammu`, in the `NAMMUSYS/sub` directory. The installation is now complete.

3 RUNNING NAMMU

3.1 Introduction

In this section we describe how to run the program. We assume that the installation procedure has been completed successfully.

3.2 Job Submission

This version of NAMMU is run in batch mode. This is done using the `nammu` command in the `NAMMUSYS/sub` directory. It is a good idea to make a link to the file `NAMMUSYS/sub/nammu` from a directory that contains application programs and which users would have included in their default path. The usual directory would be `/usr/local/bin`. The following commands will achieve this:

```
su
```

Now enter the super-user password at the `Password:` prompt.

```
cd /usr/local/bin
ln -s NAMMUSYS/sub/nammu nammu
exit
```

If this is not done the user must enter the full pathname of the `nammu` command to run the program.

The `nammu` command builds and runs the NAMMU program. It makes a subdirectory `nammu/output` (if one does not already exist) in the user's home directory. This is the default destination for the program output file. It also creates a file called `.tgs1job` (if one does not already exist) in the user's home directory. This is used to give each job a unique name of the form `usernnnn` where `user` is the user's `user-id` and `nnnn` is a four digit number which is initially set to 1000 and is incremented by one each time the program is run. The user can override this job naming facility.

The syntax for the command is:

```
nammu [-j jobname] file
```

where *file* is the name of the input file relative to the current directory or else the full pathname of the input file, and the `-j` option can be used to replace the default job name by *jobname*.

By default, the output from job *jobname* is sent to the file *jobname* in the subdirectory `nammu/output` of the user's home directory. The user can override this, either by specifying an alternative directory for the output (see section 5.1), or by specifying an explicit pathname for the output file (see section 4.1.7).

3.3 Running the Test Cases

The directory `NAMMUSYS/testdata` contains input datasets for a number of test cases for the program. The test cases can be run individually using the `nammu` command described above, or the `nammu_run_test` command in `NAMMUSYS/testdata` can be used to run the complete set of test cases. The output produced can be compared with that in the directory `NAMMUSYS/testdata_out`.

4 THE INPUT DATA FILE

The current version of NAMMU runs in batch mode. The input to the program is provided in an input data file. This file contains the statements in the TGIN input language that control the execution of the program. For a description of this language and the options available see the NAMMU Command Reference Manual listed in the Preface. The input file may be prepared using any text editor.

In addition to the TGIN statements, the input data file contains information that is used by the job submission script `nammu`. This information specifies the size of workspace to be allocated, the names of various input and output data files and any FORTRAN subroutines which may be required by the program. The information for the job submission script must appear in the form of TGIN comment lines, that is each line must either be blank or else begin with a `/*` and end with a `*/`. There is an exception to this rule when using the `INLINE FORTRAN` option (see below). Any number of comment lines may precede the job information. These are ignored both by the job submission script and by NAMMU. General comments about the job are best put here.

4.1 Job Information

The job information forms the first part of the input dataset. Each line of information must be delimited by `/*` and `*/`, so that it will be treated by NAMMU as a comment. The job information takes the following form:

```

/* SPACE ALLOCATION                                */
/* INTEGER WORKSPACE   n1                          */
/* REAL WORKSPACE      n2                          */
/*
/* SOURCE FILES                                    */
/* filename option                                  */
/*
/* COMPILED FILES                                  */
/* filename                                        */
/*
/* LIBRARY FILES                                   */
/* filename                                        */
/*
/* INLINE FORTRAN
   SUBROUTINE XXX
   ...
   ...
   ...
   END
*/

/* INPUT DATA FILES                                */
/* file_identifier filename                        */
/*
/* OUTPUT DATA FILES                              */
/* file_identifier filename                        */
/*
/* END JOB INFORMATION                             */

```

The blank lines are optional and are included here for clarity. There are seven blocks of information. Each block begins with a line containing two or more keywords. Each block ends with a line containing the keywords for the next block or the line

```

/* END JOB INFORMATION                                */

```

which acts as a terminator for the job information section of the input data file. The seven blocks are as follows:

SPACE ALLOCATION	This block specifies the space requirements for the particular run of the program.
SOURCE FILES	This block specifies the names of files containing FORTRAN subroutines, if any, that are to be compiled and linked into the program.
COMPILED FILES	This block specifies the names of files containing compiled FORTRAN subroutines, if any, that are to be linked into the program.
LIBRARY FILES	This block specifies the names of archive libraries containing compiled FORTRAN subroutines, if any, that are to be linked into the program.
INLINE FORTRAN	This block contains FORTRAN code that is to be compiled and linked into the program.
INPUT DATA FILES	This block specifies the names of files from which data is to be read by the program.
OUTPUT DATA FILES	This block specifies the names of files to which data is to be written by the program.

