ExCALIBUR 10

SiMLInt

Simulation and Machine Learning Integration

Bout++ User Meeting

Jan 9.-11. 2023

Moritz Linkmann

School of Mathematics and Maxwell Institute for Mathematical Sciences

The University of Edinburgh

Met Office

UK Research and Innovation

UK Atomic Energy Authority

SiMLInt

People

- Amy Krause (EPCC)
- Jacob Page (School of Mathematics)
- Anna Roubíčková (EPCC)
- Johnny Hay (EPCC)
- Elena Breitmoser (EPCC)

SiMLInt

ExCALIBUR cross-cutting project

ExCALIBUR programme	UK research programme that aims to deliver the next generation of high-performance simulation software for the highest-priority fields in UK research; Funded by UKRI/EPSRC
Cross-cutting theme	A coordinated approach addressing a known technology or infrastructure issue, which, if resolved, will lead to significant progress across a range of exascale software development challenges

Developed by EPCC and the School of Mathematics at the University of Edinburgh, UK, SiMLInt provides infrastructure that allows embedding Machine Learning capability to large-scale numerical simulations.

More information	https://excalibur.ac.uk/ https://excalibur.ac.uk/projects/simlint/
Contact us	simlint@mlist.is.ed.ac.uk

Motivation

$$\partial_t u_i + \partial_j \left(u_i u_j + p\delta_{ij} - \frac{1}{Re} S_{ij} \right) = f_i$$

fully resolved simulation → ground truth

Aim → simulation on coarser grid

$$\partial_t \overline{u}_i^\ell + \partial_j \left(\overline{u}_i^\ell \overline{u}_j^\ell + \overline{p}^\ell \delta_{ij} - \frac{1}{Re} \overline{S}_{ij}^\ell - \tau_{ij}^\ell \right) = \overline{f}_i^\ell$$

$$\tau_{ij}^\ell = \overline{u_i u_j}^\ell - \overline{u}_i^\ell \overline{u}_j^\ell$$

subgrid-scale stress tensor

$$\tau_{ij}^{\mathrm{model}} = (\ell C_S)^2 \sqrt{\overline{S}_{ij}^\ell \overline{S}_{ij}^\ell} \, \overline{S}_{ij}^\ell$$

Smagorinsky parametrisation

$$\tau_{ij}^{\mathrm{model}} = f(a, b, c, \ldots) \overline{S}_{ij}^\ell$$

multiple parameters → ML

$$\tau_{ij}^{\mathrm{model}} = X_{ij}(a, b, c, \ldots)$$

learn subgrid stresses

$$\tau_{ij}^{\mathrm{error}} = X_{ij}(\overline{\boldsymbol{u}}^\ell, \boldsymbol{u})$$

learned corrections

Motivation

$$\partial_t \overline{u}_i^\ell + \partial_j \left(\overline{u}_i^\ell \overline{u}_j^\ell + \overline{p}^\ell \delta_{ij} - \frac{1}{Re} \overline{S}_{ij}^\ell - \tau_{ij}^{\mathrm{model/error}} \right) = \overline{f}_i^\ell$$

$$\tau_{ij}^{\mathrm{model}} = (\ell C_S)^2 \sqrt{\overline{S}_{ij}^\ell \overline{S}_{ij}^\ell} \overline{S}_{ij}^\ell$$ Smagorinsky parametrisation

$$\tau_{ij}^{\mathrm{model}} = f(a, b, c, \ldots) \overline{S}_{ij}^\ell$$ multiple parameters → ML

$$\tau_{ij}^{\mathrm{model}} = X_{ij}(a, b, c, \ldots)$$ learn subgrid stresses

$$\tau_{ij}^{\mathrm{error}} = X_{ij}(\overline{\boldsymbol{u}}^\ell, \boldsymbol{u})$$ learned corrections

Motivation

$$\partial_t \overline{u}_i^\ell + \partial_j \left(\overline{u}_i^\ell \overline{u}_j^\ell + \overline{p}^\ell \delta_{ij} - \frac{1}{Re} \overline{S}_{ij}^\ell - \tau_{ij}^{\text{model/error}} \right) = \overline{f}_i^\ell$$

