Frequently Asked Questions about FeenoX

Jeremy Theler

2024-06-14

Contents

1 What is FeenoX?

It is "cloud-first a free no-fee no-X uniX-like finite-element(ish) computational engineering tool." Essentially, a finite-element program with a particular design basis:

> FeenoX is to finite-element programs and libraries what Markdown is to word processors (like Word) and typesetting systems (like TeX), respectively.

In increasing order of complexity and comprehensiveness, these resources explain what FeenoX is:

- The examples will give a brief overview of what FeenoX can do.
- The tutorials will walk your through how to use FeenoX to solve problems.
- The README in the GitHub repository has a brief introduction (after explaining why).
- Theler, J. (2024). FeenoX: a cloud-first finite-element(ish) computational engineering tool. Journal of Open Source Software, 9(95), 5846. https://doi.org/10.21105/joss.05846

  

- There is also a description in the documentation.
- FeenoX is an "offer" to a fictitious "tender" for a computational tool. The RFQ is the Software Requirements Specification and the explanation of how FeenoX addresses each requirement is the Software Design Specification.
- This presentation from August 2021 explains the SRS/SDS pair. The sources and the examples can be found in this Github repository. There is a recording of the presentation (audio is in Spanish).
- Finally the manual will be the ultimate guide.

2 How should I cite FeenoX?

If you use FeenoX and need to cite it, use this BiBTeX entry that points to the 2024 paper in JOSS:

```
@article{feenox-2024,
author = {Theler, Jeremy},
doi = {10.21105/joss.05846},
journal = {Journal of Open Source Software},
month = mar,
number = {95},
pages = {5846},
title = {{FeenoX: a cloud-first finite-element(ish) computational engineering tool}},
url = {https://joss.theoj.org/papers/10.21105/joss.05846},
volume = {9},
year = {2024}
}
```

If you are not using BiBTeX (which you should), just use the plain-text APA format:

Theler, J. (2024). FeenoX: a cloud-first finite-element(ish) computational engineering tool. Journal of Open Source Software, 9(95), 5846. https://doi.org/10.21105/joss.05846

3 What does FeenoX mean?

It does not mean anything particular, but

- The last X makes it rhyme with Unix and Linux.
- "noX" means that there is no graphical (i.e. *X*) interface
- Fee-no means that there are no fees involved (free as in "free beer")
- FeenoX is the successor of the now-superseded FEA program Fino
- It rhymes with FEniCS
- With some luck one can read "Finite ElEments NO-X"
- With mode luck, "FrEE" (as in "free speech")

4 How should FeenoX be pronounced?

It would be something like *fee-naaks*: /fiːnɑːks/
But whatever works for you is fine.

5 Why nothing happens when I double click on `feenox.exe`?

Because by design, FeenoX does not have a Graphical User Interface. Instead, it works like a transfer function between one or more input files and zero or more output files:

```
                            +------------+
 mesh (*.msh)  }            |            |             { terminal
 data (*.dat)  } input ---> |   FeenoX   |----> output { data files
 input (*.fee) }            |            |             { post (vtk/msh)
                            +------------+
```

Recall that FeenoX is designed as a cloud-first tool, and "the cloud" runs on Unix (essentially GNU/Linux) so FeenoX is based on the Unix philosophy. In this world, programs act like filters (i.e. transfer functions). They are executed from a command-line terminal. So instead of "double clicking" the executable, one has to open a terminal and execute it there. Without any arguments it says how to use it:

```
> feenox.exe
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.
>
```

With `-h` it gives more information:

```
> feenox.exe -h
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

  --progress         print ASCII progress bars when solving PDEs
  --mumps            ask PETSc to use the direct linear solver MUMPS
  --linear           force FeenoX to solve the PDE problem as linear
  --non-linear       force FeenoX to solve the PDE problem as non-linear

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

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

The optional [replacement arguments] part of the command line mean that
each argument after the input file that does not start with an hyphen
will be expanded verbatim in the input file in each occurrence of $1,
$2, etc. For example

    $ echo 'PRINT $1+$2' | feenox - 3 4
    7

PETSc and SLEPc options can be passed in [petsc options] as well, with
the difference that two hyphens have to be used instead of only once.
For example, to pass the PETSc option -ksp_view the actual FeenoX
invocation should be

    $ feenox input.fee --ksp_view

See https://www.seamplex.com/feenox/examples for annotated examples.

Report bugs at https://github.com/seamplex/feenox/issues
Ask questions at https://github.com/seamplex/feenox/discussions
Feenox home page: https://www.seamplex.com/feenox/
>
```

It is explained there that the main input file has to be given as the first argument. So go to the tests or examples directory, find a test you like and run it:

```
> cd tests
> feenox parallelepiped.fee
0.000295443
>
```

In any case, recall once again that FeenoX is a cloud-first tool, and Windows is not cloud-friendly, let alone cloud-first. It is time to re-think what you expect from a finite-element(ish) tool. If you still need a GUI, please check CAEplex.

