# NEMO version 1.1
# Numerical Engine for Multiphysics Operators

# Install Guide

Stefano Toninel *
David P. Schmnidt †
Salvatore Filippone ‡
Marco Rorro §

October 4, 2010

*University of Bologna; e–mail: stefano.toninel@mail.ing.unibo.it
†University of Massachusetts – Amherst; e–mail: schmidt@ecs.umass.edu
‡University of Rom – Tor Vergata; e–mail: salvatore.filippone@uniroma2.it
§CASPUR; e–mail: rorro@caspur.it

# Contents

# 1 Supported Platforms

The code has been successfully tested so far on the following operating systems/platforms:

- Linux (kernel 2.6.x) on IA–32 and AMD64;
- Mac Os X 10.4 (Tiger) on PowerPC–G4.

In order to build NEMO a standard Fortran 95 compiler is required. The following compilers are presently supported:

- GFortran 4.2.0 or later[1] included in the free GCC compiler suite [2];
- Intel Fortran Compiler 9.1 for Linux or later [3].

# 2 Prerequisites

Before compiling NEMO one should check that the following tools and packages are installed and available to use in the building procedure.

**An ISO Fortran 95 compiler** including support for TR15581 "Allocatable Extensions"; GFortran is the *reference compiler* for the NEMO project since it is free, open–source and available for practically every platform.

**BLAS** Basic Linear Algebra Subprograms. A generic implementation is available at http://www.netlib.org/blas/. This is easy to install and works well for debugging. For a better performance one can choose one of the following options:

- **ATLAS** (Automatically Tuned Linear Algebra Software) available at http://www.netlib.org/atlas/;
- a processor–dependent implementation like Intel–MKL (Math Kernel Library), ACML (AMD Core Math Library), etc.

[1] http://gcc.gnu.org/fortran/
[2] http://gcc.gnu.org/
[3] http://www.intel.com/

**LAPACK** Linear Algebra PACKage. A generic implementation is available at http://www.netlib.org/lapack/. Processor–optimized BLAS distributions, such as Intel–MKL and ACML, usually includes also their own LAPACK implementation. [4]

**MPI** An implementation of the Message Passing Interface standard, like **MPICH** available at http://www.mcs.anl.gov/research/projects/mpich2/; **MVAPICH** available at http://mvapich.cse.ohio-state.edu/ or **Open MPI** available at http://www.open-mpi.org/

**PSBLAS** 2.4.x Parallel Sparse BLAS. Available at http://www.ce.uniroma2.it/psblas/.

**CGNSlib** CFD General Notation System library. Available at http://www.cgns.org/.

**ParMETIS** Parallel METIS, a library for domain decomposition. Available at http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview

Install instructions for each one of the previously listed components are reported in the next sections.

## 2.1 GFortran

GFortran is a part of the GCC suite. Binary files are usually included in the most recent Linux distributions, such as Suse, Fedora Core, etc. However, since this open–source project is evolving very quickly, new bugs are discovered and old ones are fixed very frequently. Thus it is strongly recommended to use an up–to–date version of the compiler.

### 2.1.1 Binary Downloading

At the website http://gcc.gnu.org/wiki/GFortran#download weekly up–to–date binaries for all platforms supported by NEMO are available.

### 2.1.2 Snapshot Bootstrap

Alternatively one can download the most recent snapshot directly from a GCC mirror site[5] and compile it by means of a bootstrap procedure. Here

[4] Presently LAPACK routines are used only for benchmarking NEMO's own implementation of some numerical methods. Its installation is recommended for developers, while optional for users.

[5] http://gcc.gnu.org/mirrors.html

is a step–by–step guide for compiling GFortran on a UNIX–like box. The default install path is `$HOME/opt/gcc-4.5.0/`.

1. Install the prerequisites GMP[6] (GNU Multiple Precision Library) version 4.3.2 (or higher) and MPFR[7] version 2.4.2 (or higher). Usually Linux distributions include up–to–date releases of both packages. Otherwise one has to download the respective source files, compile, install them and update the library search path.

2. ```
   $ cd  ~/opt
   $ mkdir gcc-build
   $ cd gcc-build
   $ mkdir obj
   ```

3. Download from a GCC mirror site a recent release, for istance `gcc-4.5.0.tar.bz2`.

   The file must be saved in `$HOME/opt/gcc-build/`.

4. Expand the `.tar.bz2` archive:

   ```
   $ tar xvjf gcc-4.5.0.tar.bz2
   ```

   a directory named `gcc-4.5.0` will be created.

5. Configure the bootstrap procedure:

   ```
   $ cd obj
   $ ../gcc-4.5.0/configure --prefix=$HOME/opt/gcc-4.5.0/
   ```

   This generates the "Makefiles" needed by the building procedure. If the path for the GMP and MPFR objects is not included in the default library search path, one has to specify it explicitly by adding in the configure step:

   ```
   --with-gmp=/path/to/GMP --with-mpfr=/path/to/MPFR
   ```

6. Run the bootstrap job for building the package:

   ```
   $ nohup make bootstrap > make.log &
   ```

7. Install `gcc` and `gfortran`:

   ```
   $ make install
   ```

[6] http://gmplib.org/
[7] http://www.mpfr.org/
   ```