$\tau_{ij}^{\text{model}} = (\ell C_S)^2 \sqrt{\overline{S}_{ij}^\ell \overline{S}_{ij}^\ell} \overline{S}_{ij}^\ell$	Smagorinsky parametrisation
$\tau_{ij}^{\text{model}} = f(a, b, c, \ldots) \overline{S}_{ij}^\ell$	multiple parameters → ML
$\tau_{ij}^{\text{model}} = X_{ij}(a, b, c, \ldots)$	learn subgrid stresses
$\tau_{ij}^{\text{error}} = X_{ij}(\overline{\boldsymbol{u}}^\ell, \boldsymbol{u})$	learned corrections

Motivation

$$\partial_t \overline{u}_i^\ell + \partial_j \left(\overline{u}_i^\ell \overline{u}_j^\ell + \overline{p}^\ell \delta_{ij} - \frac{1}{Re} \overline{S}_{ij}^\ell - \tau_{ij}^{\mathrm{model/error}} \right) = \overline{f}_i^\ell$$

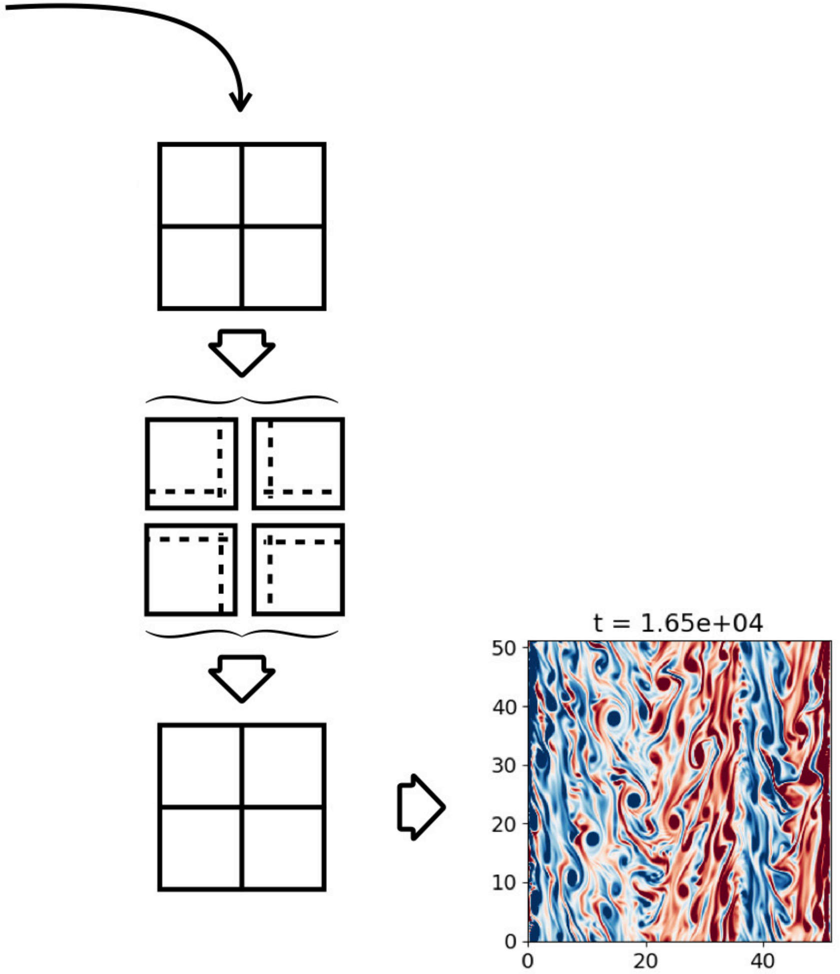
$\tau_{ij}^{\mathrm{model}} = (\ell C_S)^2 \sqrt{\overline{S}_{ij}^\ell \overline{S}_{ij}^\ell} \, \overline{S}_{ij}^\ell$	Smagorinsky parametrisation
$\tau_{ij}^{\mathrm{model}} = f(a, b, c, \ldots) \overline{S}_{ij}^\ell$	multiple parameters → ML
$\tau_{ij}^{\mathrm{model}} = X_{ij}(a, b, c, \ldots)$	learn subgrid stresses
$\tau_{ij}^{\mathrm{error}} = X_{ij}(\overline{\boldsymbol{u}}^\ell, \boldsymbol{u})$	learned corrections

