Compilation instructions

Jan/18/2022

Contents

These detailed compilation instructions are aimed at `amd64` Debian-based GNU/Linux distributions. The compilation procedure follows POSIX, so it should work in other operating systems and architectures as well. Distributions not using `apt` for packages (i.e. `yum`) should change the package installation commands (and possibly the package names). The instructions should also work for in MacOS, although the `apt-get` commands should be replaced by `brew` or similar. Same for Windows under Cygwin, the packages should be installed through the Cygwin installer. WSL was not tested, but should work as well.

1 Quickstart

Note that the quickest way to get started is to get an already-compiled statically-linked binary executable. Follow these instructions if that option is not suitable for your workflow.

On a GNU/Linux box (preferably Debian-based), follow these quick steps. See next section for detailed explanations.

To compile the Git repository, proceed as follows. This procedure does need `git` and `autoconf` but new versions can be pulled and recompiled easily.

1. Install mandatory dependencies

   ```
   sudo apt-get install gcc make git automake autoconf libgsl-dev
   ```

   If you cannot install `libgsl-dev` but still have `git` and the build toolchain, you can have the `configure` script to download and compile it for you. See point 4 below.

2. Install optional dependencies (of course these are *optional* but recommended)

   ```
   sudo apt-get install libsundials-dev petsc-dev slepc-dev
   ```

3. Clone Github repository

   ```
   git clone https://github.com/seamplex/feenox
   ```

4. Boostrap, configure, compile & make

   ```
   cd feenox
   ./autogen.sh
   ./configure
   make -j4
   ```

   If you cannot (or do not want) to use `libgsl-dev` from a package repository, call `configure` with `--enable-download-gsl`:

   ```
   ./configure --enable-download-gsl
   ```

   If you do not have Internet access, get the tarball manually, copy it to the same directory as `configure` and run again.

5. Run test suite (optional)

   ```
   make check
   ```

6. Install the binary system wide (optional)

   ```
   sudo make install
   ```

To stay up to date, pull and then autogen, configure and make (and optionally install):

```
git pull
./autogen.sh; ./configure; make -j4
sudo make install
```

2 Detailed configuration and compilation

The main target and development environment is Debian GNU/Linux, although it should be possible to compile FeenoX in any free GNU/Linux variant (and even the in non-free MacOS and Windows). As per the SRS, all dependencies have to be available on mainstream GNU/Linux distributions. But they can also be compiled from source in case the package repositories are not available or customized compilation flags are needed (i.e. optimization or debugging settings).

All the dependencies are free and open source software. PETSc/SLEPc also depend on other mathematical libraries to perform particular operations such as linear algebra. These extra dependencies can be either free (such as LAPACK) or non-free (such as MKL), but there is always at least one combination of a working setup that involves only free and open source software which is compatible with FeenoX licensing terms (GPLv3+). See the documentation of each package for licensing details.

2.1 Mandatory dependencies

FeenoX has one mandatory dependency for run-time execution and the standard build toolchain for compilation. It is written in C99 so only a C compiler is needed, although `make` is also required. Free and open source compilers are favored. The usual C compiler is `gcc` but `clang` can also be used. Nevertheless, the non-free `icc` has also been tested.

Note that there is no need to have a Fortran nor a C++ compiler to build FeenoX. They might be needed to build other dependencies (such as PETSc), but not to compile FeenoX with all the dependencies installed from package repositories. In case the build toolchain is not already installed, do so with

```
sudo apt-get install gcc make
```

If the source is to be fetched from the Git repository then of course `git` is needed but also `autoconf` and `automake` since the `configure` script is not stored in the Git repository but the `autogen.sh` script that bootstraps the tree and creates it. So if instead of compiling a source tarball one wants to clone from GitHub, these packages are also mandatory:

```
sudo apt-get install git automake autoconf
```

Again, chances are that any existing GNU/Linux box has all these tools already installed.

2.1.1 The GNU Scientific Library

The only run-time dependency is GNU GSL (not to be confused with Microsoft GSL). It can be installed with

```
sudo apt-get install libgsl-dev
```

In case this package is not available or you do not have enough permissions to install system-wide packages, there are two options.

