BOUT++ overview

Ben Dudson
BOUT++ workshop 9th January 2023

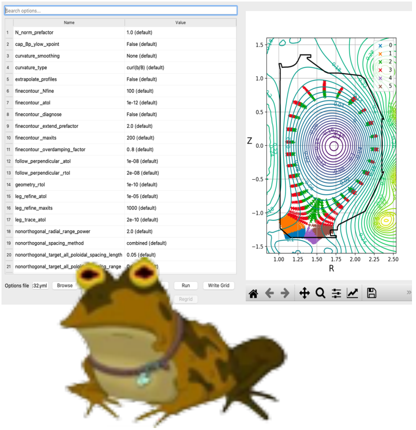
Thanks to contributors including: Peter Hill, Mike Kryjak, Joseph Parker, John Omotani, David Dickinson, Yining Qin, Steven Glenn, Xueqiao Xu, and the BOUT++ team

This work was performed in part under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344.
Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

BOUT++ is an ecosystem of plasma simulation tools

Pre-processing



Hypnotoad grid generator

Models

elm-pb (3,4,5,6-field)

Gyro-fluid models

Trans-neut

SD1D

Hermes

STORM

SOLT3D

BOUT++

- Meshing
- FD, FV methods
- Time integrators
- I/O
- MPI
- Testing

PETSc

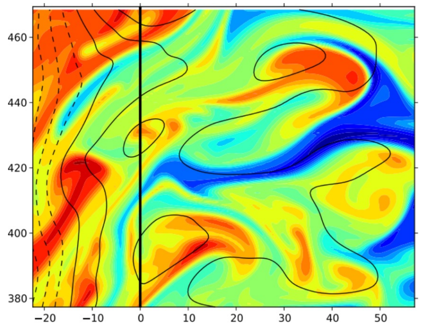
hypre

sundials

SLEPc

RAJA

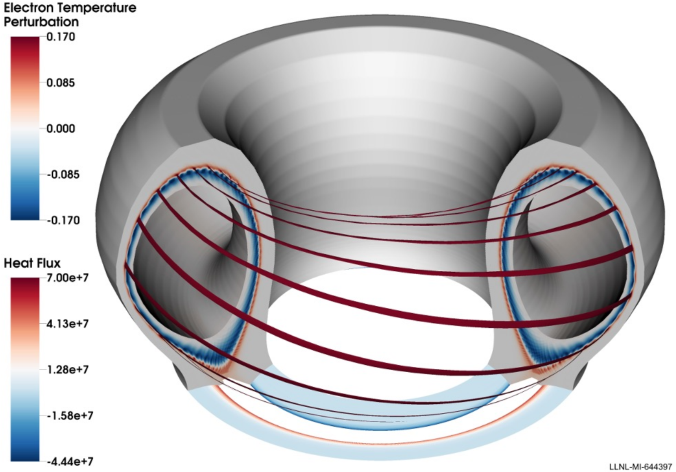
Post-processing



xarray

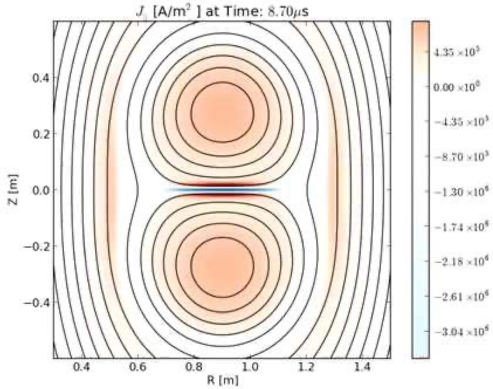
BOUT++ underpins many different models

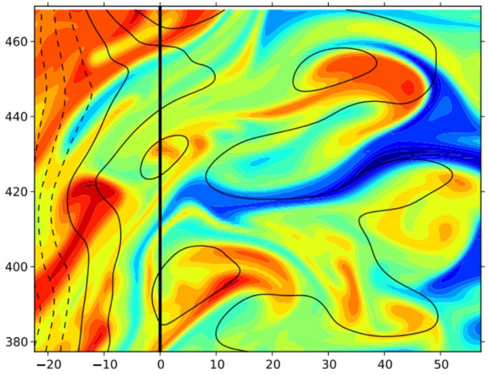
- Solves nonlinearly coupled hyperbolic, parabolic and elliptic equations
- MPI-parallelised, scales to ~4,000 cores, depending on problem size
- Turbulence ~10^6—10^8 unknowns, ~10^5



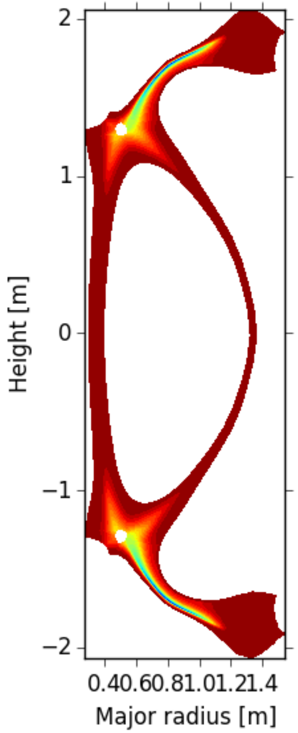
Edge Localised Modes (LLNL)

Magnetic reconnection





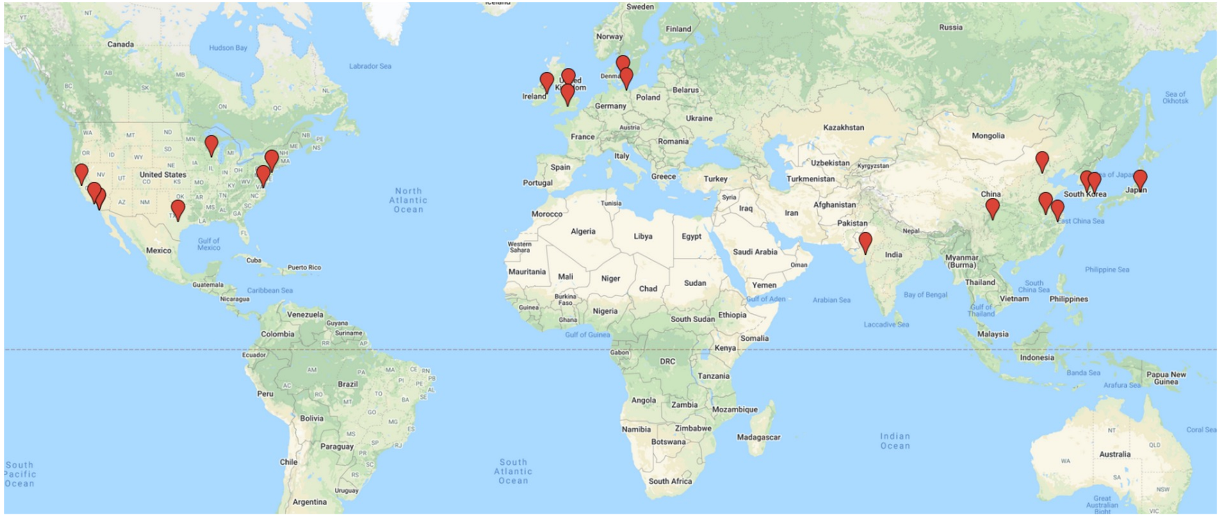
Plasma turbulence



Transport (MAST-U)

BOUT++ is open source

- Open source, users/developers worldwide
- Strong community, and investment in building capabilities to underpin research



<u>Top contributors</u>

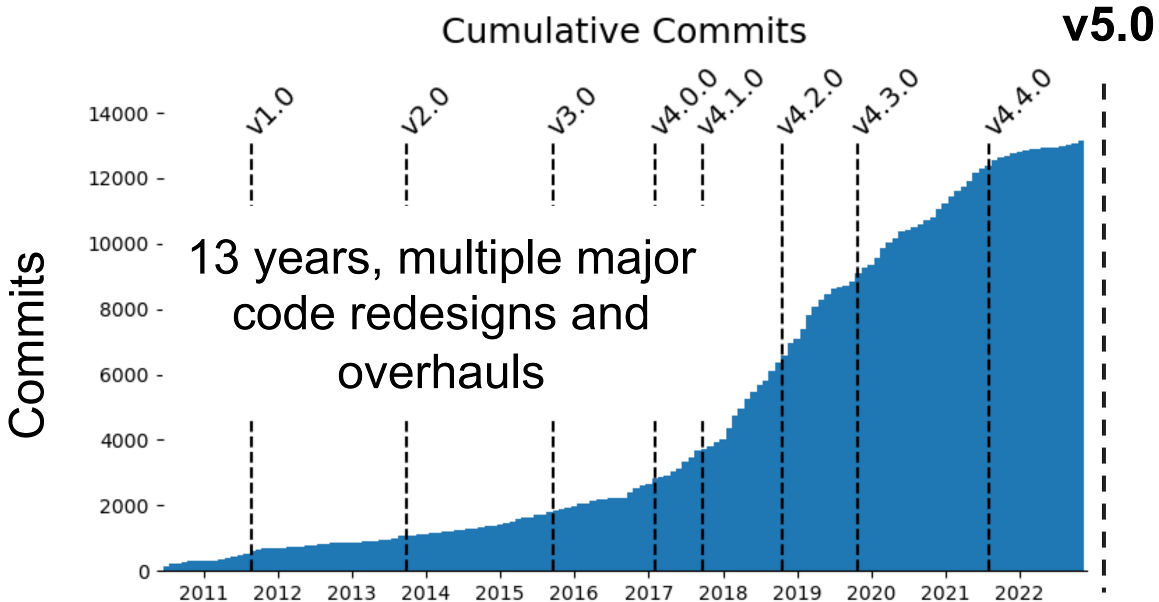
Peter Hill
Ben Dudson
David Dickinson
David Schworer
John Omotani
Michael Loiten
Joseph Parker
Jens Madsen
Jarrod Leddy
George Breyiannis
Brendan Shanahan
Ilon Joseph
Hong Zhang
+ ~35 others

https://github.com/boutproject/

http://boutproject.github.io/

BOUT++ is open source

- Open source, users/developers worldwide
- Strong community, and investment in building capabilities to underpin research



