The Journal of Open Source Software

FeenoX: a cloud-first finite-element(ish) computational engineering tool

Jeremy Theler [1,2]

1 Seamplex, Argentina **2** Instituto Balseiro, Argentina

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Kevin M. Moerman

Reviewers:

- @vijaysm
- @AnjaliSandip
- @chennachaos

Submitted: 27 July 2023
Published: unpublished

Summary

FeenoX is a cloud-first free no-X uniX-like finite-element(ish) computational engineering tool designed to solve engineering-related problems using cloud servers in parallel in such a way that the problem is defined in a plain-text near-English self descriptive input file read at run time, without requiring further user intervention after the invocation. FeenoX meets fictitious-yet-plausible Software Requirement Specifications (SRS). The FeenoX Software Design Specifications address each requirement of the SRS. FeenoX provides a set of common extents, capabilities and usefulness but offers different features (following slightly different spirits) for industry engineers, Unix hackers and academic researchers. The main features of this design basis are

- The tool has to be an already-compiled program (not a library) so regular users do not have to compile anything to solve a problem.
- Simple problems ought to need simple input files.
- There should be a one-to-one correspondence between the problem definition and FeenoX's input file, as illustrated in fig. 1.
- There should be an extension mechanism to allow hackers and researchers to add new partial differential equations to the tool.

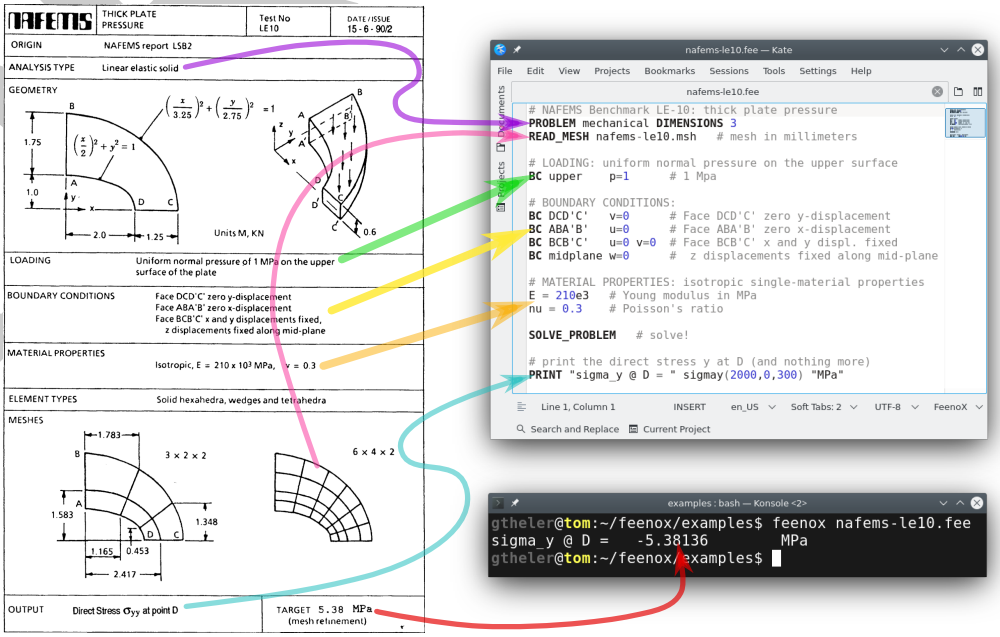


Figure 1: The NAFEMS LE10 problem statement (Finite Element Methods & Standards (Great Britain), 1990) and the corresponding FeenoX input illustrating the one-to-one correspondence between the two.

The Journal of Open Source Software

Statement of need

Open-source finite-element tools are either

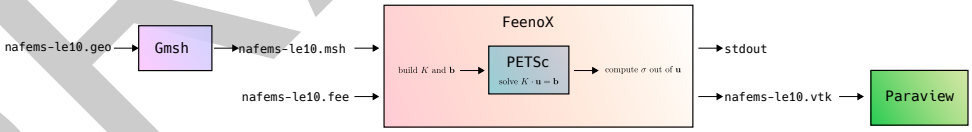
a. libraries which need code to use them such as
   - Sparselizard
   - MoFEM
   - FEniCS
   - MFEM
b. end-user programs which need a GUI such as
   - CalculiX
   - CodeAster

FeenoX sits in the middle. First, it can solve

- Basic mathematics
- Systems of ODEs/DAEs
- Laplace's equation
- Heat conduction
- Linear elasticity
- Modal analysis
- Neutron diffusion
- Neutron S_N