1. Pass the option `--enable-download-gsl` to the `configure` script below.
2. Manually download, compile and install GNU GSL

If the `configure` script cannot find both the headers and the actual library, it will refuse to proceed. Note that the FeenoX binaries already contain a static version of the GSL so it is not needed to have it installed in order to run the statically-linked binaries.

2.2 Optional dependencies

FeenoX has three optional run-time dependencies. It can be compiled without any of these but functionality will be reduced:

- SUNDIALS provides support for solving systems of ordinary differential equations (ODEs) or differential-algebraic equations (DAEs). This dependency is needed when running inputs with the `PHASE_SPACE` ↩ keyword.
- PETSc provides support for solving partial differential equations (PDEs). This dependency is needed when running inputs with the `PROBLEM` keyword.
- SLEPc provides support for solving eigen-value problems in partial differential equations (PDEs). This dependency is needed for inputs with `PROBLEM` types with eigen-value formulations such as `modal` and `neutron_transport`.

In absence of all these, FeenoX can still be used to

- solve general mathematical problems such as the ones to compute the Fibonacci sequence,
- operate on functions, either algebraically or point-wise interpolated,
- read, operate over and write meshes,
- etc.

These optional dependencies have to be installed separately. There is no option to have `configure` to download them as with `--enable-download-gsl`.

2.2.1 SUNDIALS

SUNDIALS is a SUite of Nonlinear and DIfferential/ALgebraic equation Solvers. It is used by FeenoX to solve dynamical systems casted as DAEs with the keyword `PHASE_SPACE`, like the Lorenz system.

Install either by doing

```
sudo apt-get install libsundials-dev
```

or by following the instructions in the documentation.

2.2.2 PETSc

The Portable, Extensible Toolkit for Scientific Computation, pronounced PET-see (/ˈpɛt-siː/), is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It is used by FeenoX to solve PDEs with the keyword `PROBLEM`, like thermal conduction on a slab.

Install either by doing

```
sudo apt-get install petsc-dev
```

or by following the instructions in the documentation.

Note that

- Configuring and compiling PETSc from scratch might be difficult the first time. It has a lot of dependencies and options. Read the official documentation for a detailed explanation.
- There is a huge difference in efficiency between using PETSc compiled with debugging symbols and with optimization flags. Make sure to configure `--with-debugging=0` for FeenoX production runs and leave the debugging symbols (which is the default) for development only.
- FeenoX needs PETSc to be configured with real double-precision scalars. It will compile but will complain at run-time when using complex and/or single or quad-precision scalars.
- FeenoX honors the `PETSC_DIR` and `PETSC_ARCH` environment variables when executing `configure`. If these two do not exist or are empty, it will try to use the default system-wide locations (i.e. the `petsc-dev` package).

2.2.3 SLEPc

The Scalable Library for Eigenvalue Problem Computations, is a software library for the solution of large scale sparse eigenvalue problems on parallel computers. It is used by FeenoX to solve PDEs with the keyword `PROBLEM` that need eigen-value computations, such as modal analysis of a cantilevered beam.

Install either by doing

```
sudo apt-get install slepc-dev
```

or by following the instructions in the documentation.

Note that

- SLEPc is an extension of PETSc so the latter has to be already installed and configured.
- FeenoX honors the `SLEPC_DIR` environment variable when executing `configure`. If it does not exist or is empty it will try to use the default system-wide locations (i.e. the `slepc-dev` package).
- If PETSc was configured with `--download-slepc` then the `SLEPC_DIR` variable has to be set to the directory inside `PETSC_DIR` where SLEPc was cloned and compiled.

2.3 FeenoX source code

There are two ways of getting FeenoX' source code:

1. Cloning the GitHub repository at `https://github.com/seamplex/feenox`
2. Downloading a source tarball from `https://seamplex.com/feenox/dist/src/`

2.3.1 Git repository

The main Git repository is hosted on GitHub at `https://github.com/seamplex/feenox`. It is public so it can be cloned either through HTTPS or SSH without needing any particular credentials. It can also be forked freely. See the Programming Guide for details about pull requests and/or write access to the main repository.

