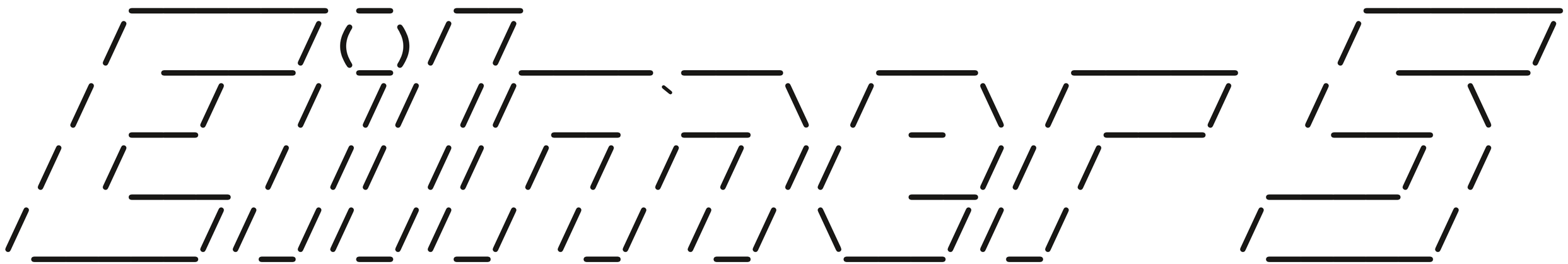
A Preview of ....



Rowan Gollan

Centre for Hypersonics

School of Mechanical & Mining Engineering

The University of Queensland, Australia

Thursday 28 September 2023

Why an Eilmer5? Why now?

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

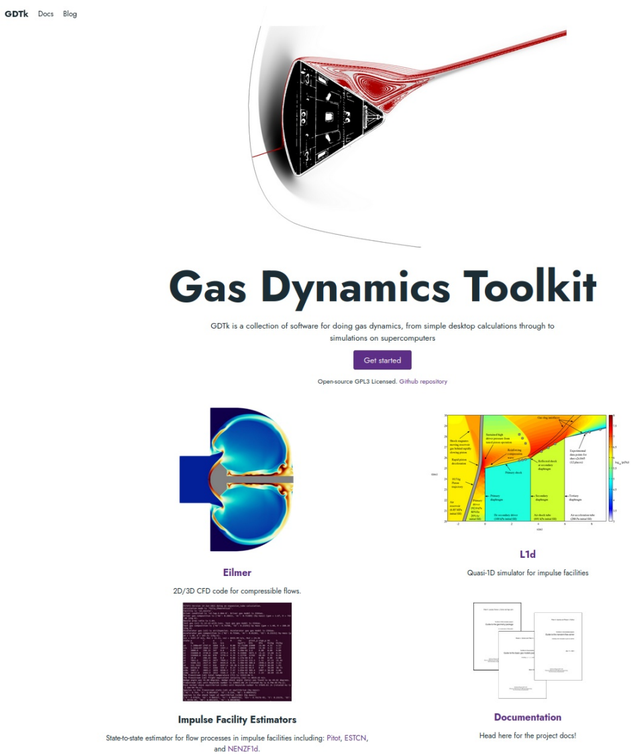
minor number new features (probably isolated features) introduced; backwards compatible; use even/odd numbers to indicate stable/experimental release

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

This style of release numbering is known as semantic versioning.

No backwards compatibility in output files between Eilmer4 and Eilmer5, hence major number bump.

Some recent history



May 2020: new website launch

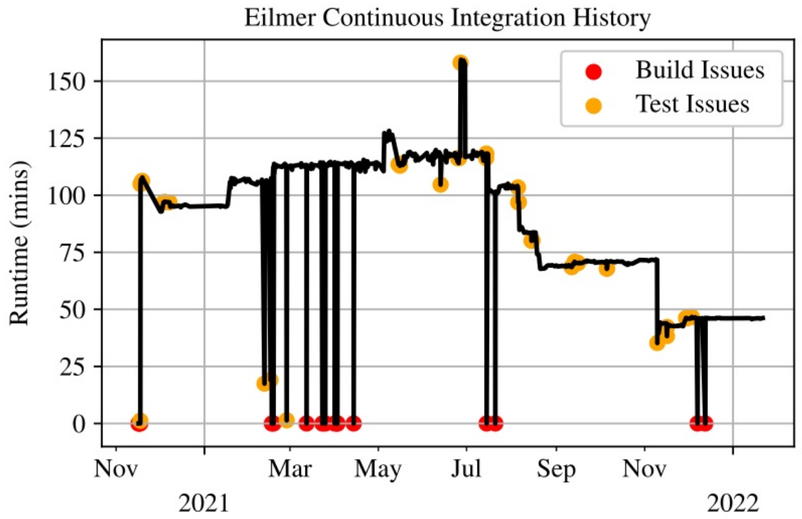
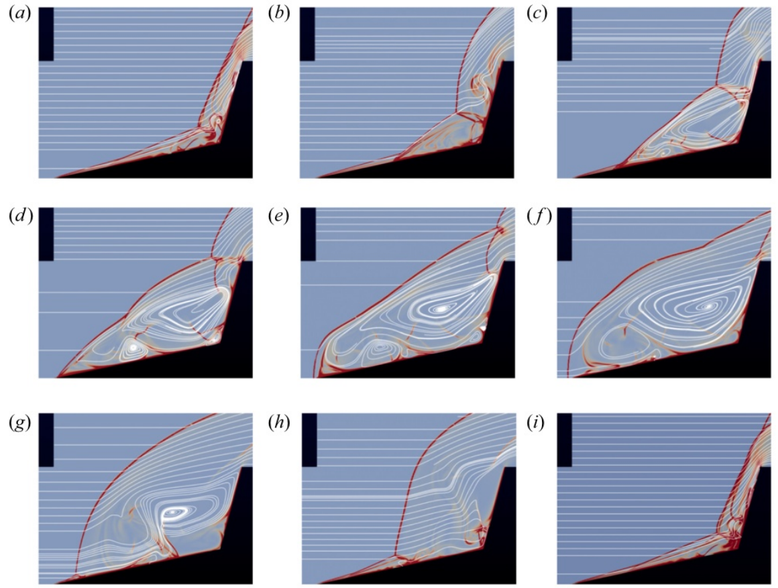


Fig. 11. Runtime history of Eilmer continuous testing system.