→ **provide infrastructure for any of these data-driven approaches**

→ **couple BOUT++ with ML-model(s)**

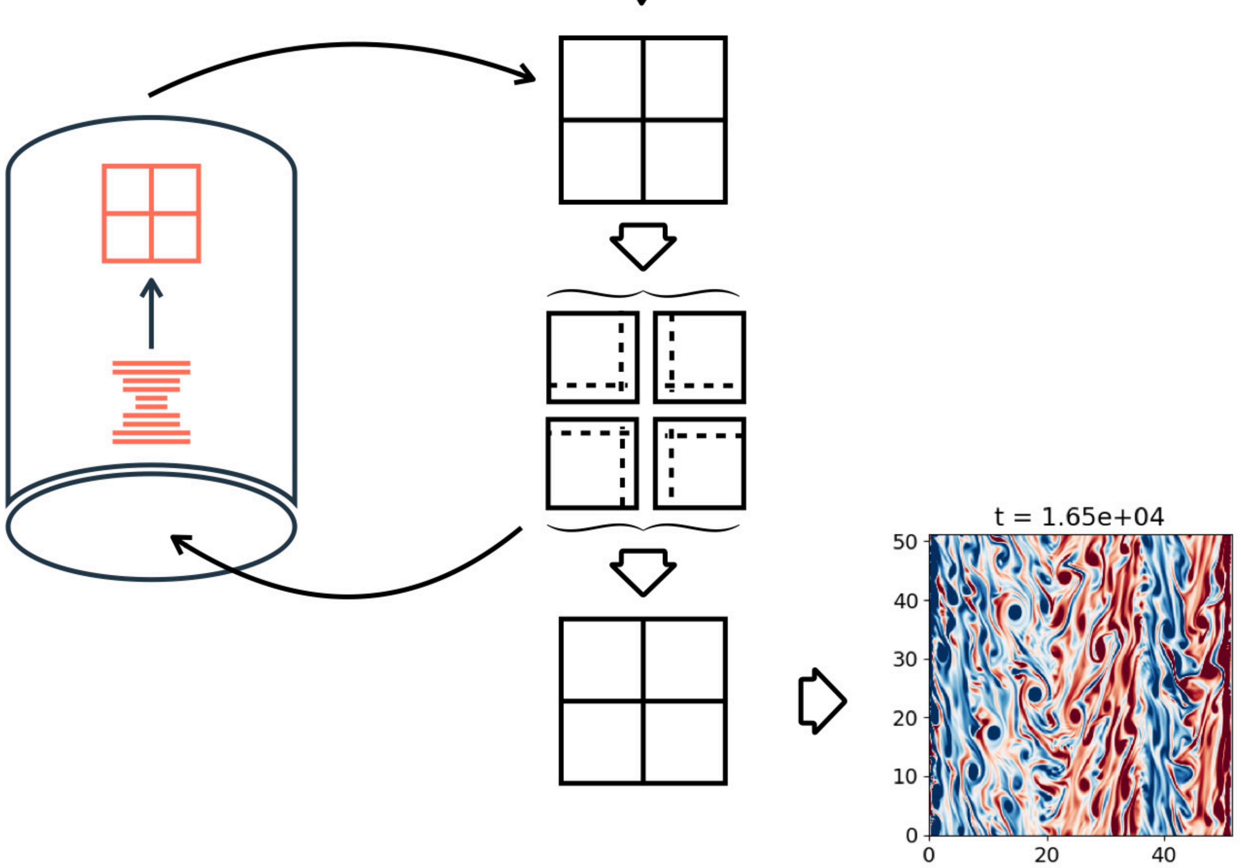
Concept and Aims

$$\frac{\partial \zeta}{\partial t} + \{\phi, \zeta\} = \alpha(\tilde{\phi} - \tilde{n}) - \mu \nabla^4 \zeta$$
$$\frac{\partial n}{\partial t} + \{\phi, n\} = \alpha(\tilde{\phi} - \tilde{n}) - \kappa \frac{\partial \phi}{\partial y} - \mu \nabla^4 n$$