Second, it is the only free and open-source tool that satisfies the Software Requirement Specifications, including that…

- in order to solve a problem one needs to prepare a (relatively) simple input file (not a script nor a deck) which is read at run-time (not code which calls a library). For example, considering the NAFEMS LE10 Benchmark problem from fig. 1, FeenoX works as two "glue layers,"
  1. between the mesher Gmsh and the PETSc library (Balay et al., 1997, 2023)
  2. between the PETSc library and a post-processor such as Paraview



- these input files can expand generic command-line options using Bash syntax as $1, $2, etc., which allow parametric or optimization loops driven by higher-level scripts.
- for solving partial differential equations (PDEs), the input file has to refer to at least one Gmsh `.msh` file that defines the domain where the PDE is solved.
- the material properties and boundary conditions are defined using physical groups and not individual nodes nor elements, so the input file is independent of the mesh and thus can be tracked with Git to increase traceability and repeatability.
- it uses the Unix philosophy which, among others, separates policy from mechanism rendering FeenoX as a natural choice for web-based interfaces like CAEplex (fig. 2).

The Journal of Open Source Software

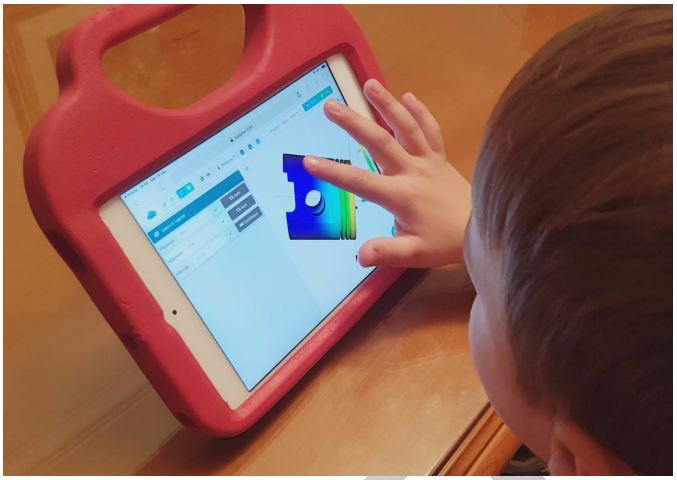


Figure 2: CAEplex is a web-based interface to solve thermo-mechanical problems in the cloud that uses FeenoX as the back end.

FeenoX tries to achieve its goals by…

- standing on both ethical (since it is free) and technical (since it is open source) grounds while interacting with other free and open operating systems, libraries, compilers and pre and post-processing tools, thus encouraging science and engineering to shift from privative environments into the free world.
- leveraging the Unix programming philosophy to come up with a cloud-first tool suitable to be automatically deployed and serve as the back end of web-based interfaces such as CAEplex.
- providing a ready-to-run program that reads an input file at run time (and not a library that has to be linked for each particular problem to be solved) as a deliberate design decision discussed in the Software Design Specifications.
- designing and implementing an extensibility mechanism to allow hackers and/or academics to add new PDE formulations by adding a new subdirectory to `src/pdes` in the repository and then
  a. re-bootstrapping with `autogen.sh`,
  b. re-configuring with `configure`, and
  c. re-compiling with `make`

In effect, FeenoX provides a general mathematical framework to solve PDEs with a bunch of entry points (as C functions) where new types of PDEs (e.g. electromagnetism, fluid mechanics, etc.) can be added to the set of what FeenoX can solve. This general framework provides means to

- parse the input file, handle command-line arguments, read mesh files, assign variables, evaluate conditionals, write results, etc.

```
PROBLEM laplace 2D
READ_MESH square-$1.msh
[...]
WRITE_RESULTS FORMAT vtk
```

- handle material properties given as algebraic expressions involving pointwise-defined functions of space, temperature, time, etc.

```
MATERIAL steel     E=210e3*(1-1e-3*(T(x,y,z)-20))   nu=0.3
MATERIAL aluminum  E=69e3                           nu=7/25
```

- read problem-specific boundary conditions as algebraic expressions

```
sigma = 5.670374419e-8  # W m^2 / K^4 as in wikipedia
e = 0.98      # non-dimensional
T0 = 1000     # K
Tinf = 300    # K

BC left  T=T0
BC right q=sigma*e*(Tinf^4-T(x,y,z)^4)
```

- access shape functions and its derivatives evaluated either at Gauss points or at arbitrary locations for computing elementary contributions to
  - stiffness matrix
  - mass matrix
  - right-hand side vector

  For example, this snippet would build the elemental stiffness matrix for the Laplace problem:

```c
int build_laplace_Ki(element_t *e, unsigned int q) {
  double wdet = feenox_fem_compute_w_det_at_gauss(e, q);
  gsl_matrix *B = feenox_fem_compute_B_at_gauss(e, q);
  feenox_call(feenox_blas_BtB_accum(B, wdet, feenox.fem.Ki));
  return FEENOX_OK;
}
```

  The calls for computing the weights and the matrices with the shape functions and/or their derivatives currently support first and second-order iso-geometric elements, but other element types can be added as well. More complex cases involving non-uniform material properties, volumetric sources, etc. can be found in the examples, tutorials and tests.