Dec 2020: continuous integration testing goes live

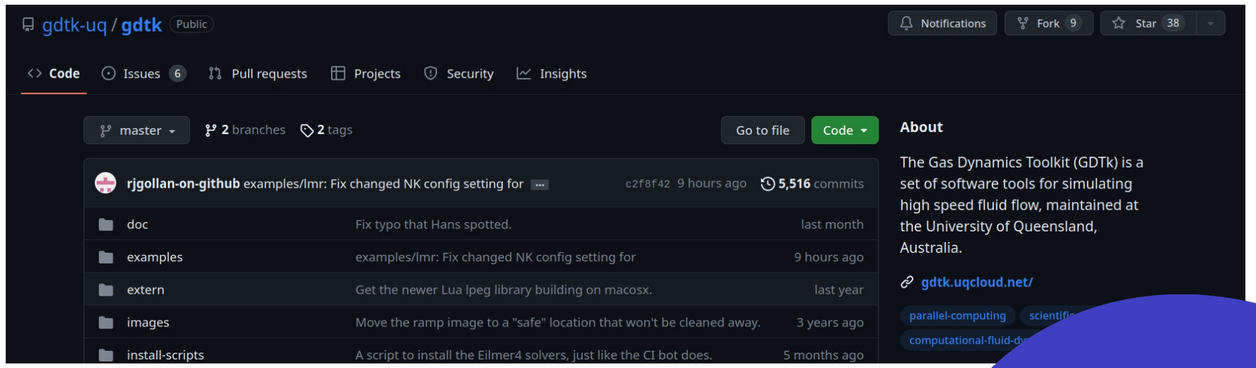
Source: *Gibbons et al. (2023), CPC 282:108551*



April 2021: largest numerical experiment conducted with Eilmer; 300+ simulations; 500 GB data

Source: *Hornung et al. (2021), JFM 916(A5)*

Some recent history



Feb 2022: repository moved to Github

GDTk CHANGELOG 2022-q4

A newsletter for the GDTk Community

23 October 2022

Welcome to the inaugural newsletter for the GDTk community! There's been a lot happening in the past few months and it seemed the timing was right to record and broadcast those happenings in an informal newsletter. Our intent is to release on a quarterly basis. The newsletter will be principally written and compiled by the developers, but we will also readily welcome contributions from the community. Without further ado, read on to find out what's been happening lately amongst the group of users.

Since this is our first newsletter, I'll take the liberty of a loose definition for the period of a quarter. Let's share some of the activities going back half a year or so. We've had some exchanges and visits, some serious paper writing and some even more serious code writing resulting in a program called `Chicken`.

Eilmer in the archive

The dev team has recently published a paper in the archival literature on the D language version of `Eilmer` (colloquially called `Eilmer 4`). The paper appears in *Computer Physics Communications*.[1] For a limited time, we've been given a link to share the work freely.[2] After that time, you can likely get to the paper through your institution's access to Elsevier journals.

`Eilmer` is in the archive in more ways than one now. *Computer Physics Communications* are serious about the reproducibility of results derived from simulation programs, so they archive the source code as well. As part of the publication process, Nick Gibbons had to ensure that the code built, installed and ran as advertised. This was done in a container environment that the article reviewers were able to inspect. The source code now has its own DOI

`https://doi.org/10.17632/gy2ds2fyxm.1`

and a location on Code Ocean

`https://codeocean.com/capsule/7226427`.

The paper covers the formulation and numerics in `Eilmer`, our development process, and gives several examples of applications. I wrote more about the story behind the paper on our GDTk blog.

`https://gdtk.uqcloud.net/blog/the-paper-on-eilmer-v4-is-out.-go-read-it`

Hacking at Chickens

Putting aside the clickbait-inpsired gore of the title, what we're talking about here is writing code (hacking) for a new flow solver called `Chicken`.

We're really excited that the team has been selected to attend a GPU Hackathon in Canberra, hosted by National Compute Infrastructure Australia. The team will get to work with NVIDIA

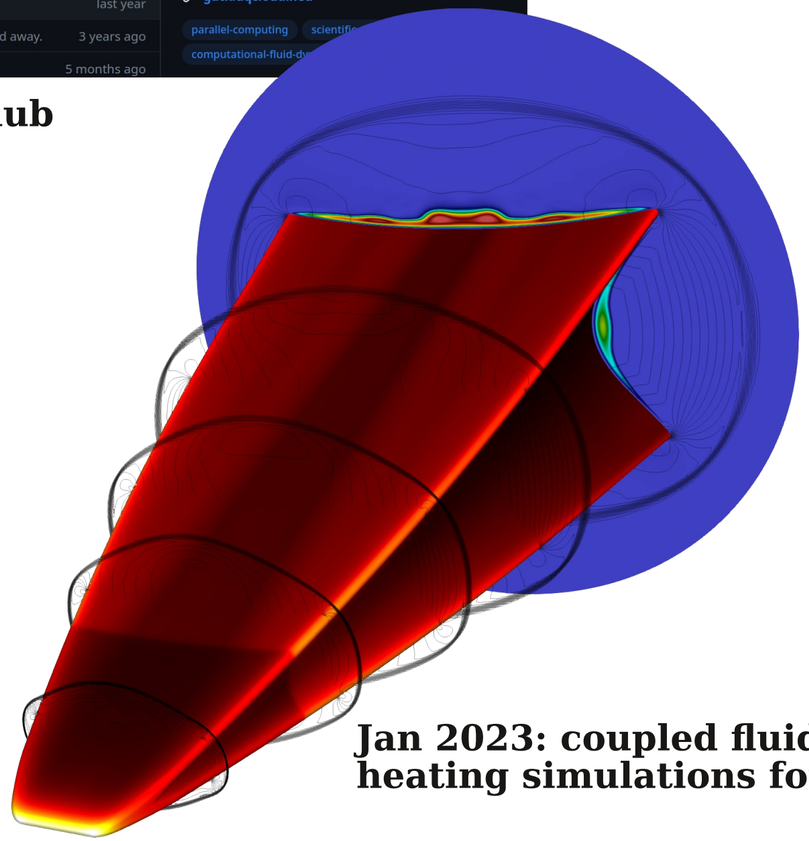
Figure 1: The `Eilmer` paper is available in CPC volume 282.