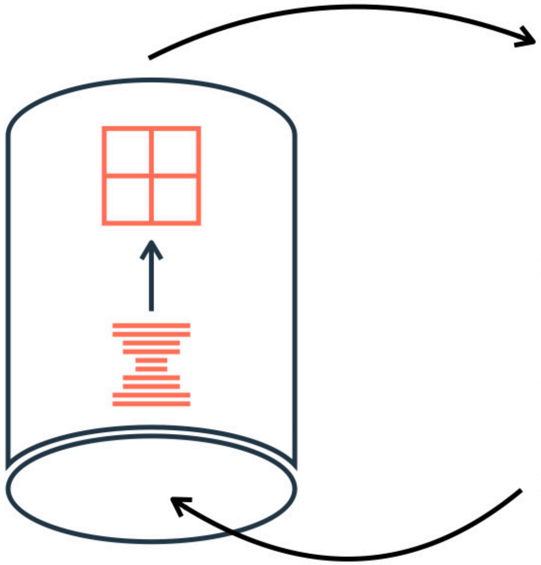
Concept and Aims

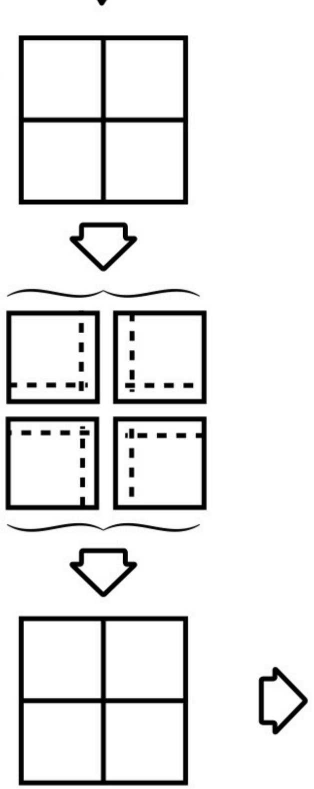
$$\frac{\partial \zeta}{\partial t} + \{\phi, \zeta\} = \alpha(\tilde{\phi} - \tilde{n}) - \mu \nabla^4 \zeta$$
$$\frac{\partial n}{\partial t} + \{\phi, n\} = \alpha(\tilde{\phi} - \tilde{n}) - \kappa \frac{\partial \phi}{\partial y} - \mu \nabla^4 n$$

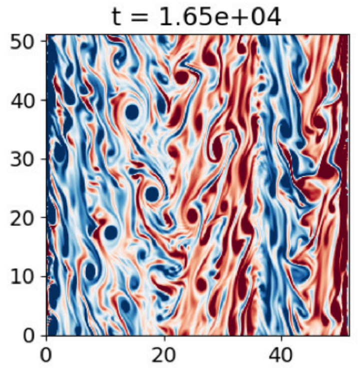


Concept and Aims

$$\frac{\partial \zeta}{\partial t} + \{\phi, \zeta\} = \alpha(\tilde{\phi} - \tilde{n}) - \mu \nabla^4 \zeta$$
$$\frac{\partial n}{\partial t} + \{\phi, n\} = \alpha(\tilde{\phi} - \tilde{n}) - \kappa \frac{\partial \phi}{\partial y} - \mu \nabla^4 n$$





