Towards an Eilmer Four Point Zero Release

Rowan Gollan, Peter Jacobs, Kyle Damm, Nicholas Gibbons, Daryl Bond

21 January 2021

Setting the scene

- 2014: First lines of code for Eilmer4 begun
- 2015: Internal (to UQ) use began; Kyle Damm began PhD project with Eilmer4 base
- 2015: first conference presentation about Eilmer4 (PAJ)
- 2016: New users strongly encouraged to use Eilmer4 instead of Eilmer3; phase out PhD student use, new students advised to use Eilmer4
- 2017: Eilmer4 first used in CFD class (MECH4480)
- 2017: External collaborators begin using Eilmer4
- 2019: First eilmer users meeting (monthly meeting for users and developers to share experience reports and heads up on upcoming developments)
- 2020: Development team increases more than doubles: Peter J, Daryl B, Kyle D, Nick G and myself.
- 2020: Dev team introduce continuous integration tool to aid quality control (thanks Nick)
- 2020: Eilmer user meetings expand to global reach (thanks to Zoom)

A fixed release strategy for Eilmer

- what is a fixed release
- why consider a fixed release for Eilmer
- what will be included in an Eilmer 4.0 release and why
- progress to date and outlook

Modes of software release

continuous release users access the software directly from the developers' repository;
each time the user updates they get the latest-n-(hopefully)-greatest version

fixed release users download a specific tagged version of the software source; tagged versions lag development versions but are stable and supported

- both modes of release can exist in the one project; we propose to do both in Eilmer
- continuous release is our only distribution mode presently
- but, we think there is now enough benefit to providing fixed releases that it is worth the effort on the developers part

The purpose of a fixed release for the Eilmer community

For the users:

- gives confidence about the stability/maturity of code features
- sends a signal about the level of support to expect
- gives a consistency of experience: input/output, run-time behvaiour, guarantee of backwards compatibility

For the developers:

- sets quality/maturity targets for features in the code
- raises the barrier for introduction of “breaking changes”
- sets priorities for bug fixes
- helps contain the time and effort on user support and responding to (possible) bugs

What's in a new release?

Users can expect:

- fixes to old bugs
- addition of new features
- new configuration options, or slight changes to meanings of old options
- incompatible changes could possibly be introduced, eg. data formats changed
- (addition of new bugs)

We can use a release numbering system to convey information about the nature and amount of change between releases.

Proposed release numbering for Eilmer series

patch number trivial changes; bug fixes or very small enhancements to features; no new features; changes are backwards compatible

minor number new features (probably isolated features) introduced; backwards compatible; use even/odd numbers to indicate stable/experimental release

major number marks boundaries of compatibility; expected to have new features/sets of features

This style of release numbering is known as semantic versioning.

… and we're starting at 4.0

- Eilmer is already widely used in production runs, so we're well past that point in history for 1.0 release
- there is a history of earlier incarnations that led to the D-language version of Eilmer, so 4.0 captures that legacy
- it also signals a degree of maturity and longevity (marketing-driven decision)
- ultimately, the starting number is arbitrary. It's consistency in its use going forward that is important

---

Examples when marketing influenced version numbers:

- Windows 7 is actually Windows 6.1 internally for the developers
- In 1999, Slackware Linux jumped from v4 to v7. Its version numbers were lagging other distros and there was a user perception that it was out of date.

What's in the Eilmer 4.0 release (1/2)

Core simulation loop

- Euler
- predictor-corrector
- RK3
- local time stepping (LTS)
- block marching

Dimensionality, reference frames and grid types

- 2D and 3D
- fixed and rotating frames
- structured and unstructured grids

Flow physics

- inviscid
- viscous
- mass diffusion (Fickian)

Gas modelling and kinetics

- ideal gas
- thermally perfect gas mixtures
- generalised finite-rate chemistry
- equilibrium gas
- two-temperature air and nitrogen
- vibrationally-specific nitrogen gas

Moving grid

- user-defined grid motion
- shock-fitting mode

Turbulence models

- $k - \omega$ model
  - with wall functions

What's in the Eilmer 4.0 release (2/2)

Conjugate coupling

- fluid/solid domains in 2D and 3D

Boundary conditions

- all should work in the context of supported gas models
- catalytic BCs included

User-defined functions for simulation customisation Available for:

- boundary conditions
- source terms
- grid motion

Parallelism

- shared-memory
- MPI (should work with all other features)

Auxiliary tools

- grid partitioner
- file compactor
- snapshot handler
- monitor program

Documentation

- User guides
- Theory book
- Catalogue of examples
- Detailed notes on examples
- FAQ
- website

What's not in the Eilmer 4.0 release

Everything not listed above should be considered at the experimental stage of development. This includes:

- accelerators to steady-state
  - Newton-Krylov update
  - LU-SGS
  - STS
- features using hardware accelerators (eg. GPU chemistry)
- adjoint solver
- S-A turbulence model
- numerics associated with LES/DNS

In some cases, the implementations are mature but we want to gather more experience with the features before including them in a supported release.

Progress to date on 4.0 release (1/2)

Feature	Status
explicit time-stepping	complete
block marching	documentation needed
2D/3D capability	complete
rotating frames	example needed
structured and unstructured grids	complete
inviscid/viscous flows	complete
mass diffusion	documentation and examples needed
ideal gas	complete
thermally perfect gas	complete
finite-rate chemistry	complete
equilibrium gas	complete
2-T air & N2	complete
vib. specific N2	complete
user-defined grid motion	more examples needed
shock-fitting mode	implementation review needed

Progress to date on 4.0 release (2/2)

Feature	Status
k-omega model	complete
wall functions	more documentation needed
conjugate coupling	documentation needed
catalytic BCs	documentation & examples needed
user-defined functions	always needs more documentation
shared-memory	complete
MPI	complete
grid partitioner	documentation needed
file compactor	complete
snapshot handler	documentation in user-guide needed
monitor program	complete
user guides	mostly complete
theory book	needs to be completed
catalogue of examples	quite sparse still
detailed notes for examples	almost non-existent
FAQ	only a beginning stages
website	in good shape

Tour of documentation and website

Website

- landing page for new and returning users
- central pointer to all resources
- project advertising

User Guides

- Written with new user in mind
- Fairly descriptive giving both how and why
- Report format released as PDF

Reference Manuals

- Serves as a reference for experienced users
- Terse but comprehensive description of all code options and configurations
- HTML format hosted online

Catalogue of examples, Detailed notes on examples, FAQ

Future release plans

- 4.0 coming soon: first half of 2021
- 4.2: steady-state accelerator; S-A model
- 4.4: adjoint solver; numerics for LES/DNS
- 4.6: hardware accelerators release (?)

Summary

- **Why fixed releases?** as a management tool and marketing tool
- **Which release mode should I choose?**
  - choose a fixed release if you use Eilmer infrequently and are adequately served by the stable features
  - choose continous release if you are a power user and need bleeding edge features; *be prepared to lend your support to the development team*
- **When?** soon