Ideally, the `main` branch should have a usable snapshot. All other branches might contain code that might not compile or might not run or might not be tested. If you find a commit in the main branch that does not pass the tests, please report it in the issue tracker ASAP.

After cloning the repository

```
git clone https://github.com/seamplex/feenox
```

the `autogen.sh` script has to be called to bootstrap the working tree, since the `configure` script is not stored in the repository but created from `configure.ac` (which is in the repository) by `autogen`.

Similarly, after updating the working tree with

```
git pull
```

it is recommended to re-run the `autogen.sh` script. It will do a `make clean` and re-compute the version string.

2.3.2 Source tarballs

When downloading a source tarball, there is no need to run `autogen.sh` since the `configure` script is already included in the tarball. This method cannot update the working tree. For each new FeenoX release, the whole tarball has to be downloaded again.

2.4 Configuration

To create a proper `Makefile` for the particular architecture, dependencies and compilation options, the script `configure` has to be executed. This procedure follows the GNU Coding Standards.

```
./configure
```

Without any particular options, `configure` will check if the mandatory GNU Scientific Library is available (both its headers and run-time library). If it is not, then the option `--enable-download-gsl` can be used. This option will try to use `wget` (which should be installed) to download a source tarball, uncompress is, configure and compile it. If these steps are successful, this GSL will be statically linked into the resulting FeenoX executable. If there is no internet connection, the `configure` script will say that the download failed. In that case, get the indicated tarball file manually , copy it into the current directory and re-run `./configure`.

The script will also check for the availability of optional dependencies. At the end of the execution, a summary of what was found (or not) is printed in the standard output:

```
$ ./configure
[...]
## ----------------------- ##
## Summary of dependencies ##
## ----------------------- ##
  GNU Scientific Library  from system
  SUNDIALS IDA            yes
  PETSc                   yes /usr/lib/petsc
  SLEPc                   no
[...]
```

If for some reason one of the optional dependencies is available but FeenoX should not use it, then pass `--without-sundials`, `--without-petsc` and/or `--without-slepc` as arguments. For example

```
$ ./configure --without-sundials --without-petsc
[...]
## ----------------------- ##
## Summary of dependencies ##
## ----------------------- ##
  GNU Scientific Library  from system
  SUNDIALS                no
  PETSc                   no
  SLEPc                   no
[...]
```

If configure complains about contradicting values from the cached ones, run `autogen.sh` again before `configure` or uncompress the source tarball in a fresh location.

To see all the available options run

```
./configure --help
```

2.5 Source code compilation

After the successful execution of `configure`, a `Makefile` is created. To compile FeenoX, just execute

```
make
```

Compilation should take a dozen of seconds. It can be even sped up by using the `-j` option

```
make -j8
```

The binary executable will be located in the `src` directory but a copy will be made in the base directory as well. Test it by running without any arguments

```
$ ./feenox
FeenoX v0.1.24-g6cfe063
a free no-fee no-X uniX-like finite-element(ish) computational engineering tool

usage: ./feenox [options] inputfile [replacement arguments]

  -h, --help         display usage and commmand-line help and exit
  -v, --version      display brief version information and exit
  -V, --versions     display detailed version information
  -s, --sumarize     list all symbols in the input file and exit

Instructions will be read from standard input if “”- is passed as
inputfile, i.e.

    $ echo "PRINT 2+2" | feenox -
    4

Report bugs at https://github.com/seamplex/feenox or to jeremy@seamplex.com
Feenox home page: https://www.seamplex.com/feenox/
$
```

The `-v` (or `--version`) option shows the version and a copyright notice:

```
$ ./feenox -v
FeenoX v0.1.24-g6cfe063
a free no-fee no-X uniX-like finite-element(ish) computational engineering tool

Copyright (C) 2009--2021 jeremy theler
GNU General Public License v3+, https://www.gnu.org/licenses/gpl.html.
FeenoX is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
$
```