8. Update the library and executable search paths by adding the following strings:

```
PATH=$HOME/opt/gcc-4.5.0/bin:$PATH
[...]
export PATH
```

in `~/.bash_profile` or `~/.bashrc`, and

```
LD_LIBRARY_PATH=$HOME/opt/gcc-4.5.0/lib:$LD_LIBRARY_PATH
[...]
export LD_LIBRARY_PATH
```

in `~/.bashrc`. In particular, the second definition is mandatory for building MPICH with GFortran.

## 2.2 BLAS

The following steps explain how to compile and install the generic implementation of BLAS.

1. 
```
$ cd ~/LIB
$ mkdir tmp
```

2. Download `blas.tgz` from http://www.netlib.org/blas/ and copy it to `~/LIB/tmp`.

3. Untar the archive:

```
$ cd tmp
$ tar xvzf blas.tgz
$ rm blas.tgz
```

4. Compile the source files:

```
$ gfortran -O3 -ffast-math -mtune=<CPUtype> -c *.f
```

5. Build the library:

```
$ ar curv libblas-gfortran.a *.o
$ ranlib libblas-gfortran.a
```

6. Install and clean:

```
$ mv libblas-gfortran.a ~/LIB
$ cd ..; rm -r tmp blas.tgz
```

## 2.3 LAPACK

The building procedure for the LAPACK library requires the specification of a BLAS implementation and consists of the following steps.

1. Download `lapack.tgz` from http://www.netlib.org/lapack/ and copy it to `~/LIB`.

2. Untar the archive:

```
$ cd ~/LIB
$ tar xvzf lapack.tgz
$ cd LAPACK
```

3. Choose the `Make.inc` for a specific platform; for instance,

```
$ cp INSTALL/make.inc.LINUX make.inc
```

4. Edit the `make.inc` file, setting the following variables:

```
FORTRAN  = gfortran
OPTS     = -O3 -ffast-math -mtune=<CPUtype>
[...]
LOADER   = gfortran
LOADOPTS = $(OPTS)
[...]
BLASLIB   = $(HOME)/LIB/libblas-gfortran.a
LAPACKLIB = liblapack-gfortran.a
```

5. Build the single and double precision objects:

```
$ cd SRC
$ make single double
$ cd ..
```

6. Install and clean:

```
$ mv liblapack-gfortran.a ~/LIB
$ cd ..
$ rm -r LAPACK lapack.tgz
```

## 2.4 MPICH

This section contains the instructions for installing MPICH using `gfortran` and `gcc` respectively as Fortran and C compiler. The default command in the MPICH distribution for starting remote sessions during parallel jobs would be `rsh`. For a security–enhanced use one can choose `ssh` instead. The following steps refer to the latter option.

1. Download `mpich-1.2.6.tar.gz` (or a later version) and copy it to `~/LIB/`

2. Untar the archive:

```
$ cd ~/LIB
$ tar xvzf mpich-1.2.6.tar.gz
$ cd mpich-1.2.6
```

3. Set up the environment for next building procedure:

```
$ export FC=gfortran
$ export F90=gfortran
$ export RSHCOMMAND=ssh
$ export F77_GETARGDECL=" "
```

4. Configure and build:

```
$ configure --prefix=$HOME/mpich-gfortran
$ make
```

5. Install and clean:

```
$ make install
$ cd ..
$ rm -r mpich-1.2.6 mpich-1.2.6.tar.gz
```

6. In order to avoid the password authentication request at the login of every ssh session initiated by the `mpirun` command, one can generate a private/public key pair. If you do not already have a key pair, execute the following steps:

```
$ cd ~/.ssh
$ ssh-keygen -t rsa
[enter no passphrase]
$ cat id_rsa.pub >> authorized_keys2
```

Note that the permissions of the authorized_keys2 file should be set to `-rw-r--r--` .

7. Update search path by adding the following line in `~/.bash_profile` or `~/.bashrc` :

```
PATH=$HOME/LIB/mpich-gfortran/bin:$PATH
```

## 2.5 PSBLAS

The PSBLAS library supplies the numerical kernel of NEMO and requires the installation of the following packages: BLAS and MPI. The building procedure consists of the following steps.

1. Download `psblas2.4.x.tgz` from http://www.ce.uniroma2.it/psblas/ and copy it to `~/LIB/`.

2. Untar the archive:

```
$ cd ~/LIB
$ tar xvzf psblas2.4.x.tgz
$ cd psblas2.4.x
```

3. Configure and build:

```
$ ./configure --prefix=~LIB/psblas/2.4.0
$ make
```

   Additional packages path can be specified by adding in the configure step, for istance:

```
 --with-blas=$HOME/LIB/libblas-gfortran.a
```

   See `./configure --help` for additional options.