<u>Top contributors</u>

Peter Hill
Ben Dudson
David Dickinson
David Schworer
John Omotani
Michael Loiten
Joseph Parker
Jens Madsen
Jarrod Leddy
George Breyiannis
Brendan Shanahan
Ilon Joseph
Hong Zhang
+ ~35 others

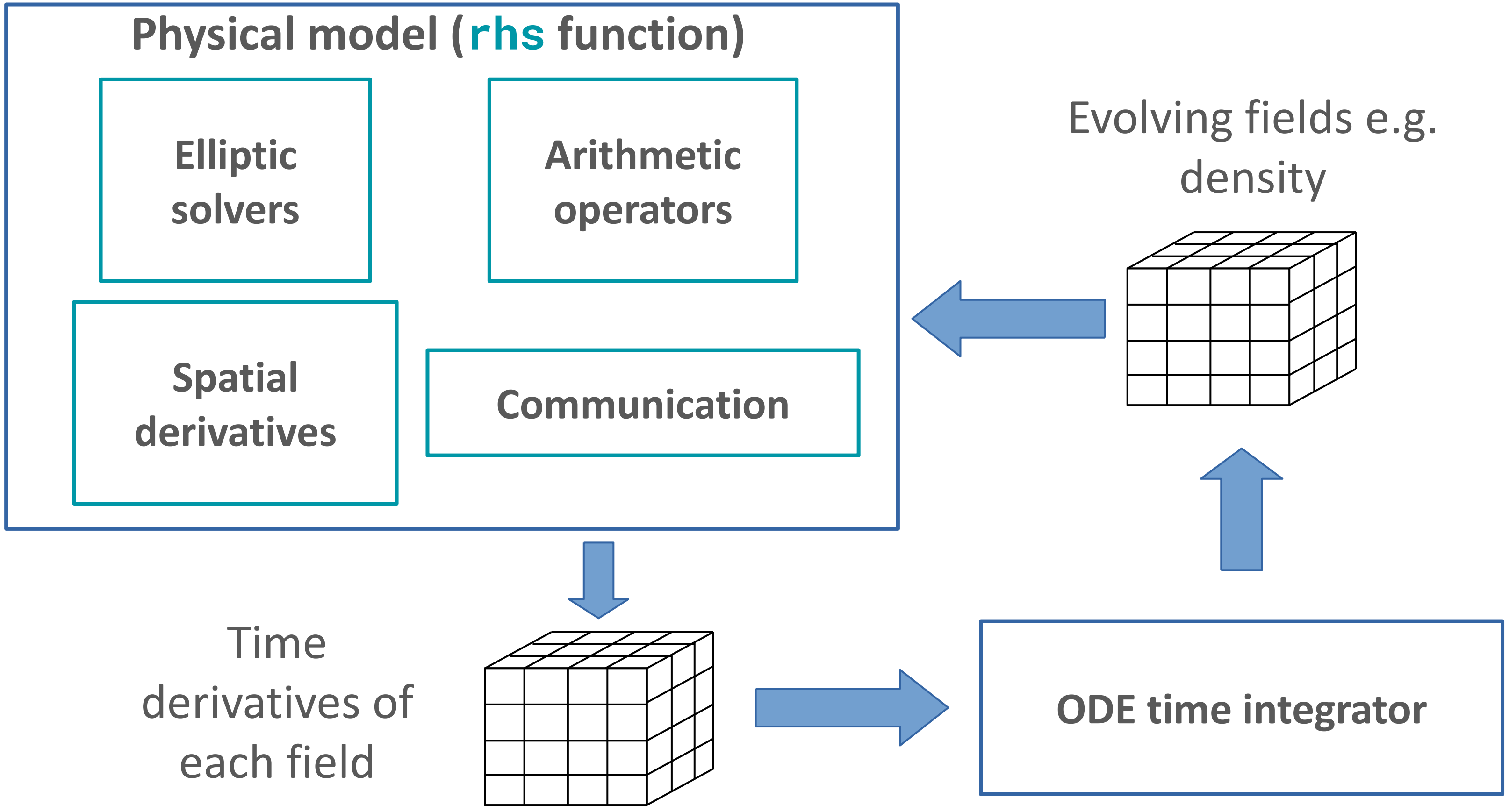
https://github.com/boutproject/

http://boutproject.github.io/

Overview

- BOUT++ structure
- Major changes
  - GPUs: RAJA and Hypre
  - 3D geometries: The FCI method
- Hermes-3: Building on BOUT++

BOUT++ uses matrix-free Method of Lines



Close correspondence between model and code

$$\frac{\partial n}{\partial t} = -\frac{1}{B}\underline{b} \times \nabla\phi \cdot \nabla n + 2\rho_s \frac{\partial n}{\partial z} + D_n \nabla_\perp^2 n$$

$$\frac{\partial \omega}{\partial t} = -\frac{1}{B}\underline{b} \times \nabla\phi \cdot \nabla \omega + \frac{2\rho_s}{n}\frac{\partial n}{\partial z} + \frac{D_{vort}}{n} \nabla_\perp^2 \omega$$

$$\omega = \nabla_\perp^2 \phi$$

Elliptic inversion

Domain-specific language in C++

```
ddt(n) = -bracket(phi, n)
         + 2 * DDZ(n) * rho_s
         + D_n * Delp2(n);

ddt(omega) = -bracket(phi, omega)
             + 2 * DDZ(n) * rho_s / n
             + D_vort * Delp2(omega) / n;

phi = laplacian->solve(omega);
```

https://github.com/boutproject/BOUT-dev/tree/master/examples/blob2d

Guiding principle of BOUT++ is flexibility

- Choice of numerical method for each operator
- Can be specified at runtime or compile time
- A flexible input configuration format, with arbitrary expressions (Turing complete)

BOUT.inp input file:

```
[mesh]
nx = 64
ny = 1
nz = 64

[mesh:ddx]
first = C4
second = C2
[mesh:ddz]
first = U2
```

blob2d.cxx source code:

```
ddt(n) = -bracket(phi, n, BRACKET_ARAKAWA)
         + 2 * DDZ(n, CELL_CENTRE, "FFT") * rho_s
         + D_n * Delp2(n);
```

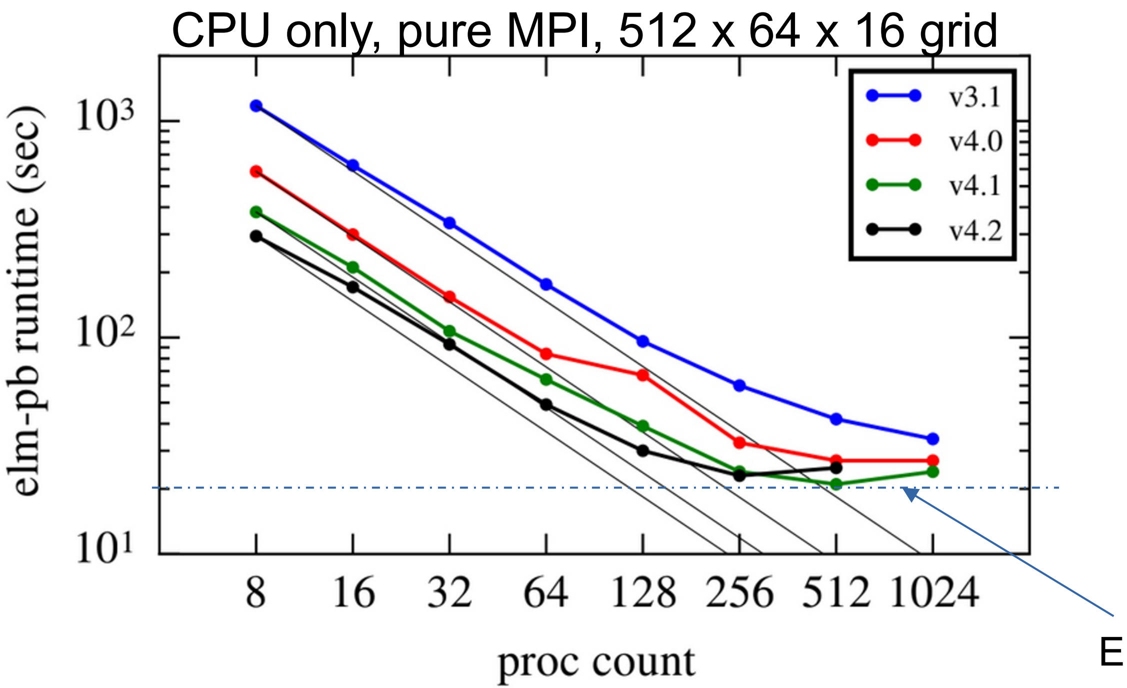
Command line:

```
./blob2d solver:type=rk4 laplace:type=petsc
    mesh:nx=128
```