The `-V` (or `--versions`) option shows the dates of the last commits, the compiler options and the versions of the linked libraries:

```
$ ./feenox -V
FeenoX v0.1.24-g6cfe063
a free no-fee no-X uniX-like finite-element(ish) computational engineering tool

Last commit date   : Sun Aug 29 11:34:04 2021 -0300
Build date         : Sun Aug 29 11:44:50 2021 -0300
Build architecture : linux-gnu x86_64
Compiler           : gcc (Ubuntu 10.3.0-1ubuntu1) 10.3.0
Compiler flags     : -O3
Builder            : gtheler@chalmers
```

```
GSL version        : 2.6
SUNDIALS version   : 4.1.0
PETSc version      : Petsc Release Version 3.14.5, Mar 03, 2021
PETSc arch         :
PETSc options      : --build=x86_64-linux-gnu --prefix=/usr --includedir=${prefix}/include --mandir=${prefix ↩
     }/share/man --infodir=${prefix}/share/info --sysconfdir=/etc --localstatedir=/var --with-option- ↩
     checking=0 --with-silent-rules=0 --libdir=${prefix}/lib/x86_64-linux-gnu --runstatedir=/run --with- ↩
     maintainer-mode=0 --with-dependency-tracking=0 --with-debugging=0 --shared-library-extension=_real -- ↩
     with-shared-libraries --with-pic=1 --with-cc=mpicc --with-cxx=mpicxx --with-fc=mpif90 --with-cxx- ↩
     dialect=C++11 --with-opencl=1 --with-blas-lib=-lblas --with-lapack-lib=-llapack --with-scalapack=1 -- ↩
     with-scalapack-lib=-lscalapack-openmpi --with-ptscotch=1 --with-ptscotch-include=/usr/include/scotch -- ↩
     with-ptscotch-lib="-lptesmumps -lptscotch -lptscotcherr" --with-fftw=1 --with-fftw-include="[]" --with- ↩
     fftw-lib="-lfftw3 -lfftw3_mpi" --with-superlu_dist=1 --with-superlu_dist-include=/usr/include/superlu- ↩
     dist --with-superlu_dist-lib=-lsuperlu_dist --with-hdf5-include=/usr/include/hdf5/openmpi --with-hdf5- ↩
     lib="-L/usr/lib/x86_64-linux-gnu/hdf5/openmpi -L/usr/lib/x86_64-linux-gnu/openmpi/lib -lhdf5 -lmpi" -- ↩
     CXX_LINKER_FLAGS=-Wl,--no-as-needed --with-hypre=1 --with-hypre-include=/usr/include/hypre --with-hypre ↩
     -lib=-lHYPRE_core --with-mumps=1 --with-mumps-include="[]" --with-mumps-lib="-ldmumps -lzmumps -lsmumps ↩
      -lcmumps -lmumps_common -lpord" --with-suitesparse=1 --with-suitesparse-include=/usr/include/ ↩
     suitesparse --with-suitesparse-lib="-lumfpack -lamd -lcholmod -lklu" --with-superlu=1 --with-superlu- ↩
     include=/usr/include/superlu --with-superlu-lib=-lsuperlu --prefix=/usr/lib/petscdir/petsc3.14/x86_64- ↩
     linux-gnu-real --PETSC_ARCH=x86_64-linux-gnu-real CFLAGS="-g -O2 -ffile-prefix-map=/build/petsc-pVufYp/ ↩
     petsc-3.14.5+dfsg1=. -flto=auto -ffat-lto-objects -fstack-protector-strong -Wformat -Werror=format- ↩
     security -fPIC" CXXFLAGS="-g -O2 -ffile-prefix-map=/build/petsc-pVufYp/petsc-3.14.5+dfsg1=. -flto=auto  ↩
     -ffat-lto-objects -fstack-protector-strong -Wformat -Werror=format-security -fPIC" FCFLAGS="-g -O2 - ↩
     ffile-prefix-map=/build/petsc-pVufYp/petsc-3.14.5+dfsg1=. -flto=auto -ffat-lto-objects -fstack- ↩
     protector-strong -fPIC -ffree-line-length-0" FFLAGS="-g -O2 -ffile-prefix-map=/build/petsc-pVufYp/petsc ↩
     -3.14.5+dfsg1=. -flto=auto -ffat-lto-objects -fstack-protector-strong -fPIC -ffree-line-length-0"  ↩
     CPPFLAGS="-Wdate-time -D_FORTIFY_SOURCE=2" LDFLAGS="-Wl,-Bsymbolic-functions -flto=auto -Wl,-z,relro - ↩
     fPIC" MAKEFLAGS=w
SLEPc version      : SLEPc Release Version 3.14.2, Feb 01, 2021
$
```

2.6 Test suite

To be explained.