[1] N. N. Gibbons, K. A. Damm, P. A. Jacobs, and R. J. Gollan. Eilmer: an open-source multi-physics hypersonic flow solver. *Computer Physics Communications*, 282(108551), 2023. DOI: 10.1016/j.cpc.2022.108551

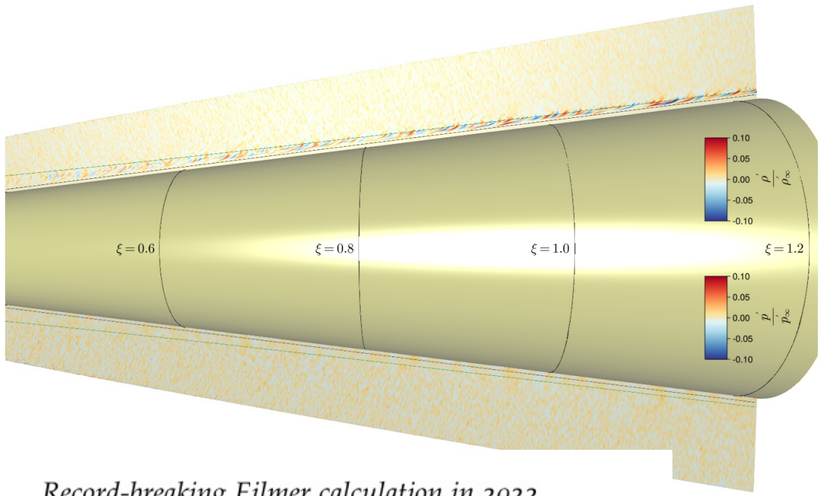
[2] `https://authors.elsevier.com/c/1fsPB2OInkAiF`

Figure 2: A postcard for Easter time by Wally Fialkowska with hens pulling a cart, first sold in 1914. `https://tuck@postcards.org/items/35560`

Oct 2022: Newsletter launch



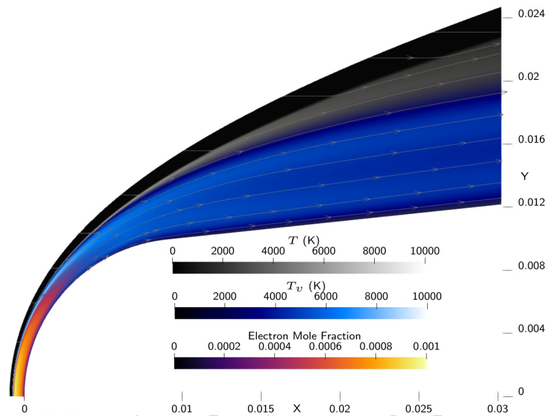
Jan 2023: coupled fluid/solid heating simulations for BoLT-II



Record-breaking Eilmer calculation in 2022

contributions by Lachlan Whyborn

In 2022, a new record for largest Eilmer simulation was set; largest in terms of number of cells and compute hours. As part of Lachlan Whyborn's PhD, he simulated high-speed flow over a 3D slice (of



Sep 2023: simulations of electron transpiration cooling effectiveness

Some recent history

Computer Physics Communications 282 (2023) 108551

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

Eilmer: An open-source multi-physics hypersonic flow solver ☆,☆☆

Nicholas N. Gibbons, Kyle A. Damm, Peter A. Jacobs, Rowan J. Gollan *

Centre for Hypersonics, School of Mechanical & Mining Engineering, The University of Queensland, Australia

ARTICLE INFO

Article history:
Received 23 May 2022
Received in revised form 10 September 2022
Accepted 19 September 2022
Available online 27 September 2022

Keywords:
Scientific computing
Computational fluid dynamics
Hypersonics
Parallel computing

ABSTRACT

This paper introduces Eilmer, a general-purpose open-source compressible flow solver developed at the University of Queensland, designed to support research calculations in hypersonics and high-speed aerothermodynamics. Eilmer has a broad userbase in several university research groups and a wide range of capabilities, which are documented on the project's website, in the accompanying reference manuals, and in an extensive catalogue of example simulations. The first part of this paper describes the formulation of the code: the equations, physical models, and numerical methods that are used in a basic fluid dynamics simulation, as well as a handful of optional multi-physics models that are commonly added on to do calculations of hypersonic flow. The second section describes the processes used to develop and maintain the code, documenting our adherence to good programming practice and endorsing certain techniques that seem to be particularly helpful for scientific codes. The final section describes a half-dozen example simulations that span the range of Eilmer's capabilities, each consisting of some sample results and a short explanation of the problem being solved, which together will hopefully assist new users in beginning to use Eilmer in their own research projects.

Program summary
Program Title: Eilmer
CPC Library link to program files: `https://doi.org/10.17632/gy2ds2fyxm.1`
Developer's repository link: `https://github.com/gdtk-uq/gdtk`
Code Ocean capsule: `https://codeocean.com/capsule/7226427`
Licensing provisions: GPLv3
Programming language: D, Lua
Supplementary material: `https://gdtk.uqcloud.net`
Nature of problem: Eilmer solves the compressible Navier-Stokes equations with a particular emphasis on flows at hypersonic speeds. The code includes modelling for high-temperature gas effects such as chemical and vibrational nonequilibrium. Eilmer can be used for the simulation for unsteady and steady flows.
Solution method: The code is implemented in D [1] and built on a finite-volume formulation that is capable of solving the Navier-Stokes equations in 2D and 3D computational domains, discretised with structured or unstructured grids. Grids may be generated using a built-in parametric scripting tool or imported from commercial gridding software. The inviscid fluxes are computed using the reconstruction-evolution approach. In structured-grid mode, reconstruction stencils up to fourth-order spatial accuracy are available. In unstructured-grid mode, least-squares reconstruction provides second-order spatial accuracy. A variety of flux calculators are available in the code. Viscous fluxes are computed with compact stencils with second-order spatial accuracy. For unsteady flows, explicit time-stepping with low-order RK-family schemes are available, along with a point-implicit Backward-Euler update scheme for stiff systems of equations. For steady flows, convergence can be greatly accelerated using a Jacobian-free Newton-Krylov update scheme, which seeks a global minimum in the residuals using a series of large pseudo-timesteps. Domain decomposition is used for parallel execution using both shared memory and distributed memory programming techniques.
Additional comments including restrictions and unusual features: Eilmer provides a programmable interface for pre-processing, post-processing and user run-time customisations. The programmable interface

