Compilation instructions

Contents

These detailed compilation instructions are aimed at `amd64` Debian-based GNU/Linux distributions. The compilation procedure follows the POSIX standard, 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 download an already-compiled statically-linked binary executable. Note that getting a binary is the quickest and easiest way to go but it is the less flexible one. Mind the following instructions if a binary-only option is not suitable for your workflow and/or you do need to compile the source code from scratch.

On a GNU/Linux box (preferably Debian-based), follow these quick steps. See sec. 2 for the actual 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. If something goes wrong and you get an error, do not hesitate to ask in FeenoX's discussion page.

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. See the detailed compilation instructions for an explanation.

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/or Windows platforms) running in virtually any hardware platform. FeenoX can run be run either in HPC cloud servers or a Raspberry Pi, and almost everything that sits in the middle.

Following the UNIX philosophy discussed in the SDS, FeenoX re-uses a lot of already-existing high-quality free and open source libraries that implement a wide variety of mathematical operations. This leads to a number of dependencies that FeenoX needs in order to implement certain features.

There is only one dependency that is mandatory, namely GNU GSL (see sec. 2.1.1), which if it not found then FeenoX cannot be compiled. All other dependencies are optional, meaning that FeenoX can be compiled but its capabilities will be partially reduced.

As per the SRS, all dependencies have to be available on mainstream GNU/Linux distributions and have to be free and open source software. 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).

In particular, PETSc (and SLEPc) also depend on other mathematical libraries to perform particular operations such as low-level linear algebra operations. These extra dependencies can be either free (such as LAPACK) or non-free (such as Intel's 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` or Intel's `icc` and the newer `icx` can also be used.

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 and its dependencies), but not to compile FeenoX if all the dependencies are installed from the oeprating system's 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 not only is `git` 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_sn`.

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

- solve general mathematical problems such as the ones to compute the Fibonacci sequence or the Logistic map,
- operate on functions, either algebraically or point-wise interpolated such as Computing the derivative of a function as a UNIX filter
- 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`. When running the test suite (sec. 2.6), those tests that need an optional dependency which was not found at compile time will be skipped.

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 the NAFEMS LE10 benchmark problem.

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 and debugging 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's 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 can 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.sh`.

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 source 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, 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` and/or clone/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 v0.2.14-gbbf48c9
a free no-fee no-X uniX-like finite-element(ish) computational engineering tool

usage: feenox [options] inputfile [replacement arguments] [petsc options]

  -h, --help         display options and detailed explanations of commmand-line usage
  -v, --version      display brief version information and exit
  -V, --versions     display detailed version information

Run with --help for further explanations.
$
```

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

```
$ ./feenox -v
FeenoX v0.2.14-gbbf48c9
a free no-fee no-X uniX-like finite-element(ish) computational engineering tool

Copyright © 2009--2022 Seamplex, https://seamplex.com/feenox
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 version   : gcc (Debian 10.2.1-6) 10.2.1 20210110
Compiler expansion : gcc -Wl,-z,relro -I/usr/include/x86_64-linux-gnu/mpich -L/usr/lib/x86_64-linux-gnu - ↩
     lmpich
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

The `test` directory contains a set of test cases whose output is known so that unintended regressions can be detected quickly (see the programming guide for more information). The test suite ought to be run after each modification in FeenoX's source code. It consists of a set of scripts and input files needed to solve dozens of cases. The output of each execution is compared to a reference solution. In case the output does not match the reference, the test suite fails.