Improvement in performance over time

Joseph Parker, David Dickinson, Peter Hill and Ben Dudson

BOUT++ Workshop 2018



procs	128	256
v3.1 Oct 16	96	60
v4.0 Feb 17	67	33
v4.1 Sep 17	39	25
v4.2 Oct 18	30	23

Run time [s]

Overview

- BOUT++ structure

- Major changes
  - GPUs: RAJA and Hypre
  - 3D geometries: The FCI method

- Hermes-3: Building on BOUT++

Major changes [Oct 2018 – Jan 2022]

- Regions for iterating over arbitrary domains. Improved vectorization and OpenMP performance [v4.2, 4.3]
- Consistent support for staggered grids [v4.2 – v5]
- Improved support for 3D coordinates and complex boundaries [v4.3 – v5]
- Input file language extended, internationalized [v4.3]
- Adopted the CMake build system [v4.4 – v5]
- Input & output data provenance tracking [v4.4-v5]
- Replaced I/O system, using flexible dictionary structure to exchange data [v5]
- GPU and CPU improved performance with RAJA [v5]
- New steady-state solver for transport problems, borrowing from UEDGE [v5]
- Many more tests: Now 1853 unit, 61 integrated, and 22 MMS tests

Version 5 has 3,699 commits, 91k lines changed, compared to v4.4.2
https://github.com/boutproject/BOUT-dev/pull/2604
Currently in the “next” github branch

Balancing usability and performance

- In BOUT++ functions operate on whole fields (arrays of data)
  - Simple interface for non-C++ experts
  - Each operation (+,-,*,/, DDX) loops over the domain
  - These loops are too small to parallelise efficiently (esp on GPUs)

$$\frac{\partial n}{\partial t} = -\frac{1}{B}\underline{b} \times \nabla\phi \cdot \nabla n + 2\rho_s \frac{\partial n}{\partial z} + D_n \nabla_\perp^2 n$$

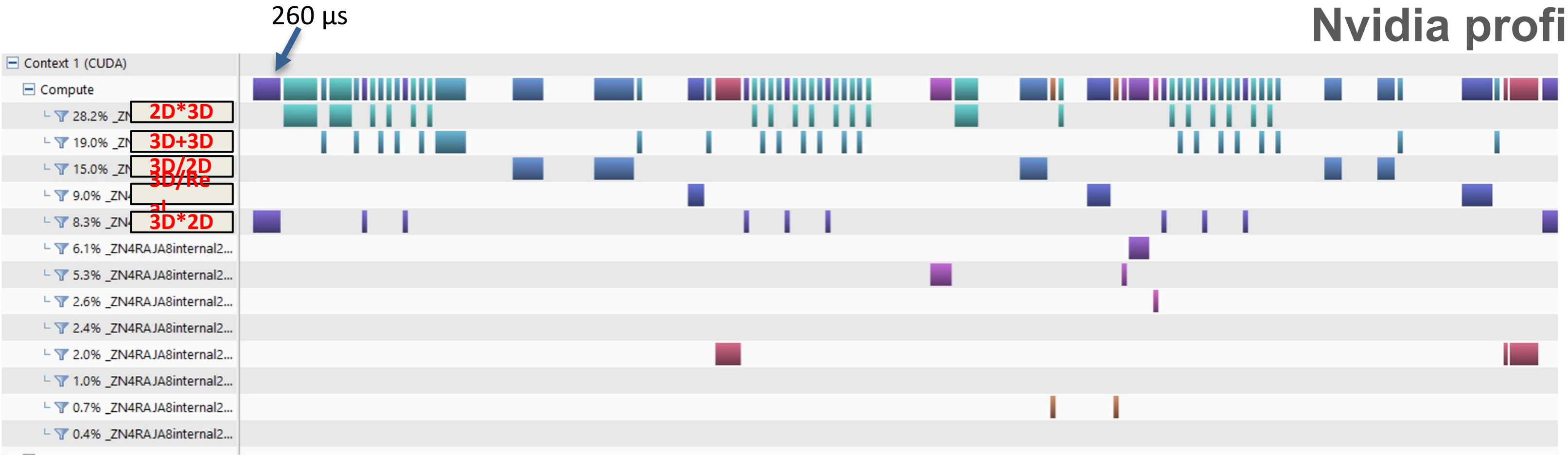
```
ddt(n) = -bracket(phi, n)
         + 2 * DDZ(n) * rho_s
         + D_n * Delp2(n);
```

This has 9 separate kernels, each with a loop over the domain

Balancing usability and performance

- In BOUT++ functions operate on whole fields (arrays of data)
  - Simple interface for non-C++ experts
  - Each operation (+,-,*,/, DDX) loops over the domain
  - These loops are too small to parallelise efficiently (esp on GPUs)

Nvidia profiler



Steve Glenn, Bill Meyer (LLNL)

Code transformation methods

MACROS

C++ templates

Code generation

- ✓ Conceptually simple
- ✓ Familiar to most programmers

- ✗ Works on a text level, not semantic
- ✗ Can lead to surprising bugs (text mangling)
- ✗ No access to type information
- ✗ Not suitable for complex transformations

Used in BOUT++ to reduce “boilerplate”, and define loops that can be changed at compile time (e.g. OpenMP, RAJA).

Code transformation methods

MACROS

C++ templates

Code generation

- ✓ General transformations (Turing complete)
- ✓ Widely used to merge loops
  e.g. Eigen, xtensor, Blitz++, Kokkos, ...
- ✗ Unhelpful error messages
- ✗ Compilation can be slow
- ✗ Complex, requires experienced developers
  to maintain and extend code
- ✗ Can run into compiler bugs

Used in BOUT++ to enable compile-time checks,
use types to specialise code

Code transformation methods

MACROS

C++ templates

Code generation

- ✓ General transformations
- ✓ Many different tools available
- ✓ Abstracts over implementation and architecture details

- ✗ Debugging can be very difficult
- ✗ Link between user code and performance can be unclear
- ✗ May need to maintain code generation tool

Jinja template engine used in BOUT++ to generate repetitious code (https://jinja.palletsprojects.com)

Using RAJA & Umpire to port to GPUs

- RAJA provides mechanisms to generate CUDA code from C++:

Execution policy e.g CUDA. Compile-time choice

```
RAJA::forall<EXEC_POL>(RAJA::RangeSegment(0, indices.size()),
     [=] RAJA_DEVICE(int id) {

     /* … your code here … */
})
```

Iteration range

C++ lambda function body

✓ Provides a path for incremental porting existing code to GPUs

– Requires additional tools to manage memory. Umpire used here.