☆ The review of this paper was arranged by Prof. Hazel Andrew.
☆☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
* Corresponding author.
E-mail address: r.gollan@uq.edu.au (R.J. Gollan).

https://doi.org/10.1016/j.cpc.2022.108551

Jan 2023: The (new) Eilmer paper

Eilmer5: a primary motivator

The Newton-Krylov steady-state accelerator is ready for prime time

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

Application of a Jacobian-Free Newton-Krylov Method to the Simulation of Hypersonic Flows

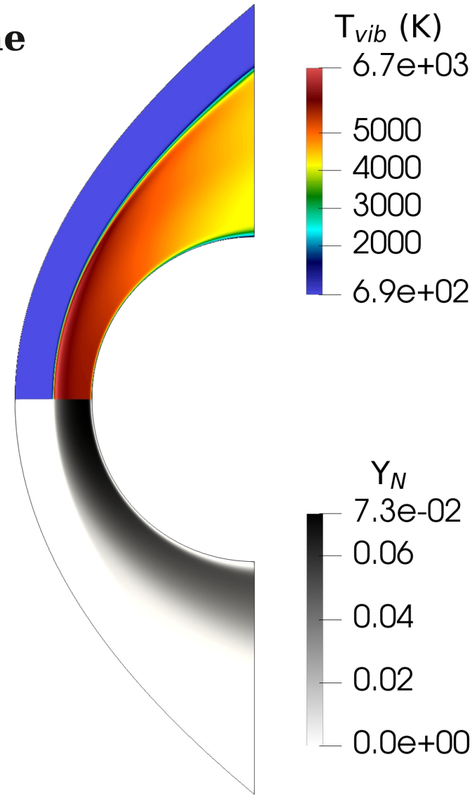
Kyle Damm, Nicholas Gibbons, Peter Jacobs and Rowan Gollan

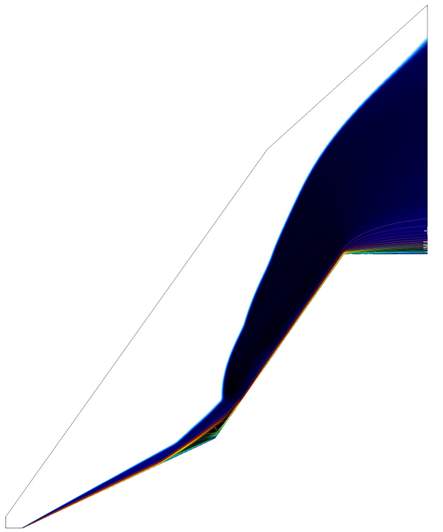
Centre for Hypersonics

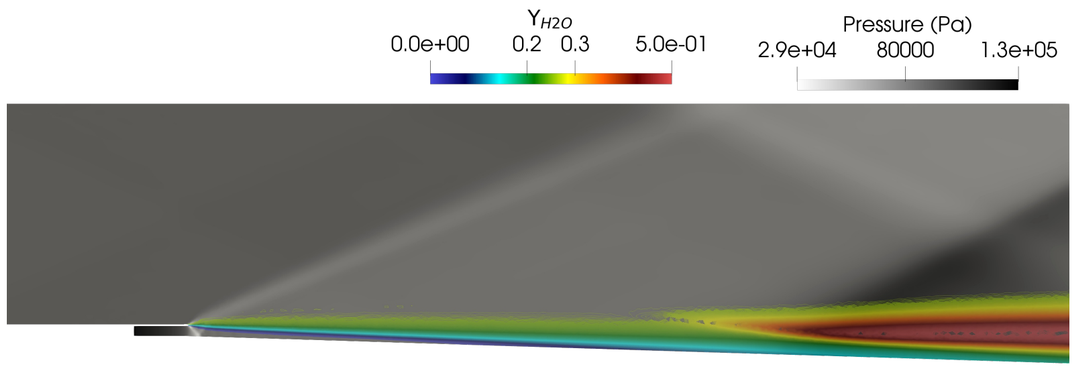
School of Mechanical & Mining Engineering

The University of Queensland, Australia

AIAA SciTech Forum, National Harbor MD, 23 -- 27 January 2023







`lmr5`: development migration path

- presently: **majority** of Eilmer5 source code **is** Eilmer4 source code
- when ready: migrate all Eilmer4 source across to Eilmer5 for active development
- Eilmer4 then goes into maintenance mode

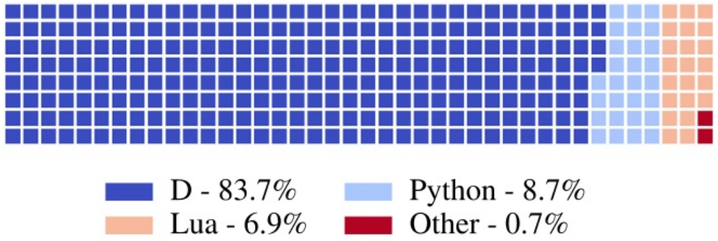


Fig. 4. Division of languages within Eilmer as of April 2022.