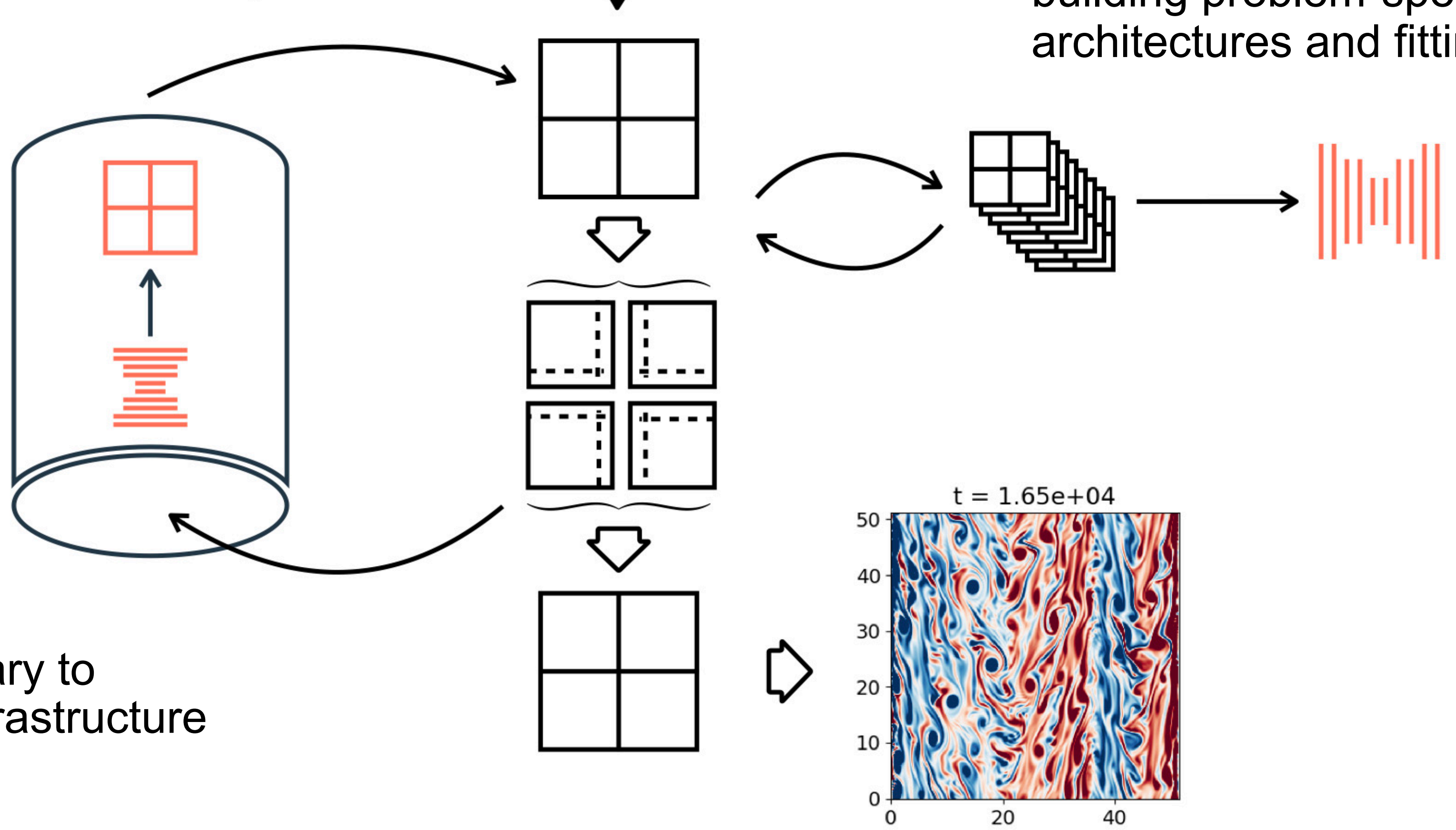


Objective 1:

Develop a library to
provide the infrastructure

Concept and Aims

$$\frac{\partial \zeta}{\partial t} + \{\phi, \zeta\} = \alpha(\tilde{\phi} - \tilde{n}) - \mu \nabla^4 \zeta$$
$$\frac{\partial n}{\partial t} + \{\phi, n\} = \alpha(\tilde{\phi} - \tilde{n}) - \kappa \frac{\partial \phi}{\partial y} - \mu \nabla^4 n$$