After compiling FeenoX as explained in sec. 2.5, the test suite can be run with `make check`. Ideally everything should be green meaning the tests passed:

```
$ make check
Making check in src
```

Compilation instructions

```
make[1]: Entering directory '/home/gtheler/codigos/feenox/src'
make[1]: Nothing to be done for 'check'.
make[1]: Leaving directory '/home/gtheler/codigos/feenox/src'
make[1]: Entering directory '/home/gtheler/codigos/feenox'
cp -r src/feenox .
make  check-TESTS
make[2]: Entering directory '/home/gtheler/codigos/feenox'
make[3]: Entering directory '/home/gtheler/codigos/feenox'
XFAIL: tests/abort.sh
PASS: tests/algebraic_expr.sh
PASS: tests/beam-modal.sh
PASS: tests/beam-ortho.sh
PASS: tests/builtin.sh
PASS: tests/cylinder-traction-force.sh
PASS: tests/default_argument_value.sh
PASS: tests/expressions_constants.sh
PASS: tests/expressions_variables.sh
PASS: tests/expressions_functions.sh
PASS: tests/exp.sh
PASS: tests/i-beam-euler-bernoulli.sh
PASS: tests/iaea-pwr.sh
PASS: tests/iterative.sh
PASS: tests/fit.sh
PASS: tests/function_algebraic.sh
PASS: tests/function_data.sh
PASS: tests/function_file.sh
PASS: tests/function_vectors.sh
PASS: tests/integral.sh
PASS: tests/laplace2d.sh
PASS: tests/materials.sh
PASS: tests/mesh.sh
PASS: tests/moment-of-inertia.sh
PASS: tests/nafems-le1.sh
PASS: tests/nafems-le10.sh
PASS: tests/nafems-le11.sh
PASS: tests/nafems-t1-4.sh
PASS: tests/nafems-t2-3.sh
PASS: tests/neutron_diffusion_src.sh
PASS: tests/neutron_diffusion_keff.sh
PASS: tests/parallelepiped.sh
PASS: tests/point-kinetics.sh
PASS: tests/print.sh
PASS: tests/thermal-1d.sh
PASS: tests/thermal-2d.sh
PASS: tests/trig.sh
PASS: tests/two-cubes-isotropic.sh
PASS: tests/two-cubes-orthotropic.sh
PASS: tests/vector.sh
XFAIL: tests/xfail-few-properties-ortho-young.sh
XFAIL: tests/xfail-few-properties-ortho-poisson.sh
XFAIL: tests/xfail-few-properties-ortho-shear.sh
============================================================================
Testsuite summary for feenox v0.2.6-g3237ce9
============================================================================
```

```
# TOTAL: 43
# PASS:  39
# SKIP:  0
# XFAIL: 4
# FAIL:  0
# XPASS: 0
# ERROR: 0
============================================================================
make[3]: Leaving directory '/home/gtheler/codigos/feenox'
make[2]: Leaving directory '/home/gtheler/codigos/feenox'
make[1]: Leaving directory '/home/gtheler/codigos/feenox'
$
```

The XFAIL result means that those cases are expected to fail (they are there to test if FeenoX can handle errors). Failure would mean they passed. In case FeenoX was not compiled with any optional dependency, the corresponding tests will be skipped. Skipped tests do not mean any failure, but that the compiled FeenoX executable does not have the full capabilities. For example, when configuring with `./configure --without-petsc` (but with SUNDIALS), the test suite output should be a mixture of green and blue:

```
$ ./configure --without-petsc
[...]
configure: creating ./src/version.h
## ---------------------- ##
## Summary of dependencies ##
## ---------------------- ##
  GNU Scientific Library  from system
  SUNDIALS                yes
  PETSc                   no
  SLEPc                   no
  Compiler                gcc
checking that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating Makefile
config.status: creating src/Makefile
config.status: creating doc/Makefile
config.status: executing depfiles commands
$ make
[...]
$ make check
Making check in src
make[1]: Entering directory '/home/gtheler/codigos/feenox/src'
make[1]: Nothing to be done for 'check'.
make[1]: Leaving directory '/home/gtheler/codigos/feenox/src'
make[1]: Entering directory '/home/gtheler/codigos/feenox'
cp -r src/feenox .
make  check-TESTS
make[2]: Entering directory '/home/gtheler/codigos/feenox'
make[3]: Entering directory '/home/gtheler/codigos/feenox'
XFAIL: tests/abort.sh
PASS: tests/algebraic_expr.sh
SKIP: tests/beam-modal.sh
SKIP: tests/beam-ortho.sh
```