Source: *Gibbons et al. (2023), CPC 282:108551*

Eilmer4:

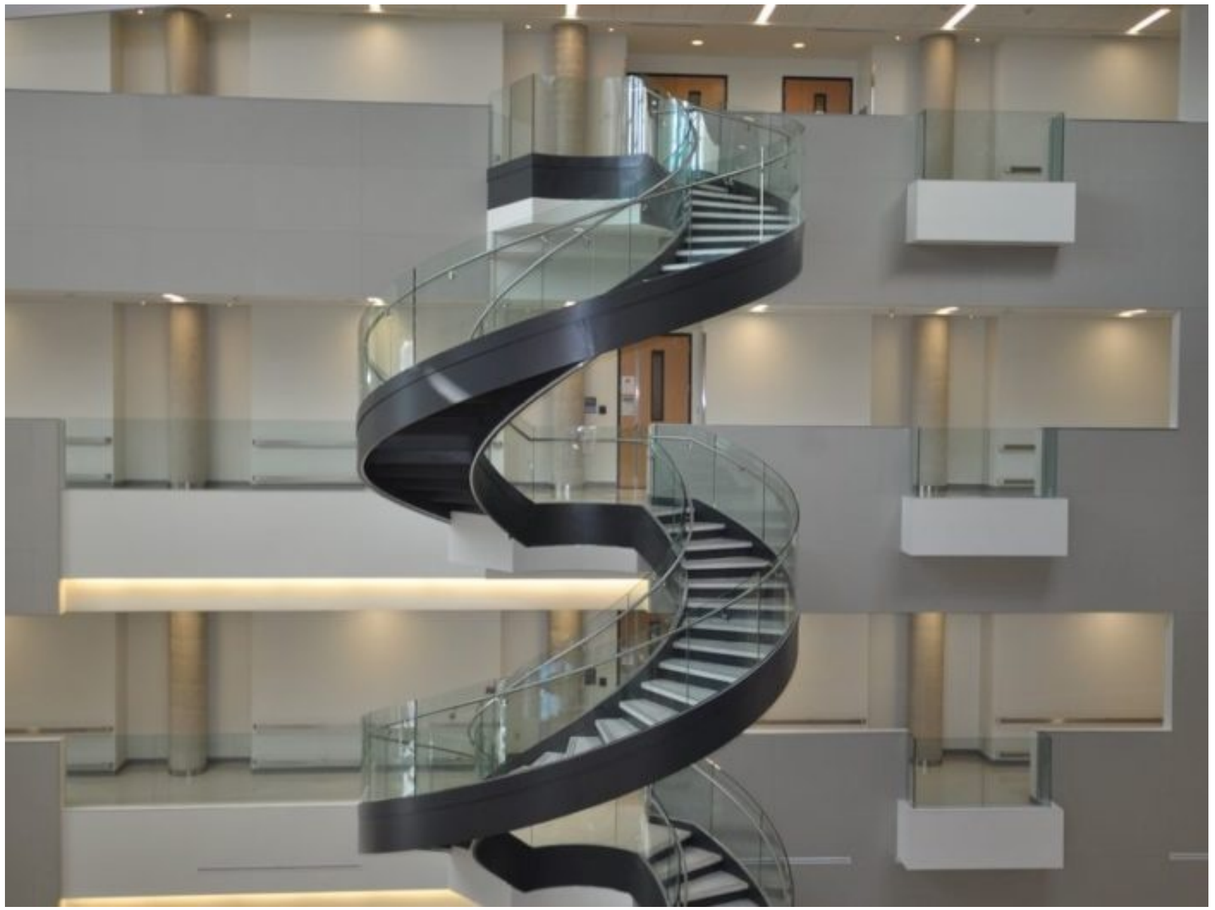
Language	files	blank	comment	code
D	106	4586	11228	61439
Lua	21	380	830	5981
Python	6	148	248	1181
make	1	100	36	1008
HTML	2	120	67	666
Ruby	4	44	61	448
Tcl/Tk	1	56	66	374
OpenCL	1	11	0	172
CUDA	1	5	1	137
Other	4	13	3	58
SUM:	147	5463	12540	71464

Eilmer5:

Language	files	blank	comment	code
D	24	1054	1988	7761
Lua	4	111	142	624
make	3	64	33	436
Python	1	25	53	213
Bourne Shell	1	1	0	3
SUM:	33	1255	2216	9037

`lmr5`: development approach

```
auto finished = false;
buildNewIOLayer();
while (!finished) {
    buildNewUI();
    refactorAlgorithms();
    performVerification();
    reworkExamples();
    writeDocumentation();
}
```



Source: *Chicago Metal Rolled Products*

`lmr5`: development approach

```
auto finished = false;
buildNewIOLayer();
while (!finished) {
    buildNewUI();
    refactorAlgorithms();
    performVerification();
    reworkExamples();
    writeDocumentation();
}
```

```
> git log --oneline | grep lmr | wc -l
```

85

Source: *Sculpture by Olafur Eliason*

lmr5: buildNewUI()

- git-like command-line interface: command + sub-commands structure
- anticipated many sub-commands, each doing "one thing well" (Unix design philosophy)
- commands are grouped: common, developer/diagnostics, meta