Objective 2:

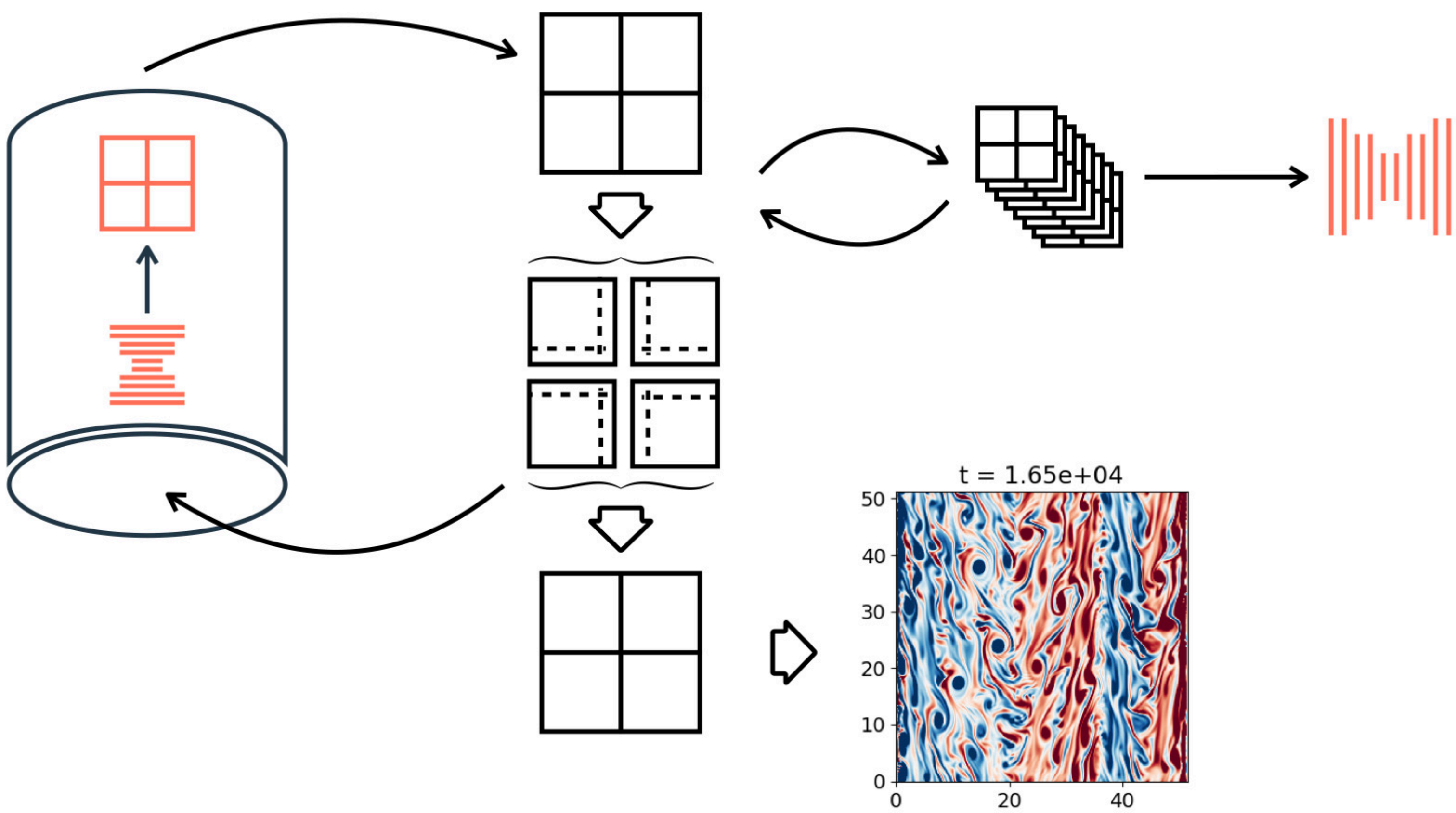
Support generation of training data, building problem-specific CNN architectures and fitting the ML model