X A “leaky abstraction”: Details of memory, CUDA limitations matter (especially with complex data structures)

Opt-in performance tuning

- Aim to maintain usability, readability for physicists
- Incrementally transition from original code to improve performance
- Ease debugging (c.f. templates, code generation)

Original

```
ddt(n) = -bracket(phi, n)
         + 2 * DDZ(n) * rho_s
         + D_n * Delp2(n);
```

Merged loops

Macro: OpenMP, vectorise, RAJA

```
BOUT_FOR(i, region) {

  ddt(n)[i] = -bracket(phi, n, i)
           + 2 * DDZ(n, i) * rho_s
           + D_n * Delp2(n, i);
}
```

Opt-in performance tuning : checks outside loops

- Borrow an API idea from SYCL: Lightweight wrappers of raw buffers
- Run-time checking performed on construction (outside loop)
- Template arguments enable compile-time checks, optimisations

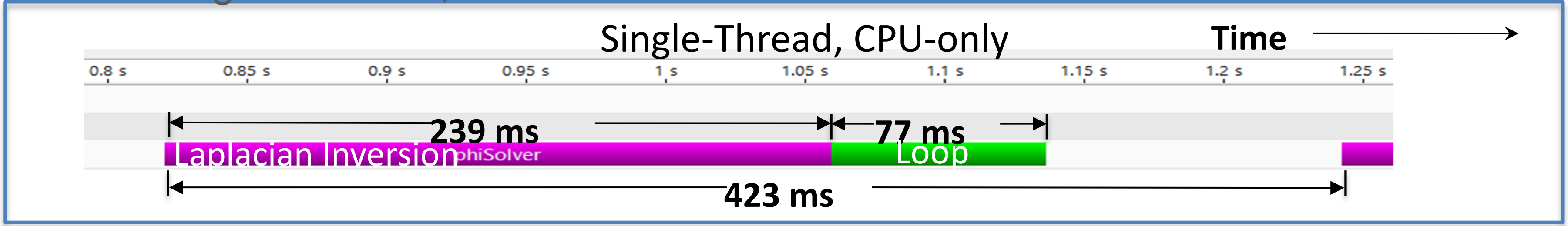
```
auto n_acc = FieldAccessor<>(n);
auto phi_acc = FieldAccessor<>(phi);
auto jpar_acc = FieldAccessor<CELL_YLOW>(Jpar);

BOUT_FOR(i, region) {

  ddt(n)[i] = -bracket(phi_acc, n_acc, i)
          + 2 * DDZ(n_acc, i) * rho_s
          + D_n * Delp2(n_acc, i);
}
```

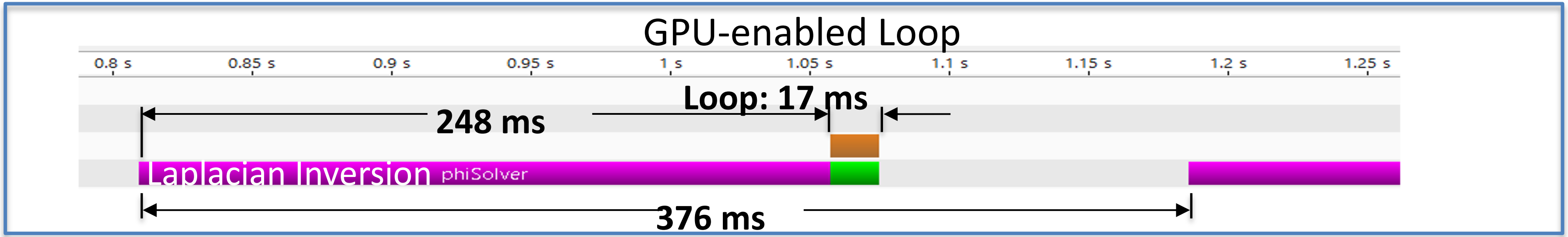
Run-time checks

Compile-time type checking

Performance improvements: RAJA

Running on Lassen, LLNL





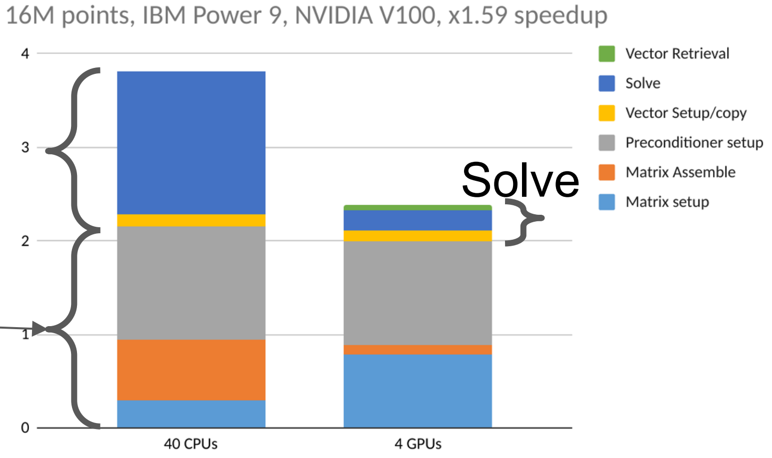
- Only one kernel launch per iteration, due to loop merging
- 1260 x 1256 grid for benchmarking

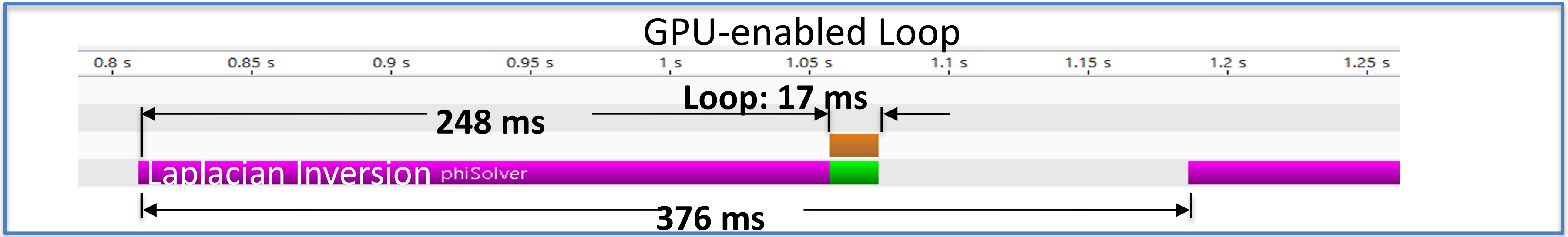
- GPU loop speedup = 77/17 ≈ 4.5X
- Overall speedup ≈ 423/376 ≈ 1.13X

Ongoing work to port to GPUs

- Merging kernels and RAJA works well
- Significant time can be spent in inversion of elliptic operators
- Keeping GPUs busy can be hard
- Setup costs are significant (note: matrix is time-dependent!)

Test on Lassen





GPU functionality is available in v5

- Check out the "next" (development) branch of BOUT++:

  ```
  $ git clone -b next https://github.com/boutproject/BOUT-dev.git
  ```

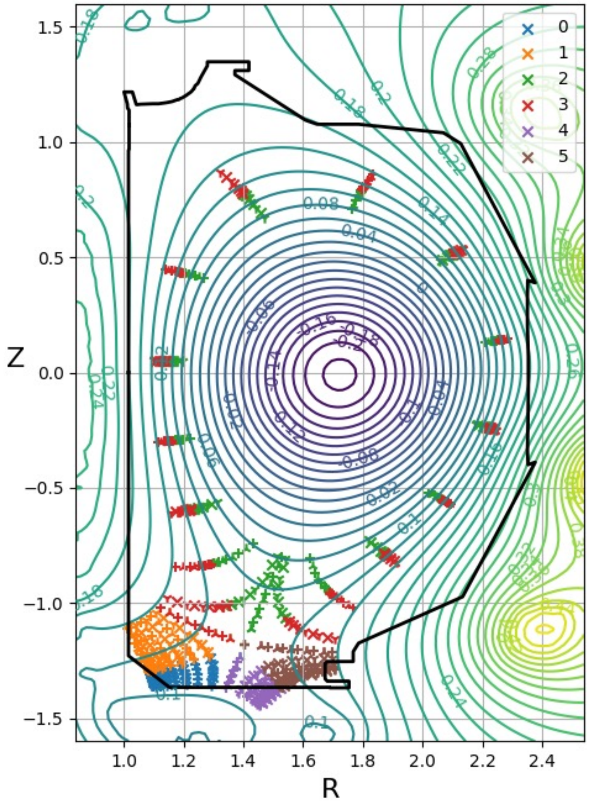
- Manual page: https://bout-dev.readthedocs.io/en/latest/user_docs/gpu_support.html

- Examples
  - `blob2d-outerloop`
    https://github.com/boutproject/BOUT-dev/tree/next/examples/blob2d-outerloop
  - `hasegawa-wakatani-3d`
    https://github.com/boutproject/BOUT-dev/tree/next/examples/hasegawa-wakatani-3d
  - `elm-pb-outerloop`
    https://github.com/boutproject/BOUT-dev/tree/next/examples/elm-pb-outerloop

Grid generation using Hypnotoad

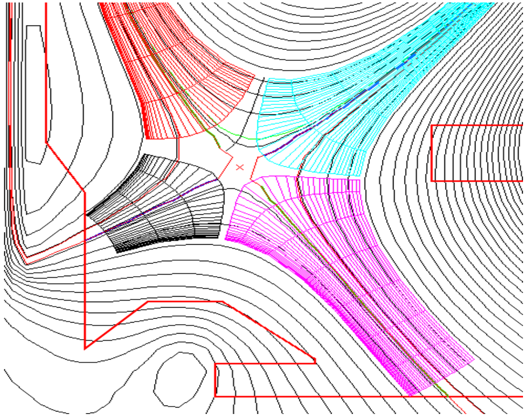
- Python grid generator, mainly written by J.Omotani
- Can generate non-orthogonal grids, here using orthogonal grids
- Interactive GUI or automated script
- Can adjust packing of cells around separatrix and/or close to targets as needed

- Sequence of grids created for convergence and performance testing
- Python tools for interpolating between grids