4. Install and clean:

```
$ make install
$ cd ..
$ rm -r psblas2.4.x psblas2.4.x.tgz
```

## 2.6 CGNSlib

1. Download `cgnslib_2.5-4.tar.gz` (or a later version) from http://cgns.sourceforge.net/ and copy it to `~/LIB`.

2. Untar the archive:

```
$ cd ~/LIB
$ tar xvzf cgnslib_2.3.tar.gz
$ mkdir CGNSlib
$ mkdir CGNSlib/lib
$ mkdir CGNSlib/include
$ cd cgnslib_2.5
```

3. Configure:

```
$ configure --prefix=$HOME/LIB/CGNSlib
```

4. Compile:

```
$ make
```

5. Install and clean:

```
$ make install
$ cd ..
$ rm -rf cgnslib_2.5 cgnslib_2.5-4.tar.gz
```

## 2.7 ParMETIS

1. Download `ParMetis-3.1.1tar.gz` (or a later version) from http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview and copy it to `~/LIB`.

2. Untar the archive:

```
$ cd ~/LIB
$ tar xvzf ParMetis-3.1.tar.gz
$ cd ParMetis-3.1
```

3. Compile:

```
$ make
```

4. Fix a pending bug in the building procedure:

```
$ mkdir tmp
$ cd tmp
$ ar xv ../libmetis.a
$ ar curv ../libparmetis.a parmetis.o
$ cd ..
$ ranlib libparmetis.a
$ rm -rf tmp
```

5. Install and clean:

```
$ mkdir ~/LIB/ParMetis
$ mv libparmetis.a libmetis.a ~/LIB/ParMetis
$ cd ..; rm -rf ParMetis-3.1 ParMetis-3.1.tar.gz
```

## 2.8 Special Remarks

1. BLAS, LAPACK, MPI and PSBLAS requires *distinct installations for different compilers*.

2. CGNSlib and PArMETIS can be built just once, by using the default compiler `gcc`.

3. Wherever not specified `$HOME/LIB` is the *default library path*.

# 3 NEMO

## 3.1 Code Compiling

After having installed all packages and tools described in the previous sections, one should be able to compile successfully NEMO, according to the following steps.

1. Enter the folder `Nemo/` containing the software distribution. Typing the `ls` command should return the following list of folders and files:

```
$ cd Nemo/
$ ls
applications  configure.ac  Makefile     nemo-ab-notes.txt
autogen.sh    docs          Make.inc.in  README
config        install-sh    missing      src
configure     LICENSE       mkdir.sh
```

2. Configure and build:

```
$ ./configure --prefix=~LIB/Nemo --with-psblas-dir=~LIB/psblas/2.4.0
$ make
```

   Additional packages path can be specified by adding in the configure step, for istance:

```
--with-blas=$HOME/LIB/libblas-gfortran.a
--with-cgns=~LIB/CGNSlib
--with-parmetis=~LIB/ParMetis
```

   See `./configure --help` for additional options. procedure described in the previous sections.

3. Install

```
$ make install
```

4. Install applications:

```
$ cd applications
$ make
```

   In order to build all NEMO–based applications included in the distribution

## 3.2 Applications running

The folder `nemo/applications` contains different executables. The subfolder `examples` contains two subfolders: `input` and `mesh`: the former provide the input files whereas the latter the mesh. In order to run an application, for instance, `steady-conduction`:

1. Set the input file:

```
$ cd Nemo/applications
$ cp examples/input/steady_conduction_3d.inp ./nemo.inp
```

   Edit the file `nemo.inp` to change some parameters, choose the resolution method and the format of output (DX ot VTK).

2. Run the application

```
$ mpiexec -np 16 ./steady-conduction
```

where `-np` specify the number of process to use.

3. Visualize the output file:
   To visualize DX file open `dx`[8] and run the appropriate .net file, for instance `temp_3d.net`.
   To visualize VTK file, run an application like ParaView[9] or Visit[10], open the corresponding .vtk file and choose the properties to visualize.

## 3.3 Documentation Compiling

The folder `Nemo/doc/` contains two documentation resources.

1. `Nemo/doc/pdf/` contains the LaTeXsource files for building the code documentation in `.pdf` format. A precompiled version of the current install–guide as well as other `.pdf` documents should be already available in the `/Nemo/doc/` folder.

2. `Nemo/doc/uml/` contains the uml diagrams which depict the object–oriented structure of the code. They can be analyzed and modified by using the free tool ArgoUML[11].

## 3.4 Cleaning

In order to clean the whole distribution and remove all compiling products (Fortran `.o` and `.mod`, as well as LaTeXfiles), type in the base directory `Nemo/`:

```
$ make clean
```

[8] http://www.opendx.org/
[9] http://www.paraview.org/
[10] https://wci.llnl.gov/codes/visit/
[11] http://argouml.tigris.org/