In the following subsections we describe, in turn, the function of each of the seven blocks.

4.1.1 Space Allocation

The SPACE ALLOCATION statement should be followed by the two lines

```
/* INTEGER WORKSPACE  n1          */
/* REAL WORKSPACE     n2          */
```

if the default values are not required, where

n1 is an integer specifying the amount of integer workspace to be allocated for the run. The unit of space used is an integer word¹. The default value is 400,000.

n2 is an integer specifying the amount of real workspace to be allocated for the run. The unit of space used is a floating point word². The default value is 400,000.

The amount of space required by NAMMU is dependent on the size of model used and the solver options. If insufficient workspace is allocated the program will print a message to this effect. If the program runs successfully a message will be printed at the end of the run indicating how much of the workspace was not used.

4.1.2 Source Files

The SOURCE FILES statement may be followed by a number of lines each specifying a file containing FORTRAN source code that is to be compiled and included in the run. NAMMU provides the user with the option of supplying user-written subroutines, for example to specify a

¹ Integer words of length 4 bytes are used throughout.

² Floating-point words of length 8 bytes are used throughout.

non-standard material properties or different forms of output. The names of the files containing the FORTRAN source are supplied at this point¹. Each file is specified on a new line, in the form:

```
/* filename option                               */
```

where

filename is the name of the file containing the FORTRAN source. If the filename begins with a / then it is assumed to be the full pathname of the file. If it begins with any other character it is assumed to be the pathname of the file relative to the user's home directory.

option is an optional keyword that indicates how the file is to be compiled. It can take the following form:

OPTOFF, the file should be compiled without optimisation.

OPTON, the file should be compiled with optimisation.

DEBUG, the file should be compiled with the `-g` flag to enable debugging to be carried out on the file.

The default value of the *option* keyword is OPTOFF.

An alternative means of including user-written subroutines in a NAMMU run is to use the `INLINE FORTRAN` block (see section 4.1.5).

If the list of `SOURCE FILES` contains more than one version of a particular subroutine, then the last version of the subroutine is used. `INLINE FORTRAN` takes precedence over `SOURCE FILES`, which in turn take precedence over `COMPILED FILES` and `LIBRARY FILES`.

4.1.3 Compiled Files

The `COMPILED FILES` statement is followed by a number of lines specifying files containing object code (`.o` files) that are to be included in the run. Each file should contain the object code generated by compiling only one FORTRAN routine. The names of the files are supplied at this point. Each file is specified on a new line, in the form:

```
/* filename                                       */
```

where

filename is the name of the file containing the object code. If the filename begins with a / then it is assumed to be the full pathname of the file. If it begins with any other character it is assumed to be the pathname of the file relative to the user's home directory. The filename should end with a `.o`.

This option can be used to supply user-written code. The source code should be compiled by the user and stored as an object file (`.o` file). This avoids the need to re-compile the code each time NAMMU is run.

If the list of `COMPILED FILES` contains more than one version of a particular subroutine, then the last version of the subroutine is used. `INLINE FORTRAN` and `SOURCE FILES` take precedence over `COMPILED FILES`, which in turn take precedence over `LIBRARY FILES`.

¹ Double precision floating-point variables are used throughout.

4.1.4 Library Files

The `LIBRARY FILES` statement is followed by a number of lines specifying library files that have been created by the UNIX `ar` command (`.a` files). The individual members of the library are files containing the object code resulting from the compilation of a single FORTRAN routine. All the members of the library are included in the run of the program by the link editor and replace like named routines in the NAMMU libraries. The names of the files are supplied at this point. Each file is specified on a new line, in the form:

```
/* filename */
```

where

filename is the name of the file containing the library. If the filename begins with a `/` then it is assumed to be the full pathname of the file. If it begins with any other character it is assumed to be the pathname of the file relative to the user's home directory. The filename should end with a `.a`.

This option can be used to supply user-written code. The source code should be compiled by the user, one subroutine at a time, and stored in a library file (`.a` file), created with the `ar` command. This avoids the need to re-compile the code each time NAMMU is run.

If the list of `LIBRARY FILES` contains more than one version of a particular subroutine, then the last version of the subroutine is used. `INLINE FORTRAN`, `SOURCE FILES` and `COMPILED FILES` take precedence over `LIBRARY FILES`.