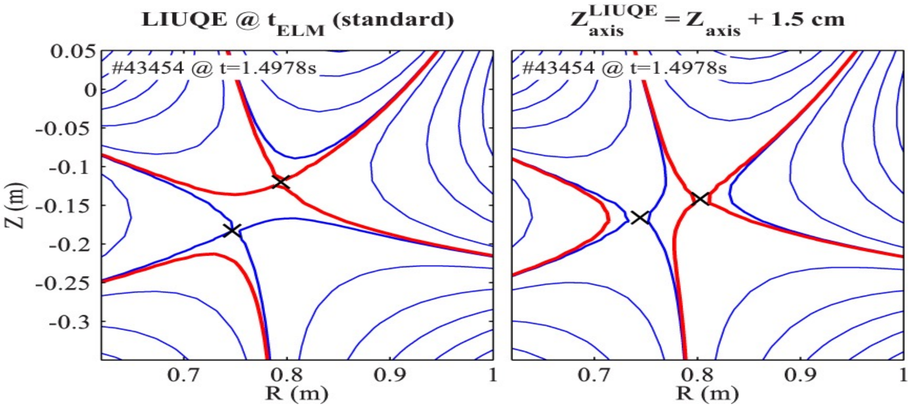


Complex meshing problems (2D & 3D)

X-point

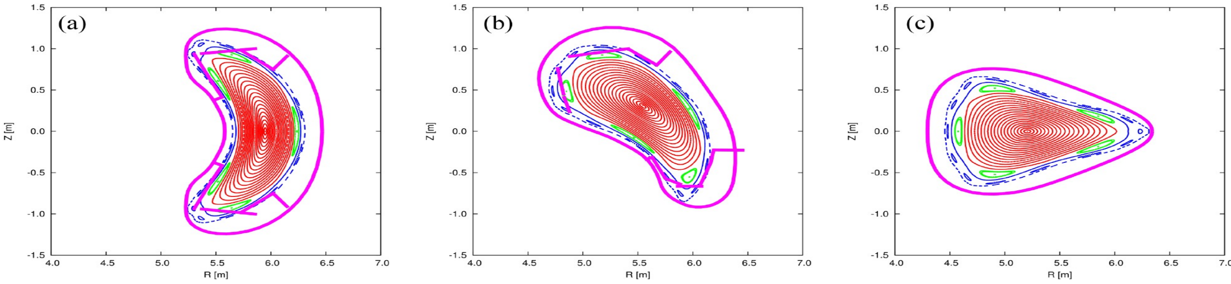


Snowflake



W.A.J. Vijvers et al 2014 Nucl. Fusion 54 023009

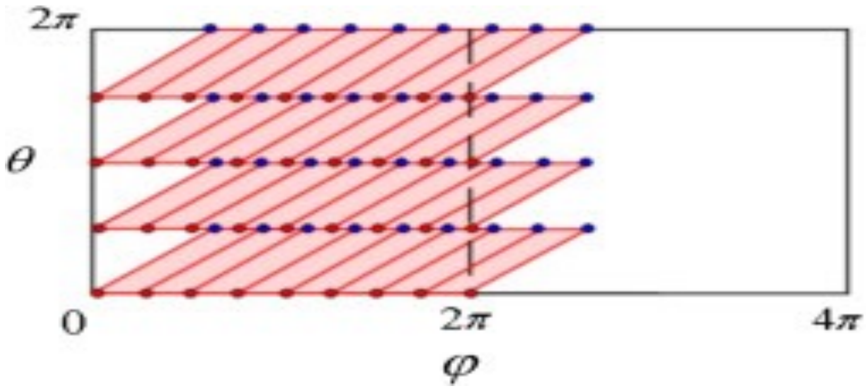
Stellarator



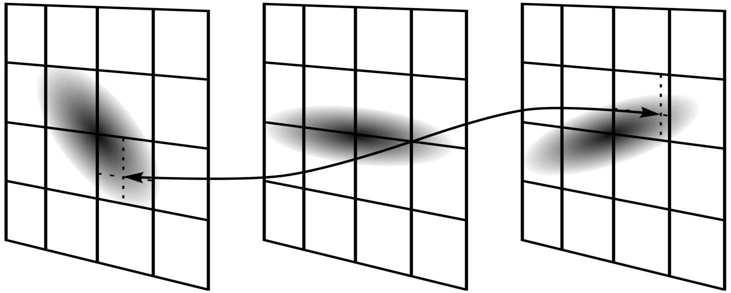
Y Suzuki and J Geiger 2016 Plasma Phys. Control. Fusion 58 064004

FCI: Field-line following + interpolation

Shifted metric (Dimits / Scott) : 1D interpolation



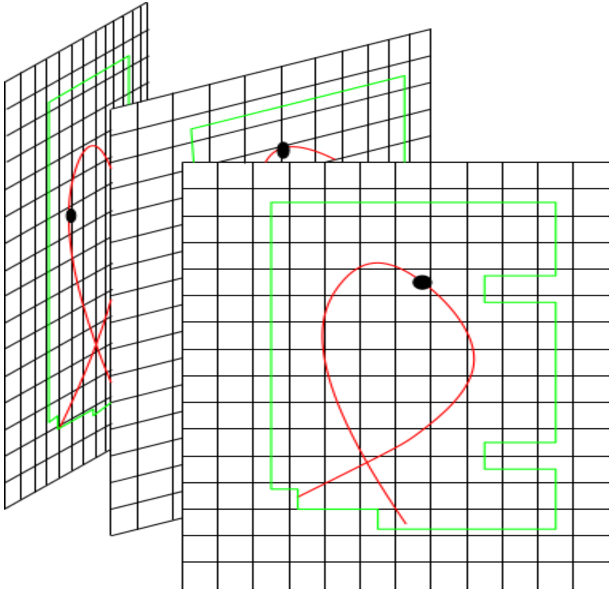
FCI: 2D interpolation

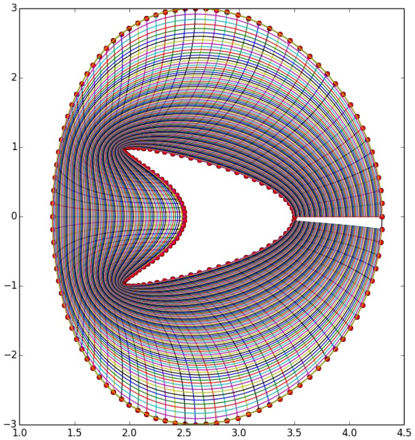


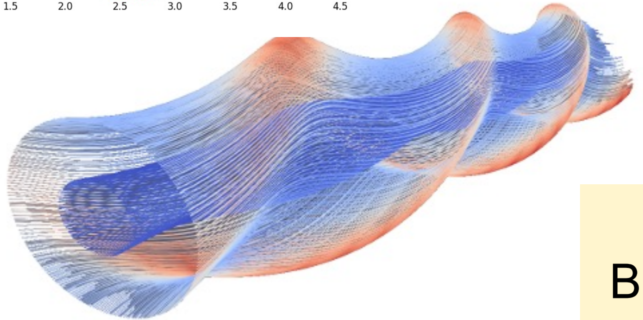
[8] F Hariri and M Ottaviani CPC **184** 2419 (2013)
[9] B Shanahan, P Hill and B Dudson. Journal of Physics; Conference Series **775**, 012012 (2016)
[10] P Hill, B Shanahan and B Dudson CPC **213**, 9-18 (2017)

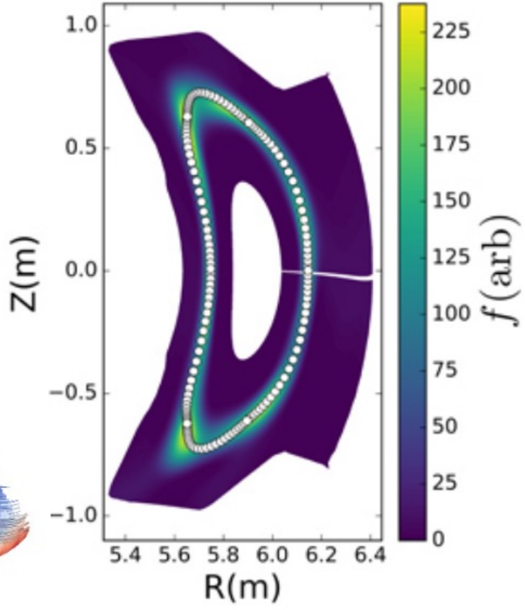
An illustration of the Flux Coordinate Independent method for parallel derivatives [9].