- solve the discretized equations using the appropriate PETSc (Balay et al., 1997, 2023) or SLEPc (Hernandez et al., 2005; Roman et al., 2023) objects, i.e.
  - KSP for linear static problems
  - SNES for non-linear static problems
  - TS for transient problems
  - EPS for eigenvalue problems

The particular functions that implement each problem type are located in subdirectories `src/pdes`, namely

- `laplace`
- `thermal`
- `mechanical`
- `modal`
- `neutron_diffusion`
- `neutron_sn`

Researchers with both knowledge of mathematical theory of finite elements and programming skills might, with the aid of the community, add support for other PDEs. They might do that by using one of these directories (say `laplace`) as a template and

1. replace every occurrence of `laplace` in symbol names with the name of the new PDE
2. modify the initialization functions in `init.c` and set
   - the names of the unknowns
   - the names of the material properties
   - the mathematical type and characteristics of problem
   - etc.

3. modify the contents of the elemental matrices in `bulk.c` in the FEM formulation of the problem being added
4. modify the contents of how the boundary conditions are parsed and set in `bc.c`
5. re-run `autogen.sh`, `./configure` and `make` to get a FeenoX executable with support for the new PDE.

The addition of non-trivial PDEs is not straightforward, but possible. The programming guide contains further details about how to contribute to the code base.

Conclusions

FeenoX's main goal is to keep things simple as possible from the user's point of view without sacrificing flexibility. There exist other tools which are similar in functionality but differ in the way the problem is set up. For example, FeniCSx uses the Unified Form Language where the PDE being solved has to be written by the user in weak form (Alnæs et al., 2014). This approach is very flexible, but even simple problems end up with non-trivial input files so it does not fulfill the first requirement stated in the summary. As simple as it is, FeenoX is still pretty flexible. A proof of this fact is that its applications range from coupling neutronics with CFD in nuclear reactors (Vasconcelos et al., 2018) to providing a back end to web-based thermo-mechanical solvers.

References

Alnæs, M. S., Logg, A., Ølgaard, K. B., Rognes, M. E., & Wells, G. N. (2014). Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Trans. Math. Softw.*, *40*(2). https://doi.org/10.1145/2566630

Balay, S., Abhyankar, S., Adams, M. F., Benson, S., Brown, J., Brune, P., Buschelman, K., Constantinescu, E., Dalcin, L., Dener, A., Eijkhout, V., Faibussowitsch, J., Gropp, W. D., Hapla, V., Isaac, T., Jolivet, P., Karpeev, D., Kaushik, D., Knepley, M. G., … Zhang, J. (2023). *PETSc/TAO users manual* (ANL-21/39 - Revision 3.19). Argonne National Laboratory. https://doi.org/10.2172/1968587

Balay, S., Gropp, W. D., McInnes, L. C., & Smith, B. F. (1997). Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, & H. P. Langtangen (Eds.), *Modern software tools in scientific computing* (pp. 163–202). Birkhäuser Press. https://doi.org/10.1007/978-1-4612-1986-6_8

Finite Element Methods & Standards (Great Britain), N. A. for. (1990). *NAFEMS: The standard NAFEMS benchmarks*. NAFEMS. https://books.google.com.ar/books?id=--qwHAAACAAJ

Hernandez, V., Roman, J. E., & Vidal, V. (2005). SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Trans. Math. Software*, *31*(3), 351–362. https://doi.org/10.1145/1089014.1089019

Roman, J. E., Campos, C., Dalcin, L., Romero, E., & Tomas, A. (2023). *SLEPc users manual* (DSIC-II/24/02 - Revision 3.19). D. Sistemes Informàtics i Computació, Universitat Politècnica de València.

Vasconcelos, V., Santos, A., Campolina, D., Theler, G., & Pereira, C. (2018). Coupled unstructured fine-mesh neutronics and thermal-hydraulics methodology using open software: A proof-of-concept. *Annals of Nuclear Energy*, *115*, 173–185. https://doi.org/10.1016/j.anucene.2018.01.021