4.1.5 Inline FORTRAN

The `INLINE FORTRAN` block of the job information allows FORTRAN source to be included in the input data file¹. The source is compiled and included in the run. If no FORTRAN source is to be included this block should be omitted from the input data. The `INLINE FORTRAN` block begins with the line:

```
/* INLINE FORTRAN
```

Note that there is no `*/` at the end of this line. The following lines contain the FORTRAN source in the standard format (i.e. labels in columns 1-5, continuation markers in column 6 and statements in columns 7-72). Note that these lines should not contain the `/*` or `*/` symbols; the whole of the `INLINE FORTRAN` block is treated as a comment by NAMMU. The block ends with the following line:

```
*/
```

It does not matter where the `*/` appears on the line.

If the `INLINE FORTRAN` contains more than one version of a particular subroutine, then the last version of the subroutine is used. `INLINE FORTRAN` takes precedence over `SOURCE FILES`, `COMPILED FILES` and `LIBRARY FILES`.

4.1.6 Input Data Files

The `INPUT DATA FILES` statement is followed by a number of lines specifying files from which data is to be read by the program. Each file must be entered on a new line, in the form:

¹ Double precision floating-point variables are used throughout.

```
/* file_identifier filename */
```

where

file_identifier is an identifier for the input stream in the program. It can be either a symbolic name (see below) which is associated with a particular function in the program, or an integer between 1 and 99, in which case it is interpreted as a FORTRAN unit identifier. Default integer identifiers and their meanings are as follows:

GRIDIN, the file from which the finite-element model is to be read. The >> RESTORE MODEL command assumes that GRIDIN is connected to a file containing a valid finite-element model in the NAMMU format. This should have been produced by a previous run of the program (see the GRIDOUT entry in the OUTPUT DATA FILES section).

GFIN, the file from which the global freedoms are to be read. The >> RESTORE GLOBAL FREEDOMS command assumes that GFIN is connected to a file containing the global freedoms output from a previous run of the program (see the GFOUT entry in the OUTPUT DATA FILES section).

filename is the name of the file from which the data are to be read. If the filename begins with a / then it is assumed to be the full pathname of the file. If it begins with any other character it is assumed to be the pathname of the file relative to the user's home directory. A useful convention is to identify the type of file by its extension. For example model files could have the extension .mdl and solution files the extension .gfl.

4.1.7 Output Data Files

The OUTPUT DATA FILES statement is followed by a number of lines specifying files to which data is to be written by the program. Each file must be entered on a new line, in the form:

```
/* file_identifier filename */
```

where

file_identifier is an identifier for the output stream in the program. It can be either a symbolic name (see below) which is associated with a particular function in the program, or an integer between 1 and 99, in which case it is interpreted as a FORTRAN unit identifier. The standard output from the job is written to stream 6. Valid symbolic names and their meanings are as follows:

GRIDOUT, the file to which the finite-element model is to be written.

GFOUT, the file to which the global freedoms are to be written.

GRAPHICS, the file to which graphics output is to be written. This output is produced by the various PLOT commands in the input data file (see the NAMMU (Release 7.2) Command Reference Manual).

filename is the name of the file to which the data is to be written. If the filename begins with a / then it is assumed to be the full pathname of the file. If it begins with any other character it is assumed to be the pathname of the file relative to the user's home directory.

5 ADVANCED FEATURES

This section describes, in a little more detail, how NAMMU is run on a UNIX machine. It also describes how to run the program across a network of UNIX workstations.

5.1 The `.namhosts` File

When the `nammu` script is run it needs the following information:

- The name of the machine the program is to run on;
- The type of machine the program is to run on;
- The name of the directory where the compiled libraries for NAMMU are kept;
- The name of a directory that NAMMU can use for scratch storage;
- The name of the directory to which the program output will be written.

The `nammu` script gets this information from a file called `.namhosts` in the `NAMMUSYS` directory. This file contains the above information for each machine in the network on which the user might want to run NAMMU. Each machine has an entry in the file consisting of five lines as follows:

- The name of the machine on which the program is to be run.
- The type of machine the program is to run on. Machine types supported at present are:
 - `cygwin`, Cygwin tools;
 - `linux-gnu`, Linux with the GNU `g77` compiler;
 - `linux-pgi`, Linux with the Portland Group `pgf90` compiler;
 - `linux-intel`, Linux with the Intel `ifc` compiler;
 - `rs`, an IBM `rs/6000` workstation;
 - `sg-r8000` or `sg-r10000`, a Silicon Graphics workstation with either `sg-r8000` or `sg-r10000` processor;
 - `sun`, a Sun workstation;
 - `win`, a Windows platform.
- The name of the directory containing the NAMMU compiled libraries for the type of machine on which the program is to be run. The name should be in the form `hostname:path` unless the compiled libraries are local to the machine on which the program is to be run, in which case the full local pathname can be used.
- The name of a directory on the machine on which the program is to be run, which can be used for scratch storage by NAMMU. A temporary directory is created by the program and deleted at the end of the run unless the debug option is turned on (see below). The entry may be either a full pathname i.e. beginning with a `/`, or else the pathname relative to the user's home directory on the machine on which the job is to be run. For example if the user's home directory is to be used for the temporary NAMMU directory this line of the file should contain only a dot i.e. `."`.
- The name of a directory, on the machine from which the job is submitted, to which the program output will be written. The same naming convention is used as for the previous item.