Compilation instructions

```
PASS: tests/builtin.sh
SKIP: tests/cylinder-traction-force.sh
PASS: tests/default_argument_value.sh
PASS: tests/expressions_constants.sh
PASS: tests/expressions_variables.sh
PASS: tests/expressions_functions.sh
PASS: tests/exp.sh
SKIP: tests/i-beam-euler-bernoulli.sh
SKIP: tests/iaea-pwr.sh
PASS: tests/iterative.sh
PASS: tests/fit.sh
PASS: tests/function_algebraic.sh
PASS: tests/function_data.sh
PASS: tests/function_file.sh
PASS: tests/function_vectors.sh
PASS: tests/integral.sh
SKIP: tests/laplace2d.sh
PASS: tests/materials.sh
PASS: tests/mesh.sh
PASS: tests/moment-of-inertia.sh
SKIP: tests/nafems-le1.sh
SKIP: tests/nafems-le10.sh
SKIP: tests/nafems-le11.sh
SKIP: tests/nafems-t1-4.sh
SKIP: tests/nafems-t2-3.sh
SKIP: tests/neutron_diffusion_src.sh
SKIP: tests/neutron_diffusion_keff.sh
SKIP: tests/parallelepiped.sh
PASS: tests/point-kinetics.sh
PASS: tests/print.sh
SKIP: tests/thermal-1d.sh
SKIP: tests/thermal-2d.sh
PASS: tests/trig.sh
SKIP: tests/two-cubes-isotropic.sh
SKIP: tests/two-cubes-orthotropic.sh
PASS: tests/vector.sh
SKIP: tests/xfail-few-properties-ortho-young.sh
SKIP: tests/xfail-few-properties-ortho-poisson.sh
SKIP: tests/xfail-few-properties-ortho-shear.sh
============================================================================
Testsuite summary for feenox v0.2.6-g3237ce9
============================================================================
# TOTAL: 43
# PASS:  21
# SKIP:  21
# XFAIL: 1
# FAIL:  0
# XPASS: 0
# ERROR: 0
============================================================================
make[3]: Leaving directory '/home/gtheler/codigos/feenox'
make[2]: Leaving directory '/home/gtheler/codigos/feenox'
make[1]: Leaving directory '/home/gtheler/codigos/feenox'
$
```

To illustrate how regressions can be detected, let us add a bug deliberately and re-run the test suite.

Edit the source file that contains the shape functions of the second-order tetrahedra `src/mesh/tet10.c`, find the function `feenox_mesh_tet10_h()` and randomly change a sign, i.e. replace

```
      return t*(2*t-1);
```

with

```
      return t*(2*t+1);
```

Save, recompile, and re-run the test suite to obtain some red:

```
$ git diff src/mesh/
diff --git a/src/mesh/tet10.c b/src/mesh/tet10.c
index 72bc838..293c290 100644
--- a/src/mesh/tet10.c
+++ b/src/mesh/tet10.c
@@ -227,7 +227,7 @@ double feenox_mesh_tet10_h(int j, double *vec_r) {
       return s*(2*s-1);
       break;
     case 3:
-      return t*(2*t-1);
+      return t*(2*t+1);
       break;

     case 4:
$ make
[...]
$ make check
Making check in src
make[1]: Entering directory '/home/gtheler/codigos/feenox/src'
make[1]: Nothing to be done for 'check'.
make[1]: Leaving directory '/home/gtheler/codigos/feenox/src'
make[1]: Entering directory '/home/gtheler/codigos/feenox'
cp -r src/feenox .
make  check-TESTS
make[2]: Entering directory '/home/gtheler/codigos/feenox'
make[3]: Entering directory '/home/gtheler/codigos/feenox'
XFAIL: tests/abort.sh
PASS: tests/algebraic_expr.sh
FAIL: tests/beam-modal.sh
PASS: tests/beam-ortho.sh
PASS: tests/builtin.sh
PASS: tests/cylinder-traction-force.sh
PASS: tests/default_argument_value.sh
PASS: tests/expressions_constants.sh
PASS: tests/expressions_variables.sh
PASS: tests/expressions_functions.sh
PASS: tests/exp.sh
PASS: tests/i-beam-euler-bernoulli.sh
```