Try to avoid Windows as much as you can. The binaries are provided as transitional packages for people that for some reason still use such an outdated, anachronous, awful and invasive operating system. They are compiled with Cygwin and have no support whatsoever. Really, really, **get rid of Windows ASAP**.

> “It is really worth any amount of time and effort to get away from Windows if you are doing computational science.”

> https://lists.mcs.anl.gov/pipermail/petsc-users/2015-July/026388.html

6 How do I create input decks for FeenoX?

FeenoX does not have "input decks." It has "input files," which are syntactically-sugared English-like plain-text ASCII files that describe the problem to be solved. First see the examples and the test directory. Then read the documentation.

There are syntax highlighting files for Kate and for Vim that helps the edition of input files. Contributions for other editors (emacs?) are welcome.

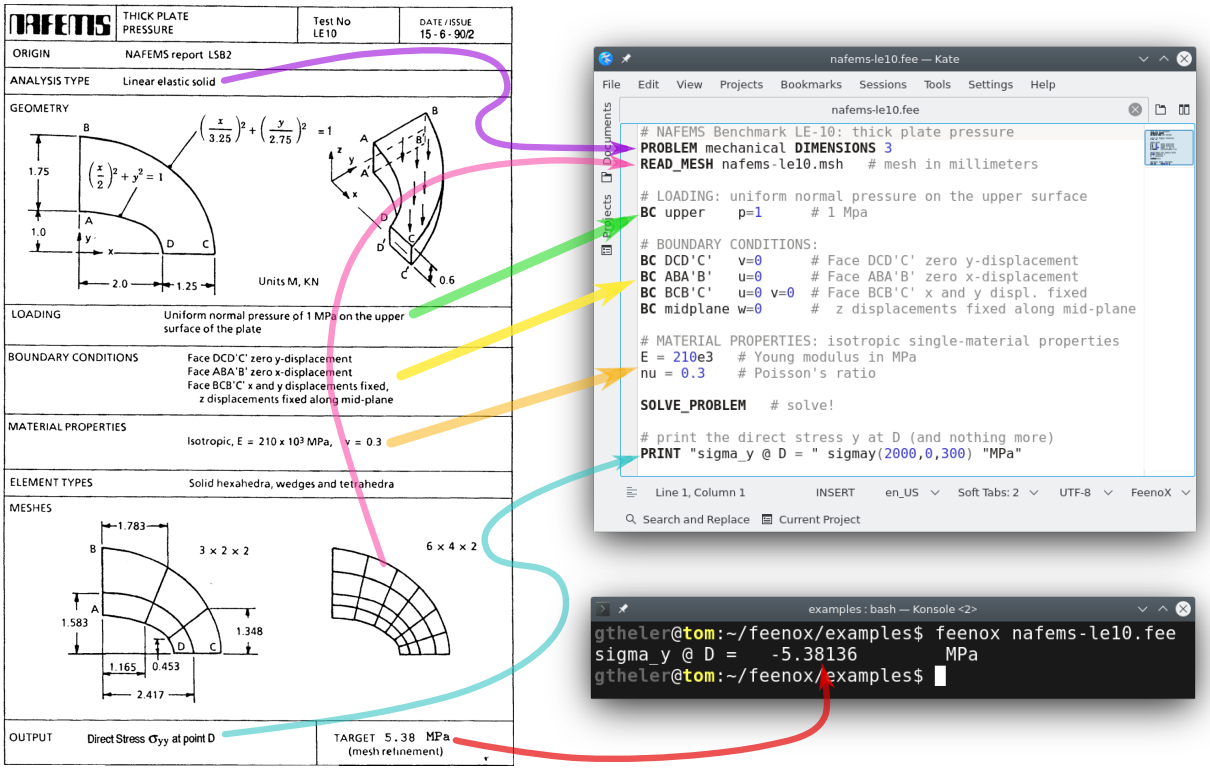

Figure 1: The Kate Text Editor can be used to prepare input files with syntax highlighting.

7 Does FeenoX support beam and/or shell elements?

No, it currently supports solid elements only. Therefore, three-dimensional problems need to have tetrahedra, hexahedra, prisms and/or pyramids; and two-dimensional problems need to have triangles or quadrangles.

It might support non-solid elements for elasticity in future versions, though. Contributions are welcome. Check out the contributing guidelines.

8 What license does FeenoX have?

TL;DR:

- The code is GPLv3+: you can use it, modify it and re-distribute it freely (as in free speech) as long as you keep the same licensing terms.
- The documentation is released under the terms of the GNU Free Documentation License version 1.3 or, at your option, any later version: same thing but with particular considerations for documentation instead of code.