Gridding of poloidal plane independent of magnetic field structure









Zoidberg

http://bout-dev.readthedocs.io/en/latest/user_docs/zoidberg.html

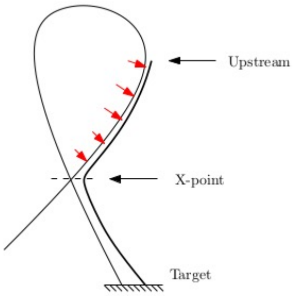
BSTING project
B.Shanahan, D.Bold
IPP Greifswald

Overview

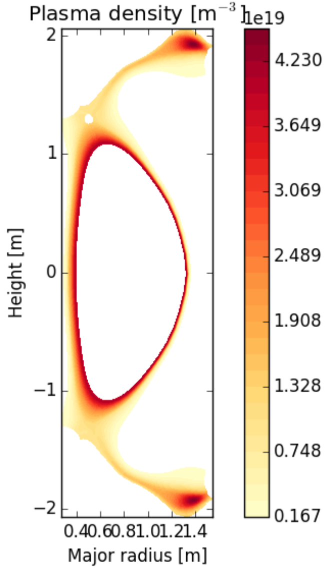
- BOUT++ structure
- Major changes
  - GPUs: RAJA and Hypre
  - 3D geometries: The FCI method
- Hermes-3: Building on BOUT++

Model development: SOL & Divertor turbulence

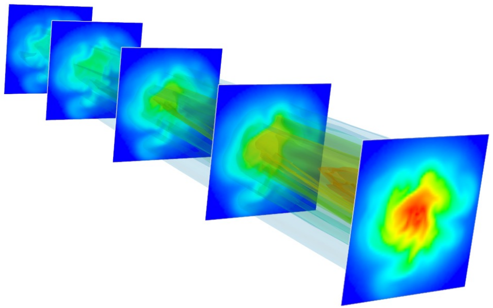
SD1D

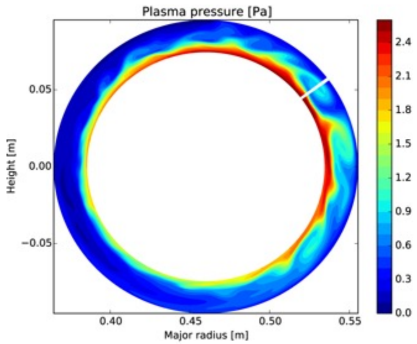


Hermes (2D)



Hermes (3D)





Key features:

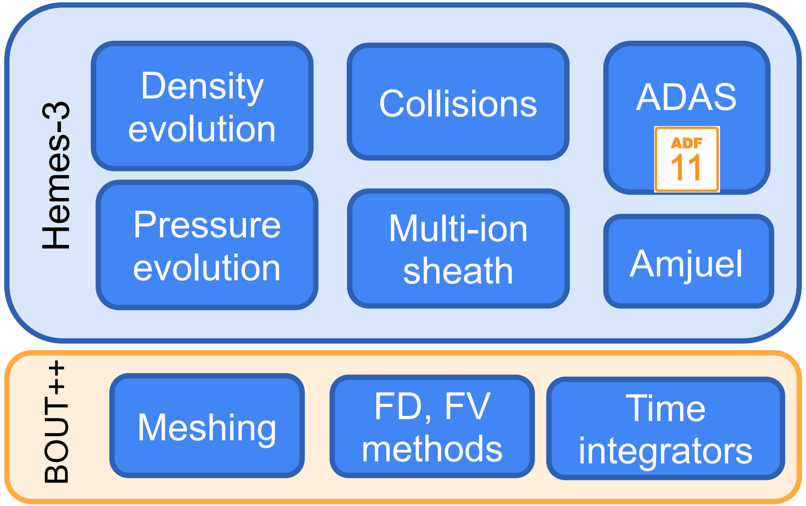
- 'full-f' : Evolve <u>profiles + fluctuations</u>
- Includes transport physics e.g. neutrals

B.Dudson et al PPCF 63 055013 (2021) https://doi.org/10.1088/1361-6587/abe21d
B.Dudson, J.Leddy PPCF 59 054010 (2017) https://doi.org/10.1088/1361-6587/aa63d2
J.Leddy, et al. Comp. Phys. Comm 212,59-68 (2017) http://dx.doi.org/10.1016/j.cpc.2016.10.009
J.Leddy, B.Dudson JNM (2016) https://doi.org/10.1016/j.nme.2016.09.020

Hermes-3: Multi-species transport and turbulence models

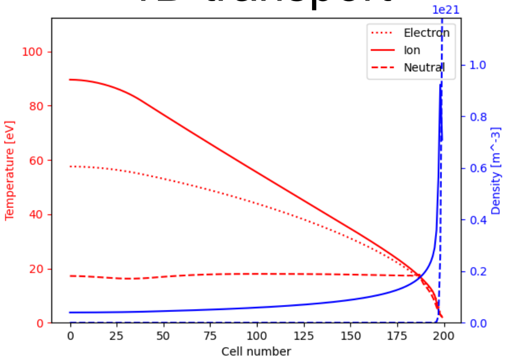
https://github.com/bendudson/hermes-3 https://hermes3.readthedocs.io

- Arbitrary number of species and ionisation states: D, T, He, Ne, …
- Full-f, flux-driven transport & turbulence

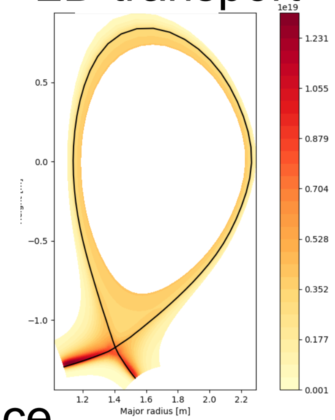


PETSc hypre sundials

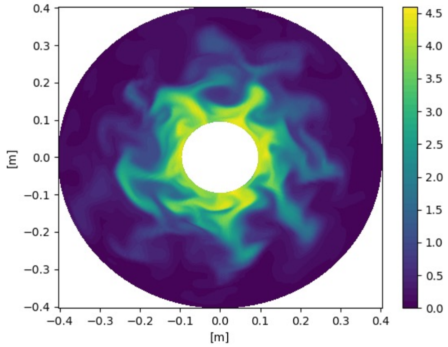
1D transport



2D transport



3D turbulence



Recent simulations in 1-, 2- and 3D

- 1D transport of neon in deuterium SOL plasma
  - Evolving each charge state and atomic species as a separate fluid
- 2D transport in DIII-D geometry
  - Solve for deuterium, tritium and helium ions and atoms
- 3D turbulence and transport in LAPD
  - Both isothermal, and hot electron (ionisation source). Single ion species
- 3D turbulence and transport in DIII-D & limiter plasmas
  - Isothermal, single ion species to start with

System of equations is specified in the input file

- Top-level components specify the species, collective effects and modifiers

```
[hermes]
components = (e, d+, sound_speed, vorticity,
              sheath_boundary, collisions, polarisation_drift)
```

- Each species' equations are:

```
[e]
type = evolve_density, evolve_momentum, isothermal

[d+]
type = quasineutral, evolve_momentum, isothermal
```

Solving drift-reduced fluid equations

Electrons

$$\frac{\partial n_e}{\partial t} = -\nabla \cdot \left[n_e \left(\mathbf{v}_{E\times B} + \mathbf{b}v_{||e}\right)\right]$$

$$\frac{\partial}{\partial t}\left(m_e n_e v_{||e}\right) = -\nabla \cdot \left[m_e n_e v_{||e} \left(\mathbf{v}_{E\times B} + \mathbf{b}v_{||e}\right)\right] - \partial_{||}p_e - e n_e E_{||} + \underbrace{m_e n_e \nu_{ei}\left(v_{||d+} - v_{||e}\right)}_{\texttt{collisions}}$$

$$p_e = n_e T_e$$

Ions

$$n_{d+} = n_e$$

$$\frac{\partial}{\partial t}\left(m_{d+} n_{d+} v_{||d+}\right) = -\nabla \cdot \left[m_{d+} n_{d+} v_{||d+} \left(\mathbf{v}_{E\times B} + \mathbf{b}v_{||d+}\right)\right] - \partial_{||}p_{d+} + e n_{d+} E_{||} + \underbrace{m_e n_e \nu_{ei}\left(v_{||e} - v_{||d+}\right)}_{\texttt{collisions}}$$