Compilation instructions

```
PASS: tests/iaea-pwr.sh
PASS: tests/iterative.sh
PASS: tests/fit.sh
PASS: tests/function_algebraic.sh
PASS: tests/function_data.sh
PASS: tests/function_file.sh
PASS: tests/function_vectors.sh
PASS: tests/integral.sh
PASS: tests/laplace2d.sh
PASS: tests/materials.sh
PASS: tests/mesh.sh
PASS: tests/moment-of-inertia.sh
PASS: tests/nafems-le1.sh
FAIL: tests/nafems-le10.sh
FAIL: tests/nafems-le11.sh
PASS: tests/nafems-t1-4.sh
PASS: tests/nafems-t2-3.sh
PASS: tests/neutron_diffusion_src.sh
PASS: tests/neutron_diffusion_keff.sh
FAIL: tests/parallelepiped.sh
PASS: tests/point-kinetics.sh
PASS: tests/print.sh
PASS: tests/thermal-1d.sh
PASS: tests/thermal-2d.sh
PASS: tests/trig.sh
PASS: tests/two-cubes-isotropic.sh
PASS: tests/two-cubes-orthotropic.sh
PASS: tests/vector.sh
XFAIL: tests/xfail-few-properties-ortho-young.sh
XFAIL: tests/xfail-few-properties-ortho-poisson.sh
XFAIL: tests/xfail-few-properties-ortho-shear.sh
============================================================================
Testsuite summary for feenox v0.2.6-g3237ce9
============================================================================
# TOTAL: 43
# PASS:  35
# SKIP:  0
# XFAIL: 4
# FAIL:  4
# XPASS: 0
# ERROR: 0
============================================================================
See ./test-suite.log
Please report to jeremy@seamplex.com
============================================================================
make[3]: *** [Makefile:1152: test-suite.log] Error 1
make[3]: Leaving directory '/home/gtheler/codigos/feenox'
make[2]: *** [Makefile:1260: check-TESTS] Error 2
make[2]: Leaving directory '/home/gtheler/codigos/feenox'
make[1]: *** [Makefile:1791: check-am] Error 2
make[1]: Leaving directory '/home/gtheler/codigos/feenox'
make: *** [Makefile:1037: check-recursive] Error 1
$
```

2.7 Installation

To be able to execute FeenoX from any directory, the binary has to be copied to a directory available in the `PATH` environment variable. If you have root access, the easiest and cleanest way of doing this is by calling `make install` with `sudo` or `su`:

```
$ sudo make install
Making install in src
make[1]: Entering directory '/home/gtheler/codigos/feenox/src'
gmake[2]: Entering directory '/home/gtheler/codigos/feenox/src'
 /usr/bin/mkdir -p '/usr/local/bin'
  /usr/bin/install -c feenox '/usr/local/bin'
gmake[2]: Nothing to be done for 'install-data-am'.
gmake[2]: Leaving directory '/home/gtheler/codigos/feenox/src'
make[1]: Leaving directory '/home/gtheler/codigos/feenox/src'
make[1]: Entering directory '/home/gtheler/codigos/feenox'
cp -r src/feenox .
make[2]: Entering directory '/home/gtheler/codigos/feenox'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/gtheler/codigos/feenox'
make[1]: Leaving directory '/home/gtheler/codigos/feenox'
$
```

If you do not have root access or do not want to populate `/usr/local/bin`, you can either

- Configure with a different prefix (not covered here), or
- Copy (or symlink) the `feenox` executable to `$HOME/bin`:

  ```
  mkdir -p ${HOME}/bin
  cp feenox ${HOME}/bin
  ```