2.7 Installation

To be explained.

3 Advanced settings

3.1 Compiling with debug symbols

By default the C flags are `-O3`, without debugging. To add the `-g` flag, just use `CFLAGS` when configuring:

```
./configure CFLAGS="-g -O0"
```

3.2 Using a different compiler

Without PETSc, FeenoX uses the `CC` environment variable to set the compiler. So configure like

```
./configure CC=clang
```

When PETSc is detected FeenoX uses the `mpicc` executable, which is a wrapper to an actual C compiler with extra flags needed to find the headers and the MPI library. To change the wrapped compiler, you should set `MPICH_CC` or `OMPI_CC`, depending if you are using MPICH or OpenMPI. For example, to force MPICH to use `clang` do

```
./configure MPICH_CC=clang CC=clang
```

To know which is the default MPI implementation, just run `configure` without arguments and pay attention to the "Compiler" line in the "Summary of dependencies" section. For example, for OpenMPI a typical summary would be

```
## ---------------------- ##
## Summary of dependencies ##
## ---------------------- ##
  GNU Scientific Library  from system
  SUNDIALS                yes
  PETSc                   yes /usr/lib/petsc
  SLEPc                   yes /usr/lib/slepc
  Compiler                gcc -I/usr/lib/x86_64-linux-gnu/openmpi/include/openmpi -I/usr/lib/x86_64-linux- ↩
       gnu/openmpi/include -pthread -L/usr/lib/x86_64-linux-gnu/openmpi/lib -lmpi
```

For MPICH:

```
## ---------------------- ##
## Summary of dependencies ##
## ---------------------- ##
  GNU Scientific Library  from system
  SUNDIALS                yes
  PETSc                   yes /home/gtheler/libs/petsc-3.15.0 arch-linux2-c-debug
  SLEPc                   yes /home/gtheler/libs/slepc-3.15.1
  Compiler                gcc -Wl,-z,relro -I/usr/include/x86_64-linux-gnu/mpich -L/usr/lib/x86_64-linux-gnu ↩
        -lmpich
```

Other non-free implementations like Intel MPI might work but were not tested. However, it should be noted that the MPI implementation used to compile FeenoX has to match the one used to compile PETSc. Therefore, if you compiled PETSc on your own, it is up to you to ensure MPI compatibility. If you are using PETSc as provided by your distribution's repositories, you will have to find out which one was used (it is usually OpenMPI) and use the same one when compiling FeenoX.

The FeenoX executable will show the configured compiler and flags when invoked with the `--versions` option:

```
$ feenox --versions
FeenoX v0.1.47-g868dbb7-dirty
a free no-fee no-X uniX-like finite-element(ish) computational engineering tool
```

```
Last commit date   : Mon Sep 6 16:39:53 2021 -0300
Build date         : Tue Sep 07 14:29:42 2021 -0300
Build architecture : linux-gnu x86_64
Compiler           : gcc (Debian 10.2.1-6) 10.2.1 20210110
Compiler flags     : -O3
Builder            : gtheler@tom
GSL version        : 2.6
SUNDIALS version   : 5.7.0
PETSc version      : Petsc Release Version 3.15.0, Mar 30, 2021
PETSc arch         : arch-linux2-c-debug
PETSc options      : --download-eigen --download-hdf5 --download-hypre --download-metis --download-mumps -- ↩
     download-parmetis --download-pragmatic --download-scalapack --with-x=0
SLEPc version      : SLEPc Release Version 3.15.1, May 28, 2021
$
```

Note that the reported values are the ones used in `configure` and not in `make`. Thus, the recommended way to set flags is in `configure` and not in `make`.

3.3 Compiling PETSc

To be explained.