$$p_{d+} = n_{d+} T_{d+}$$

Drifts

$$\boldsymbol{v}_{E\times B} = \frac{\boldsymbol{b}\times\nabla\phi}{B}$$

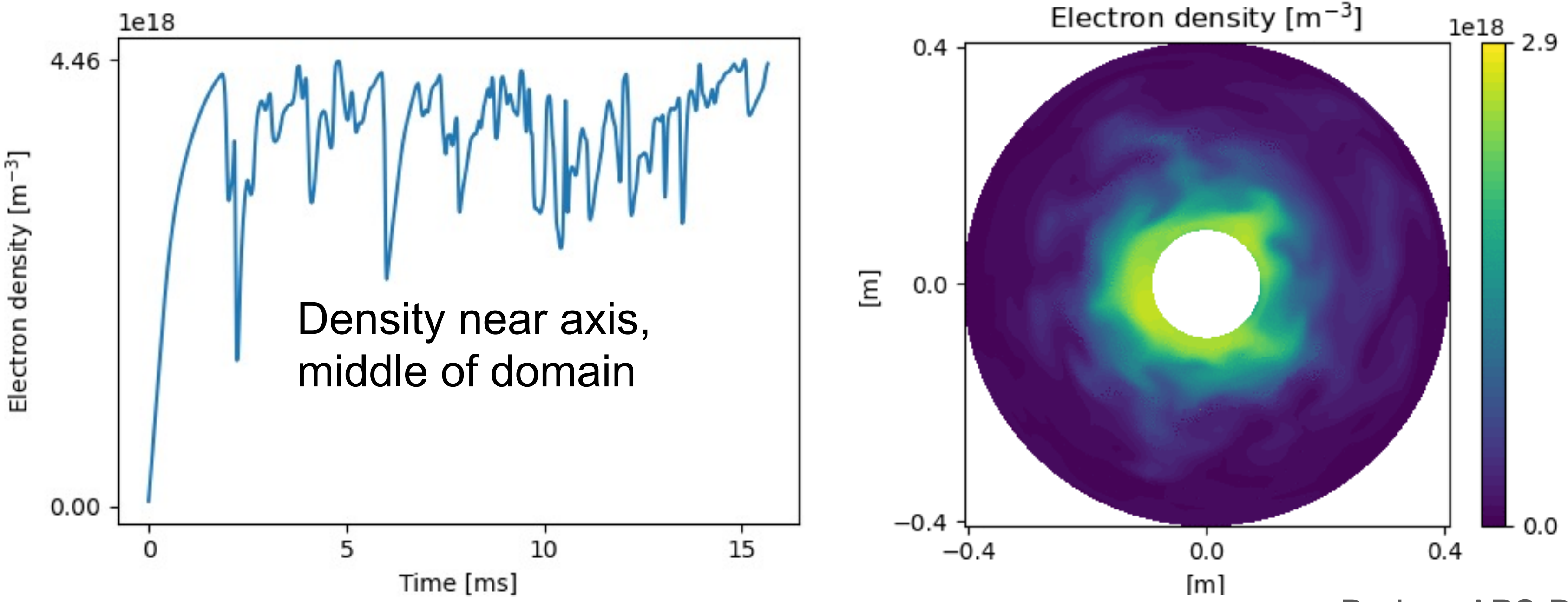
Vorticity

$$\nabla \cdot \left(\frac{\overline{m_i n}}{B^2}\nabla_\perp \phi\right) = \omega$$

$$\frac{\partial \omega}{\partial t} = -\nabla \cdot \left(\omega v_{E\times B}\right) + \nabla \cdot \left(n_{d+} v_{||d+} - n_e v_{||e}\right)$$

3D turbulence in LAPD geometry

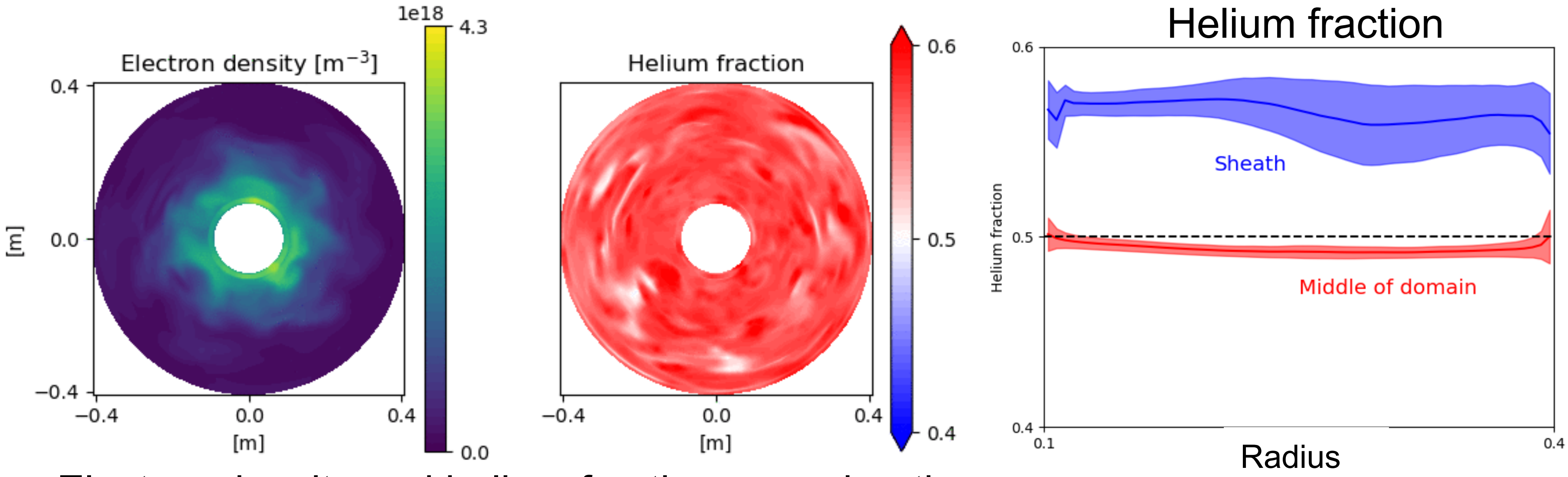
- Isothermal, single ion species, no neutrals
- Uniform source of particles in domain; sheath boundary at both ends
- Resolution: 64 x 16 x 64 (radial x parallel x azimuthal)



Dudson APS-DPP 2022

We can run turbulence simulations with an arbitrary number of ion species (e.g. D + He)

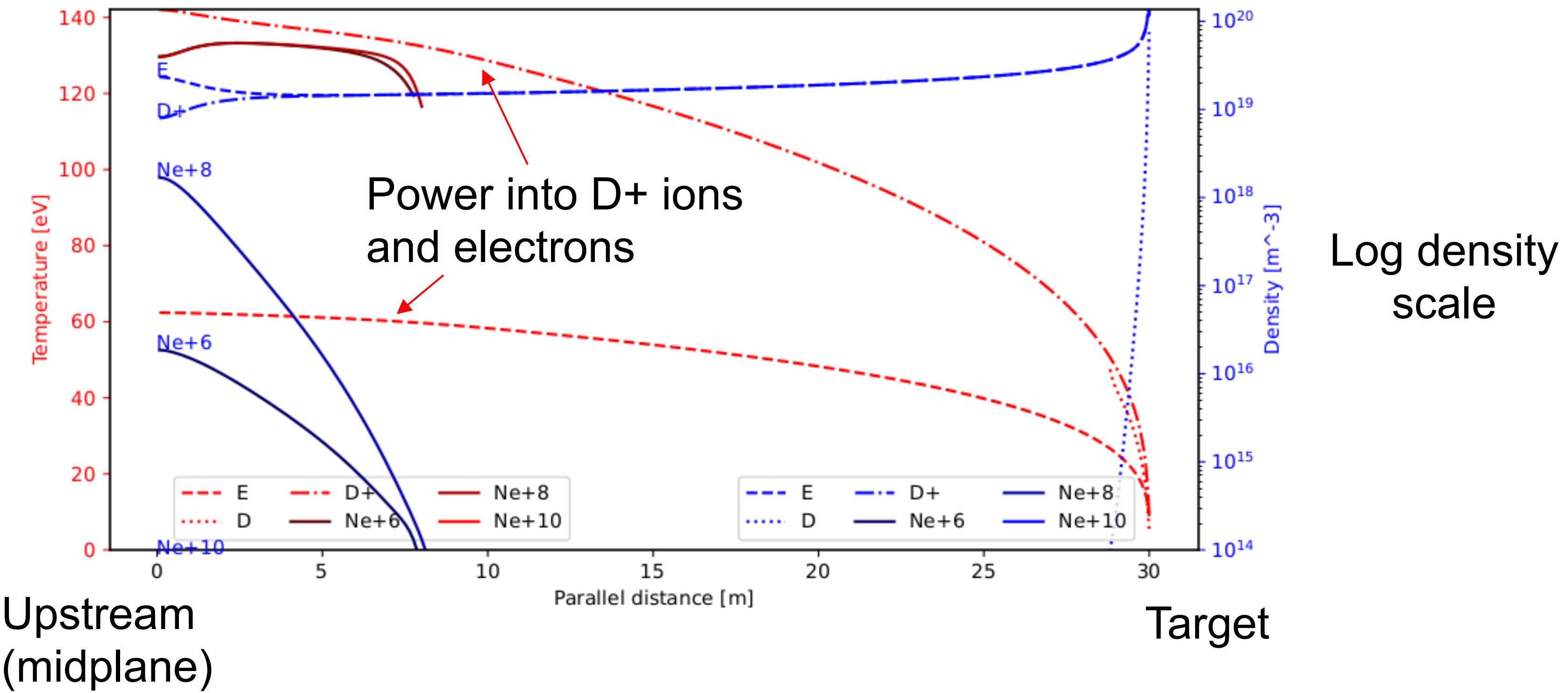
- No code changes needed. Input file specifies species and equations
- Here showing deuterium and helium (1+) ions
- Fuelling at 50/50 ratio, enhanced helium fraction near sheaths