> lmr help -a

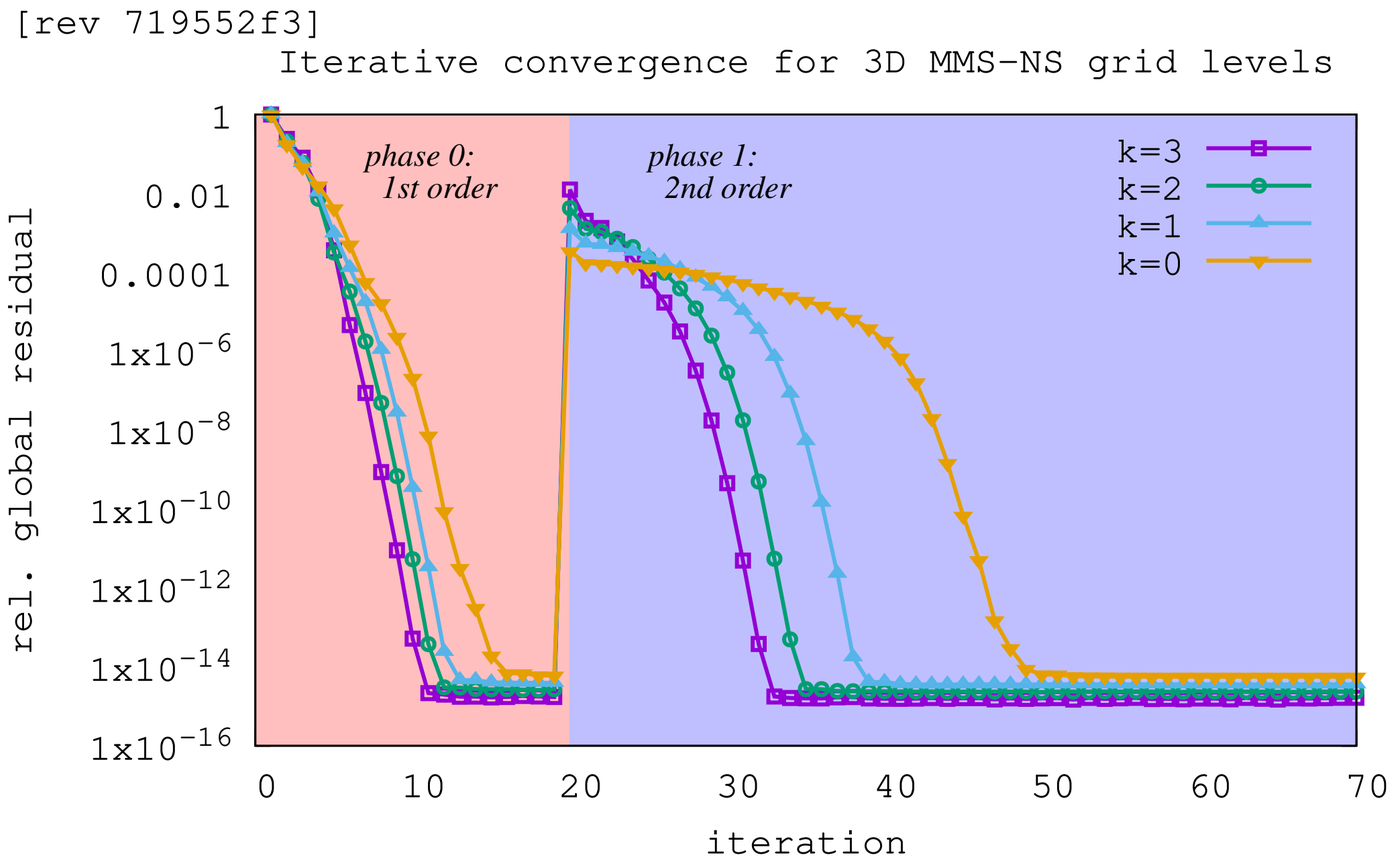
```
See 'lmr help <command>' to read about a specific subcommand.

Available commands
   compute-norms        Compute field norms (possibly with respect to a reference solution).
   help                 Display help about using Eilmer.
   limiter2vtk          Convert fields of limiter values to VTK format.
   prep-flow            Prepare initial flow fields for an Eilmer simulation.
   prep-grid            Prepare grids for an Eilmer simulation.
   prep-grids           Prepare grids for an Eilmer simulation.
   revision-id          Print version control revision ID.
   run-steady           Run a steady-state simulation with Eilmer.
   snapshot2vtk         Convert snapshots of flow fields to VTK format.

Developer/diagnostics commands
   check-jacobian       Check the formation of the Jacobian.

Meta commands
   version              Print condensed version information about lmr program.
   version-long         Print full version information about lmr program.
```

lmr5: refactorAlgorithms()