FeenoX is licensed under the terms of the GNU General Public License version 3 or, at the user convenience, any later version. This means that users get the four essential freedoms:[1]

0. The freedom to *run* the program as they wish, for *any* purpose.
1. The freedom to *study* how the program works, and *change* it so it does their computing as they wish.
2. The freedom to *redistribute* copies so they can help others.
3. The freedom to *distribute* copies of their *modified* versions to others.

So a free program has to be open source, but it also has to explicitly provide the four freedoms above both through the written license and through appropriate mechanisms to get, modify, compile, run and document these modifications using well-established and/or reasonable straightforward procedures. That is why licensing FeenoX as GPLv3+ also implies that the source code and all the scripts and makefiles needed to compile and run it are available for anyone that requires it (i.e. it is compiled with `./configure ↩ && make`). Anyone wanting to modify the program either to fix bugs, improve it or add new features is free to do so. And if they do not know how to program, the have the freedom to hire a programmer to do it without needing to ask permission to the original authors. Even more, the documentation is released under the terms of the GNU Free Documentation License so these new (or modified) features can be properly documented as well.

Nevertheless, since these original authors are the copyright holders, they still can use it to either enforce or prevent further actions from the users that receive FeenoX under the GPLv3+. In particular, the license allows re-distribution of modified versions only if

a. they are clearly marked as different from the original, and
b. they are distributed under the same terms of the GPLv3+.

There are also some other subtle technicalities that need not be discussed here such as

- what constitutes a modified version (which cannot be redistributed under a different license)
- what is an aggregate (in which each part be distributed under different licenses)
- usage over a network and the possibility of using AGPL instead of GPL to further enforce freedom

These issues are already taken into account in the FeenoX licensing scheme.

It should be noted that not only is FeenoX free and open source, but also all of the libraries it depends on (and their dependencies) also are. It can also be compiled using free and open source build tool chains running over free and open source operating systems.

[1] There are some examples of pieces of computational software which are described as "open source" in which even the first of the four freedoms is denied. The most iconic case is that of Android, whose sources are readily available online but there is no straightforward way of updating one's mobile phone firmware with a customized version, not to mention vendor and hardware lock ins and the possibility of bricking devices if something unexpected happens. In the nuclear industry, it is the case of a Monte Carlo particle-transport program that requests users to sign an agreement about the objective of its usage before allowing its execution. The software itself might be open source because the source code is provided after signing the agreement, but it is not free (as in freedom) at all.

9 Why is FeenoX written in C and not in…

See the programming guide for further discussion.

9.1 C++?

Let us first start with some generalities

Why is C still in use even though we have C++? https://www.quora.com/Why-is-C-still-in-use-even-though-we-have-C++-Is-there-anything-that-C-can-do-but-C++-cant-or-maybe-something-that-is-easier-to-do-in-C-rather-than-C++

As a C programmer, why didn't you switch to C++ in your career? https://qr.ae/pGzfAO

Why is PETSc programmed in C, instead of Fortran or C++? C enables us to build data structures for storing sparse matrices, solver information, etc. in ways that Fortran simply does not allow. ANSI C is a complete standard that all modern C compilers support. The language is identical on all machines. C++ is still evolving and compilers on different machines are not identical. Using C function pointers to provide data encapsulation and polymorphism allows us to get many of the advantages of C++ without using such a large and more complicated language. It would be natural and reasonable to have coded PETSc in C++; we opted to use C instead.

https://www.mcs.anl.gov/petsc/documentation/faq.html#why-c

Why Git is written in C and not in C++, by Linus Torvalds C++ is a horrible language. It's made more horrible by the fact that a lot of substandard programmers use it, to the point where it's much much easier to generate total and utter crap with it. Quite frankly, even if the choice of C were to do *nothing* but keep the C++ programmers out, that in itself would be a huge reason to use C.

http://harmful.cat-v.org/software/c++/linus

In particular, I think that even though object-oriented programming does provide a heck of a paradigm (hey, I actually rewrote my Blackjack engine in C++ from its first version in C), it might subtly "force" people to do stuff that is either way too

- convoluted
- artificial
- hard to debug
- long to compile

I nevertheless think that

- std containers are pretty cool
- templating can give and edge on some cases
- sometimes OOP may be a better approach to the Unix rule of representation (sometimes)

However, the pros of C++ do not outweigh its cons for a cloud-first finite-elementish tool. Also, the name C++ is lame.

9.2 Fortran?

Because I am not insane (yet). I do not know any sane person that would start writing a piece of software from scratch using Fortran in the 21st century AD.

9.3 Python or R?

Python was not designed to perform actual computations but to add another layer so as to ease some (and only some) tasks. The actual computations are written in low-level languages, not in Python (nor Octave, R, etc.) And even if it was, I would not choose a language where scope depends on the indentation.

9.4 Go, Rust or Julia?

I don't know them in detail so I cannot tell if any of these languages would be a good fit for FeenoX. Keep in mind that it took me a while to know why not Fortran nor C++ even though there are people that would choose them over C. Maybe something of the sort happens with these new ideas (or not, I don't know).