Electron density and helium fraction near sheath

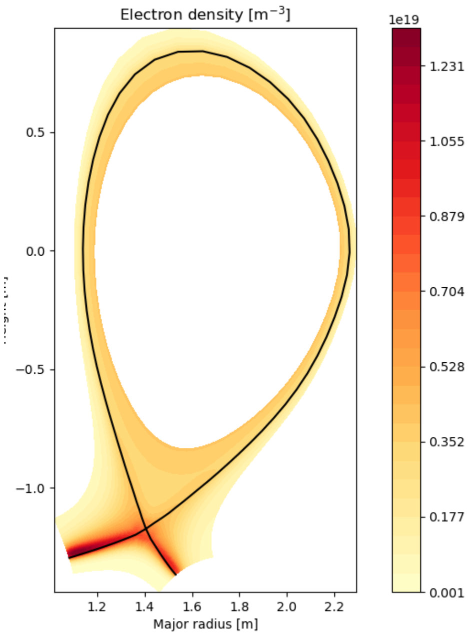
1D multi-fluid transport

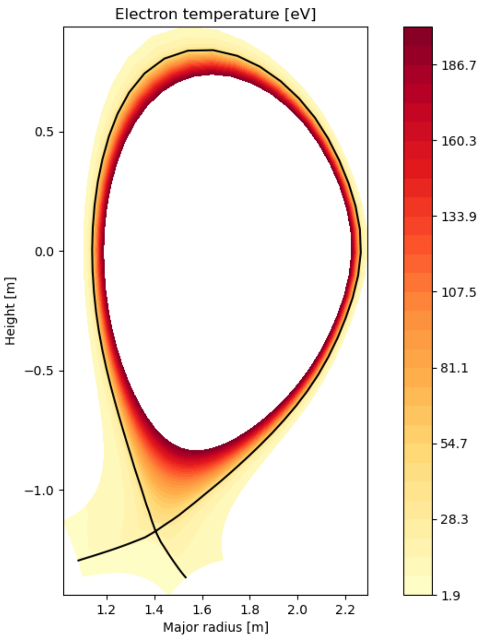
- Model a 1D domain, from “upstream” (no-flow) to “target” (sheath)
- D+, Ne+ … Ne+10 ions, D & Ne atoms. Only plotting highest density species.



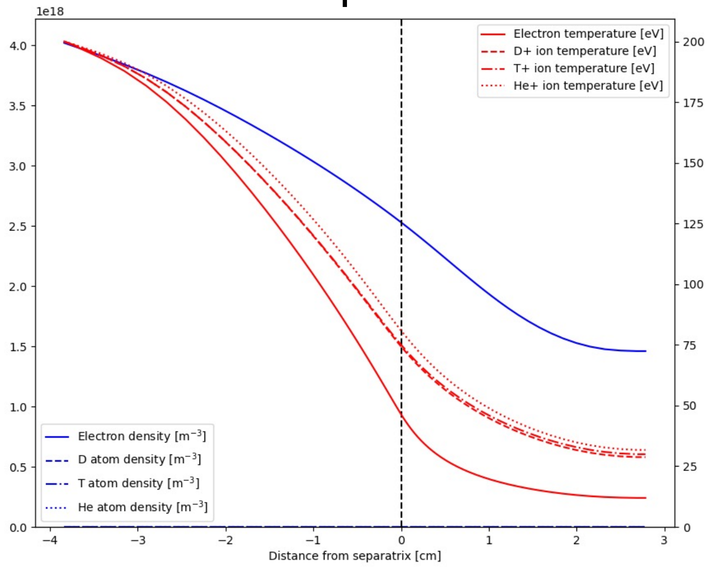
2D transport in DIII-D geometry

- Resolution: 64 x 128 (radial x poloidal, excluding boundary)
- D+, T+ and He+ ions; D, T and He atoms (fluids)



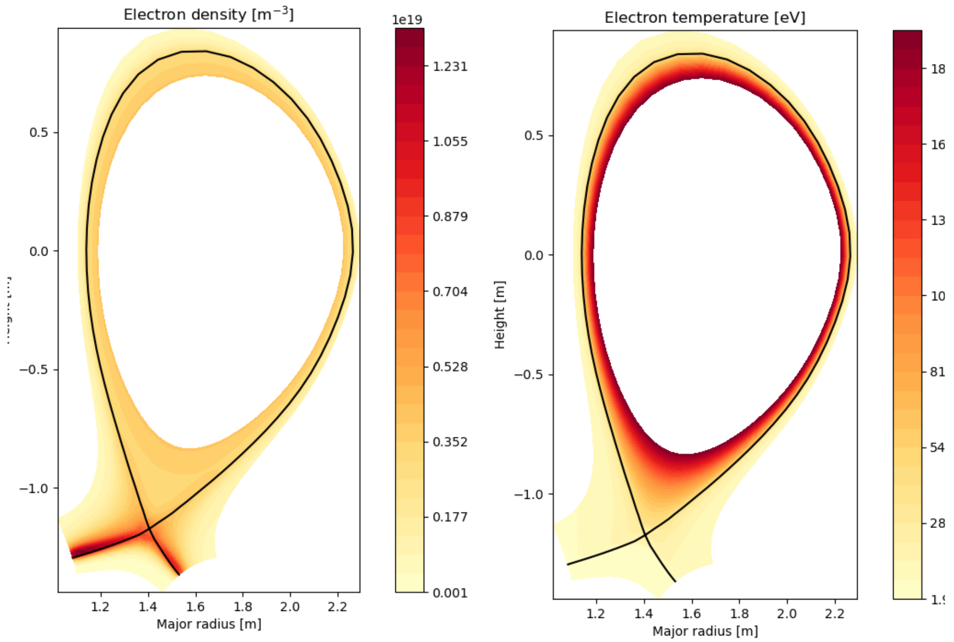


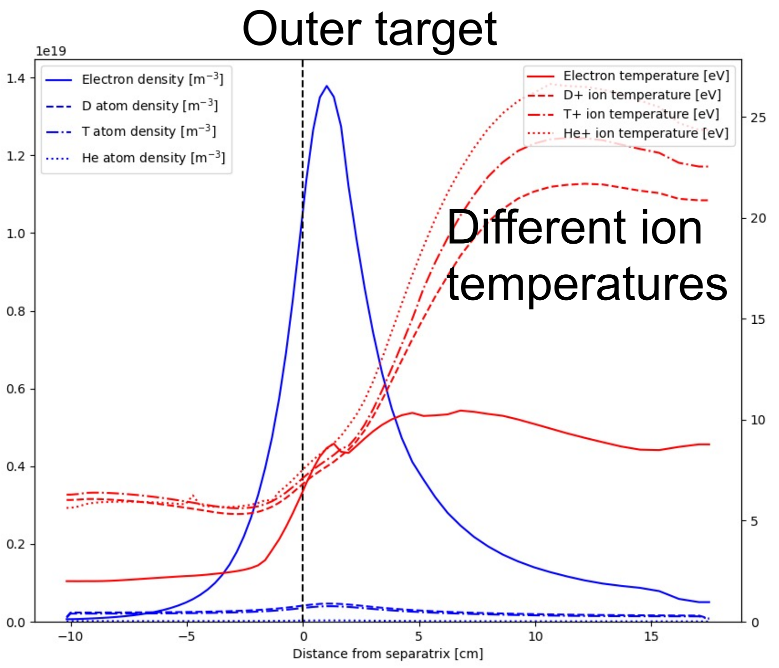
Midplane



2D transport in DIII-D geometry

- Resolution: 64 x 128 (radial x poloidal, excluding boundary)
- D+, T+ and He+ ions; D, T and He atoms (fluids)





Flexible tool for edge simulations
Note: **under development**

- All of these simulations run the same executable
  - Species, equations & reactions are configured in input file
  - Geometry (1,2,3D; linear, tokamak) in input or separate mesh file

- Atomic reactions and multi-ion support:
  - Hydrogen and helium atomic reactions from Amjuel
  - Neon reactions from ADAS
  - Tskhakaya & Kuhn multi-ion sheath boundary conditions

- Many solver / time integration options, making use of PETSc, Hypre & SUNDIALS

Tskhakaya, David and S. Kuhn. “Boundary conditions for the multi-ion magnetized plasma-wall transition.” JNM 337 (2005): 405-409.

Some applications
Note: **under development**

- Tokamak edge and divertor transport & turbulence modelling
  - Including neutrals, impurities and drifts
  - Comparison to DIII-D data, particularly impurity injection effects on turbulence

- Multi-ion plasma turbulence
  - Validation on LAPD (experiments proposed)
  - More work on multi-ion closures probably needed

- Interested?
  - Github repository : https://github.com/bendudson/hermes-3
  - Manual : https://hermes3.readthedocs.io/

Conclusions

- BOUT++ underpins a wide range of research
- Continues to develop to meet research needs, with contributions from a global community

Thank you to all contributors!

Welcome to the 2023 BOUT++ workshop!