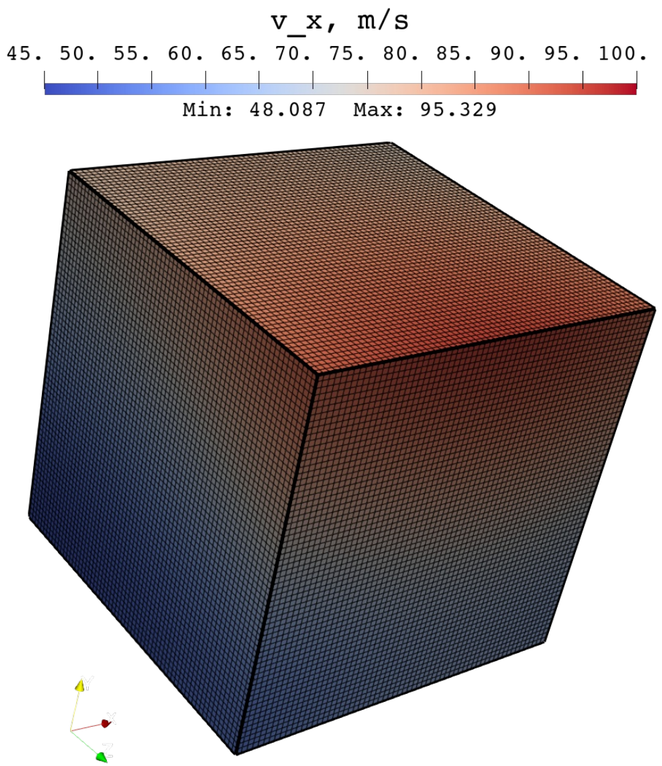


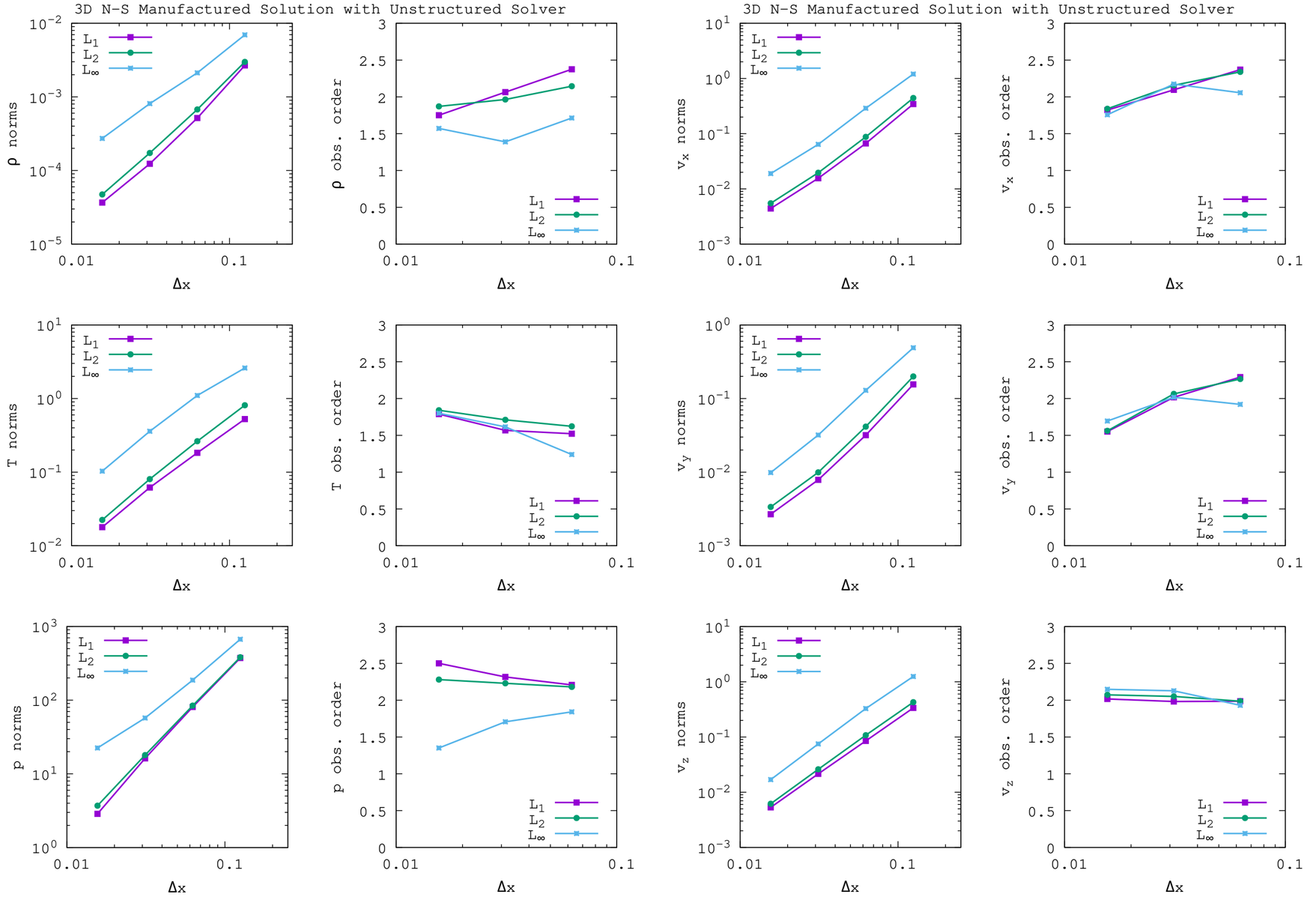

```
NewtonKrylovPhase:new{
   residual_interpolation_order = 1,
   jacobian_interpolation_order = 1,
   frozen_preconditioner = true,
   frozen_limiter_for_jacobian = false,
   use_adaptive_preconditioner = false,
   steps_between_preconditioner_update = 10,
   linear_solve_tolerance = 0.1,
   use_local_timestep = true,
   use_auto_cfl = true,
   threshold_relative_residual_for_cfl_growth = 0.9,
   start_cfl = 100.0,
   max_cfl = 1.0e6,
   auto_cfl_exponent = 1.0
}

NewtonKrylovPhase:new{
   residual_interpolation_order = 2,
   jacobian_interpolation_order = 2,
   start_cfl = 100.0,
}
```

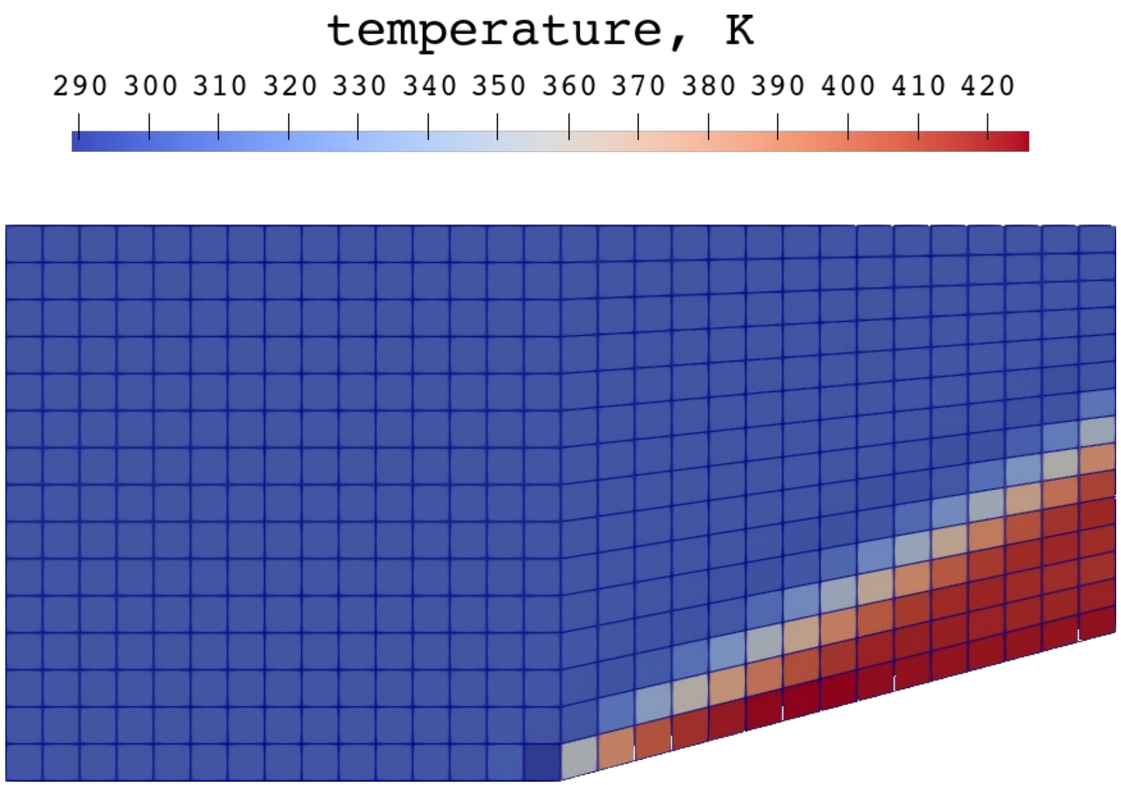
lmr5: performVerification()

```
> lmr-verify
```





lmr5: reworkExamples()



