Supporting complex simulations with open source finite element software

Wolfgang Bangerth
Colorado State University

In collaboration with many many others around the world.
Simulation often uses the *finite element method*. Essentially every manufactured object today goes through the following workflow:
Simulation:

*Computationally predict the physical response to external stimuli of interest.*

Typical areas:
- Solid mechanics (statics + dynamics)
- Fluid dynamics
- Electrodynamics
Available tools

Many commercial packages for “common” problems:

- Fluent
- Abaqus
- Comsol
- ...

- Covers most problems of “traditional engineering”
- Provide GUIs and integration in typical workflows

- Generally does it well and reliably, though:
  - does not use modern mathematical methods
  - scales poorly to parallel computers

Essentially no open source software in this arena.
Available tools

Large collection of open source software libraries:

- deal.II
- libmesh
- FEniCS
- ...

- Serve as the basis for
  - method development
  - solving “non-standard” problems

- The biggest of these libraries are
  - high quality
  - provide modern mathematical methods
  - some scale very well to parallel computers

Generally no GUIs; integration in typical workflows via external interfaces.
Available tools

One example: deal.II \( (https://www.dealii.org) \)

- 100s to 1000s of users
- Used in 1,400+ scientific publications we know of

- 1.4M lines of C++
- One major release per year:
  - 30-50 contributors to each release
  - 5-10 pull requests per day, every day

- Big focus on documentation:
  - 1000s of pages of doxygen-generated HTML
  - ~70 tutorial programs
  - 68 video lectures
  - short courses around the world

- Runs efficiently from laptops \( \rightarrow \) 300,000 processor cores.
Example applications: Aortic stents

Patient-specific fluid-structure interaction with realistic material models for vascular modeling
Example applications: Biomorphic growth

Morphoelastic development of mollusk shells
Example applications: Microscopic antennae

Homogenization of models for plasmonic crystals
Example applications: Complex models

Micromorphic models for the flexoelectric effect
A “typical” application in computational mechanics

What we generally need:

- Non-trivial 2d/3d geometries
- Coupled system of nonlinear PDEs
- Efficient non-linear iteration strategy
- Efficient linear solver
- Ways to visualize the solution

What we may need:

- Parallel execution on large systems
- Mixed or higher order finite elements
- Combination of meshes (overlapping domains, micro/macros)
- …
A “typical” application in computational mechanics

Question:

How can we write such a code?

Surely, it will take 10,000s–100,000s of lines of code!
(Recall: 20k lines of code per man-year.)
deal.\textsc{II}

A library for finite element computations that supports...

...a large variety of PDE applications tailored to non-experts.
Goals for this library:

- Supports complex computations in many fields
- Is general (not area-specific)
- Has fully adaptive, dynamically changing 3d meshes
- Scales to 10,000s of processors
- Is efficient on today's multicore machines

Fundamental premise:
Provide building blocks that can be used in many different ways, not a rigid framework.
deal.II provides:

- Adaptive meshes in 1d, 2d, and 3d
- Interfaces to all major graphics programs
- Standard refinement indicators built in
- Many standard finite element types (continuous, discontinuous, mixed, Raviart-Thomas, ...)
- Low and high order elements
- Support for multi-component problems
- Its own sub-library for dense + sparse linear algebra
- Interfaces to PETSC, Trilinos, UMFPACK, ARPACK, ...
- Supports SMP + cluster systems
deal.II

Status today:

- ~1000 downloads per month
- 1.4M lines of C++ code
- 10,000+ pages of documentation
- Portable build environment
- Used in teaching at many universities

- ~250 people have contributed to it
- ~40 people contribute to each release
- ~10 pull requests merged each day
Publications using deal.II:
Examples

Examples of what can be done with deal.II (2013 only):

- Biomedical imaging
- Brain biomechanics
- E-M brain stimulation
- Microfluidics
- Oil reservoir flow
- Fuel cells
- Transonic aerodynamics
- Foam modeling
- Fluid-structure interactions
- Atmospheric sciences
- Quantum mechanics
- Neutron transport
- Nuclear reactor modeling
- Numerical methods research

- Fracture mechanics
- Damage models
- Solidification of alloys
- Laser hardening of steel
- Glacier mechanics
- Plasticity
- Contact/lubrication models
- Electronic structure
- Photonic crystals
- Financial modeling
- Chemically reactive flow
- Flow in the Earth mantle
What makes such projects successful?

General observations:

Success or failure of scientific software projects is not decided on technical merit alone.

The *true* factors are beyond the code! It is not enough to be a good programmer!

In particular, what counts:

- Utility and quality
- Documentation
- Community

All of the big libraries provide this for their users.
Utility + quality

How deal.II makes itself easy to use:

● Lots of error checking in the code
● Extensive testsuites
● Meaningful error messages and assertions rather than cryptic error codes
● Cataloged use cases
● FAQs
● Well documented examples of debugging common problems
How we teach using deal.II:

- Installation instructions/README
- Within-function comments
- Function interface documentation
- Class-level documentation
- Module-level documentation
- Worked “tutorial” programs
- Recorded, interactive demonstrations

**Example:** deal.II has 10,000+ HTML pages. 170,000 lines of code are actually documentation (~10 man years of work). There are 67 recorded video lectures on YouTube.
deal.II comes with ~70 tutorial programs:

- From small Laplace solvers (~100s of lines)
- To medium-sized applications (~1000s of lines)
- Intent:
  - teach deal.II
  - teach advanced numerical methods
  - teach software development skills
There are also a number of large applications built on deal.II:

- **Aspect**: Advanced Solver for Problems in Earth Convection
  - ~140,000 lines of code
  - Open source: [http://aspect.geodynamics.org](http://aspect.geodynamics.org)

- **OpenFCST**: A fuel cell simulation package
  - Supported by an industrial consortium
  - Open source: [http://www.openfcst.org/](http://www.openfcst.org/)

- **DFT-FE**: A density functional theory code
  - Open source: [https://sites.google.com/umich.edu/dftfe](https://sites.google.com/umich.edu/dftfe)

- ...
How much work does it take?

Use case: Grad student with 3 years for research

- Solve a complex model
- With realistic geometries, unstructured meshes
- Higher order finite elements
- Multigrid-based solver
- Parallelization
- Output in formats for high-quality graphics
- Results almost from the beginning: a wide variety of tutorials allow a gentle start
- There are ways to share your codes with others
How much work does it take?

Use case: Expert user for a commercial project, 2 weeks of full-time work

- Complex model
- Realistic geometries, unstructured meshes
- Higher order finite elements
- Simple solver
- Parallelization
- Output in formats for high-quality graphics
- Validated against another code/experimental results
Effects

What this development model means for users:

- We can solve problems that were previously intractable
- Methods developers can demonstrate applicability
- Applications scientists can use state of the art methods

- Our codes become far smaller:
  - less potential for error
  - less need for documentation
  - lower hurdle for “reproducible” research (publishing the code along with the paper)

- More impact/more citations when publishing one's code
Deal.II is a library that supports building finite element codes:

- Widely used
- High quality, professionally developed
- Allows building codes much faster, much better
- Used to solve complex, more realistic problems
- Scales far better than almost all commercial software