The installation script `nammu_install` produces a version of `.namhosts` that allows the program to be run on the machine from which the job is submitted. If the machine is called

`local_machine` and is of type `local_machine_type` and the `NAMMUSYS` directory is `/home/nammu` then the file is as follows:

```
local_machine
local_machine_type
/home/nammu/libnam
./tmp
nammu/output
```

Suppose now that NAMMU has also been compiled on another machine in the network called `remote_machine` which is of type `remote_machine_type`, and the compiled libraries have been placed in the directory `/home/nammu/libnam` on the machine `remote_machine`. In order to run NAMMU remotely on this machine the `.namhosts` file on `local_machine` should be edited to read as follows:

```
local_machine
local_machine_type
/home/nammu/libnam
./tmp
nammu/output
remote_machine
remote_machine_type
remote_machine:/home/nammu/libnam
./tmp
nammu/output
```

Finally, the user can make a private copy of the `.namhosts` file and specify that this is to be read rather than `NAMMUSYS/.namhosts` by using the `-f` option of the `nammu` command (see below).

5.2 The `nammu` Command

In this section we give a more complete description of the `nammu` command. The full syntax for the command is

```
nammu [-f filename] [-j jobname] [-v version] [-d] file
```

The options are:

- | | |
|---------------------------------|--|
| <code>-f <i>filename</i></code> | allows the user to specify that <i>filename</i> is to be used instead of <code>NAMMUSYS/.namhosts</code> . |
| <code>-j <i>jobname</i></code> | specifies that the job is to be called <i>jobname</i> . If this option is omitted a job name will be generated automatically (see section 3.2). |
| <code>-v <i>version</i></code> | specifies that version <i>version</i> of NAMMU to be used. The default version is 7.2. |
| <code>-d</code> | specifies that the debug option is to be used. The program is linked with the <code>-g</code> option so that debugging tools such as <code>dbx</code> can be used. Also the temporary directory created by the program is not removed at the end of the job. |
| <i>File</i> | the name of the file containing the input for the program. |

5.3 Temporary Directories Used

A number of temporary files are generated during a NAMMU run. These are stored in two temporary directories and are normally deleted at the end of a run. If the run terminates

NAMMU Installation and Running Guide

abnormally, or if a specified output file cannot be created (for example, if a disk is full, or file permissions are incorrect) these temporary directories and files will not be deleted. If this is the case they can safely be deleted by the user. The directories generated are:

<i>tgtemp.number</i>	in the user's home directory. This contains a number of files which are used during the job submission phase of a run.
<i>user/jobname</i>	in the scratch storage directory (see section 5.1). This contains the files produced during the execution of NAMMU. If the <code>-d</code> option of the <code>nammu</code> command is used the files in this directory will not be deleted at the end of the run. This is to enable the user to carry out various debugging operations.

Main Office

Risley

Serco Assurance
Thomson House
Risley, Warrington
Cheshire WA3 6AT
United Kingdom

T +44 (0)1925 254245

F +44 (0)1925 254464

www.sercoassurance.com

Regional Offices

Culham

Culham Science Centre
Culham, Abingdon
Oxfordshire OX14 3ED
United Kingdom

T +44 (0)1235 463722

F +44 (0)1235 464160

Dounreay

Dounreay
Thurso, Caithness
KW14 7TZ
United Kingdom

T +44 (0)1235 463722

F +44 (0)1235 464160

Harwell

Harwell, Didcot
Oxfordshire
OX11 0QJ
United Kingdom

T +44 (0)1235 433882

F +44 (0)1235 433674

Winfrith

Winfrith Technology Centre
Winfrith, Dorchester
Dorset DT2 8ZE
United Kingdom

T +44 (0)1305 202352

F +44 (0)1305 202194

Devonport

Unit 21
Mary Seacole Road
The Millfields
Plymouth
Devon

T +44 (0)1752 313306

F +44 (0)1752 313261