`Eilmer` (v5) Examples Listing

Rowan J. Gollan · Peter A. Jacobs · Kyle A. Damm · Nick N. Gibbons – 2023-07-05

Examples

Supersonic flow over a convex corner

`gdtk/examples/lmr/2D/convex-corner`

Kyle A. Damm & Rowan J. Gollan 2023-07-04

This example is of supersonic flow over a convex corner: a Prandtl-Meyer expansion fan. It is test case B.1 "Prandtl-Meyer Expansion Fan" in the paper by Ghia et al. (2010). This particular example was set up by Kyle Damm with conditions to match the example in Ghia et al.:

$$M_\infty = 2.0; \quad \delta = -10.0^\circ; \quad \gamma = 1.4$$

Supersonic flow over a wedge

`gdtk/examples/lmr/2D/wedge`

Kyle A. Damm & Rowan J. Gollan 2023-07-05

This example is of supersonic flow over a wedge: essentially, flow through an oblique shock. It is test case B.2 "Steady-state Oblique Shock Wave" in the paper by Ghia et al. (2010). This particular example was set up by Kyle Damm with conditions to match the example in Ghia et al.:

$$M_\infty = 3.0; \delta = 15.0^\circ; \gamma = 1.4$$

TIP

This example shows some advanced grid manipulation:

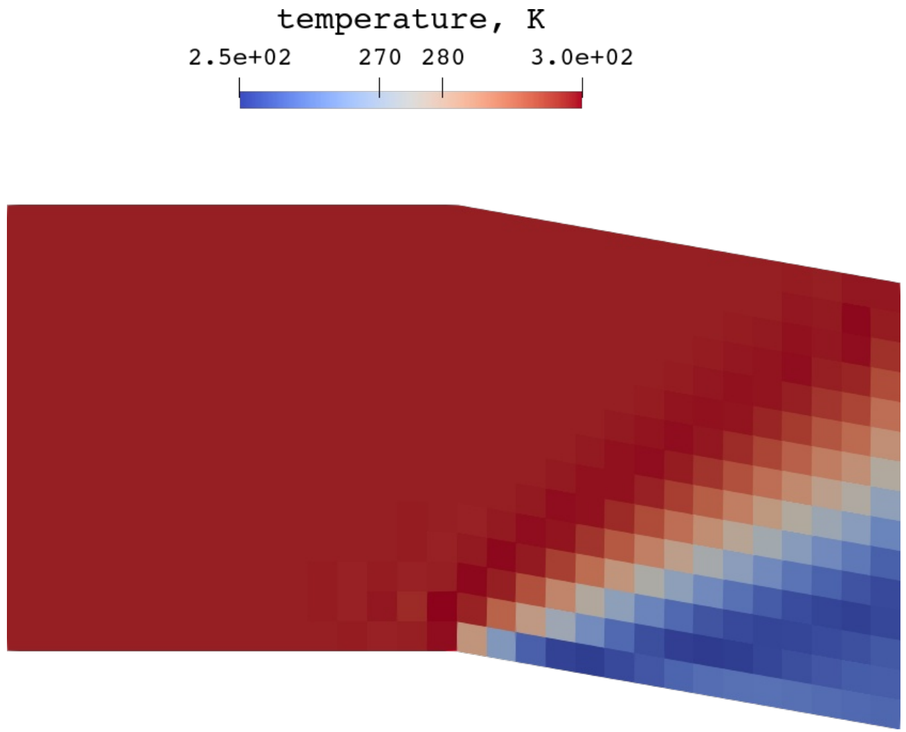
1. How to join `StructuredGrid` grids; and
2. How to form an `UnstructuredGrid` from a `StructuredGrid`.

lmr5: live demo

```
> lmr help
```

`lmr5`: summary of changes

- new command-line interface
- new file and directory layout
- staged preparation is default
- temporal settings/BCs only have meaning for transient mode solver
- common names for job files expected, eg. `job.lua`
- summary listing of examples
- tests coordinated with pytest
- aim for reproducibility in workflows: makefiles, `lmr revision-id`



`lmr5`: what it means for you

2022.Q4 -- 2023.Q3 : pre-alpha

- capabilities: single-species, structured/unstructured, parallel, turbulence models
- RJG developing
- RBO limited use

2023.Q4 : alpha

- capabilities: add multi-species and multi-temperature
- all principal developers move to Eilmer5
- gdtk core group start using

2024.Q1 : beta

- capabilities: feature complete with Eilmer4
- members of CfH invited to migrated to Eilmer5

2024.Q3 : tagged release on github

`> git log -n 1 32c20b27`

```
commit 32c20b27d221849e5f100e57d2627bcea8006de2
Author: Rowan J. Gollan <r.gollan@uq.edu.au>
Date:   Sat Oct 22 23:43:18 2022 +1000

    lmr5: first commit in public master

    Don't panic! v4 of Eilmer is not going away anytime soon.

    This commit brings the eilmer 5 work-in-progress code into
    the master branch on the public-facing github. This is so
    that I can work more closely with the developments in master.

    Eilmer v5 is a change at the interface level, but uses almost
    all of the core files from Eilmer v4. The migration path
    is that new code goes in:

    src/lmr

    but uses a lot of code from src/eilmer.

    There is an example of the new interface in:

    examples/lmr/2D/convex-corner
```