  If you plan to regularly update FeenoX (which you should), you might want to symlink instead of copy so you do not need to update the binary in `$HOME/bin` each time you recompile:

  ```
  mkdir -p ${HOME}/bin
  ln -sf feenox ${HOME}/bin
  ```

Check that FeenoX is now available from any directory (note the command is `feenox` and not `./feenox`):

```
$ cd
$ feenox -v
FeenoX v0.2.14-gbbf48c9
a free no-fee no-X uniX-like finite-element(ish) computational engineering tool

Copyright © 2009--2022 Seamplex, https://seamplex.com/feenox
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.
$
```

If it is not and you went through the `$HOME/bin` path, make sure it is in the `PATH` (pun). Add

```
export PATH=${PATH}:${HOME}/bin
```

to your `.bashrc` in your home directory and re-login.

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

FeenoX uses the `CC` environment variable to set the compiler. So configure like

```
export CC=clang; ./configure
```

Note that the `CC` variable has to be *exported* and not *passed* to configure. That is to say, don't configure like

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

Mind also the following environment variables when using MPI-enabled PETSc:

- `MPICH_CC`
- `OMPI_CC`
- `I_MPI_CC`

Depending on how your system is configured, this last command might show `clang` but not actually use it. The FeenoX executable will show the configured compiler and flags when invoked with the `--versions` option:

```
$ feenox --versions
FeenoX v0.2.14-gbbf48c9
a free no-fee no-X uniX-like finite-element(ish) computational engineering tool

Last commit date   : Sat Feb 12 15:35:05 2022 -0300
Build date         : Sat Feb 12 15:35:44 2022 -0300
Build architecture : linux-gnu x86_64
Compiler version   : gcc (Debian 10.2.1-6) 10.2.1 20210110
Compiler expansion : gcc -Wl,-z,relro -I/usr/include/x86_64-linux-gnu/mpich -L/usr/lib/x86_64-linux-gnu - ↩
     lmpich
Compiler flags     : -O3
Builder            : gtheler@tom
GSL version        : 2.6
SUNDIALS version   : 5.7.0
PETSc version      : Petsc Release Version 3.16.3, Jan 05, 2022
PETSc arch         : arch-linux-c-debug
```

```
PETSc options      : --download-eigen --download-hdf5 --download-hypre --download-metis --download-mumps -- ↩
    download-parmetis --download-pragmatic --download-scalapack
SLEPc version      : SLEPc Release Version 3.16.1, Nov 17, 2021
$
```

You can check which compiler was actually used by analyzing the `feenox` binary as

```
$ objdump -s --section .comment ./feenox

./feenox:     file format elf64-x86-64

Contents of section .comment:
 0000 4743433a 20284465 6269616e 2031322e  GCC: (Debian 12.
 0010 322e302d 31342920 31322e32 2e300044  2.0-14) 12.2.0.D
 0020 65626961 6e20636c 616e6720 76657273  ebian clang vers
 0030 696f6e20 31342e30 2e3600             ion 14.0.6.
$
```

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. FeenoX has been tested using PETSc compiled with

- MPICH
- OpenMPI
- Intel MPI

3.3 Compiling PETSc

Particular explanation for FeenoX is to be done. For now, follow the general explanation from PETSc's website.

```
export PETSC_DIR=$PWD
export PETSC_ARCH=arch-linux-c-opt
./configure --with-debugging=0 --download-mumps --download-scalapack --with-cxx=0 --COPTFLAGS=-O3 -- ↩
    FOPTFLAGS=-O3
```

```
export PETSC_DIR=$PWD
./configure --with-debugging=0 --with-openmp=0 --with-x=0 --with-cxx=0 --COPTFLAGS=-O3 --FOPTFLAGS=-O3
make PETSC_DIR=/home/ubuntu/reflex-deps/petsc-3.17.2 PETSC_ARCH=arch-linux-c-opt all
```