Objective 1:

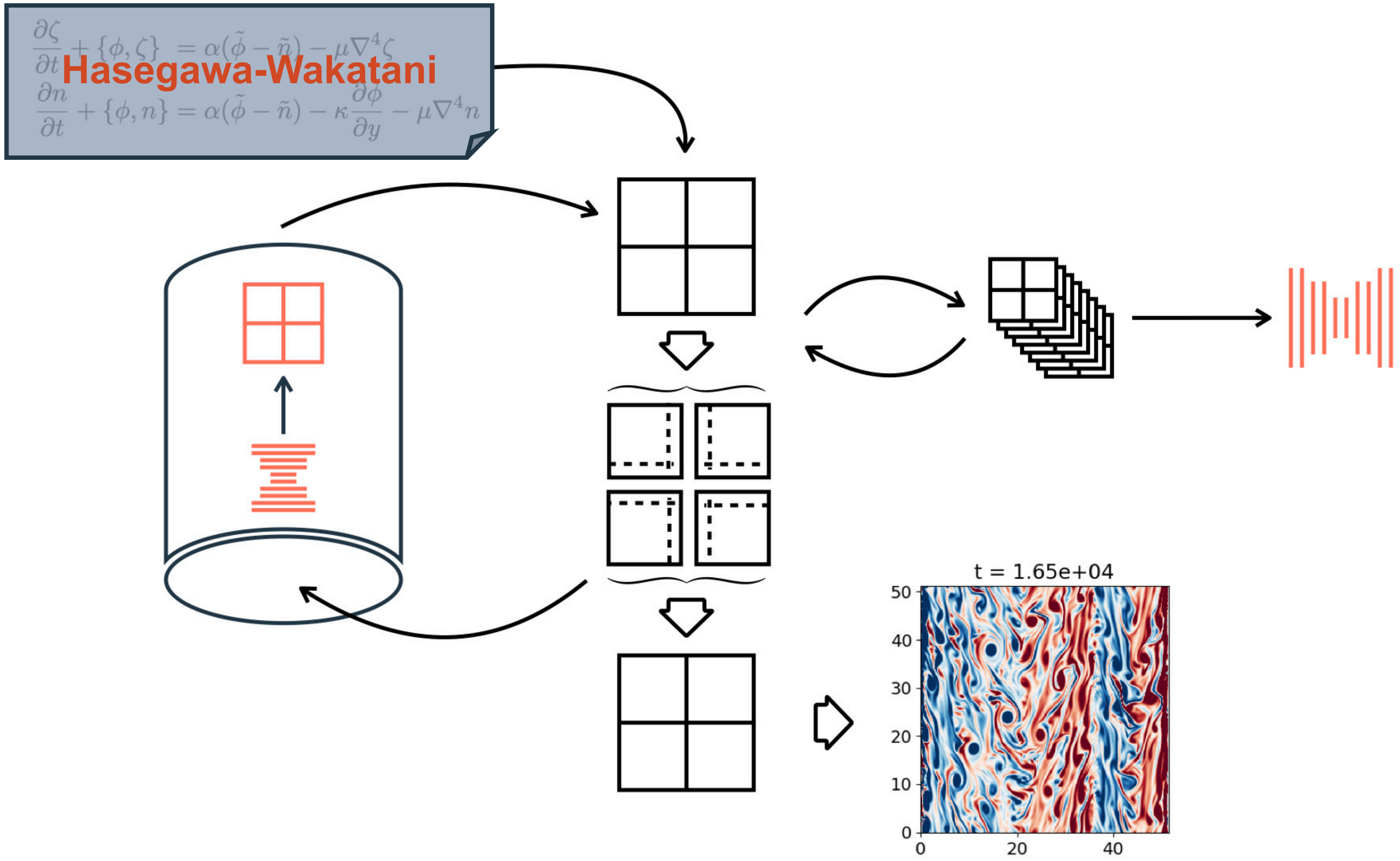
Develop a library to provide the infrastructure

Identified tools/technologies

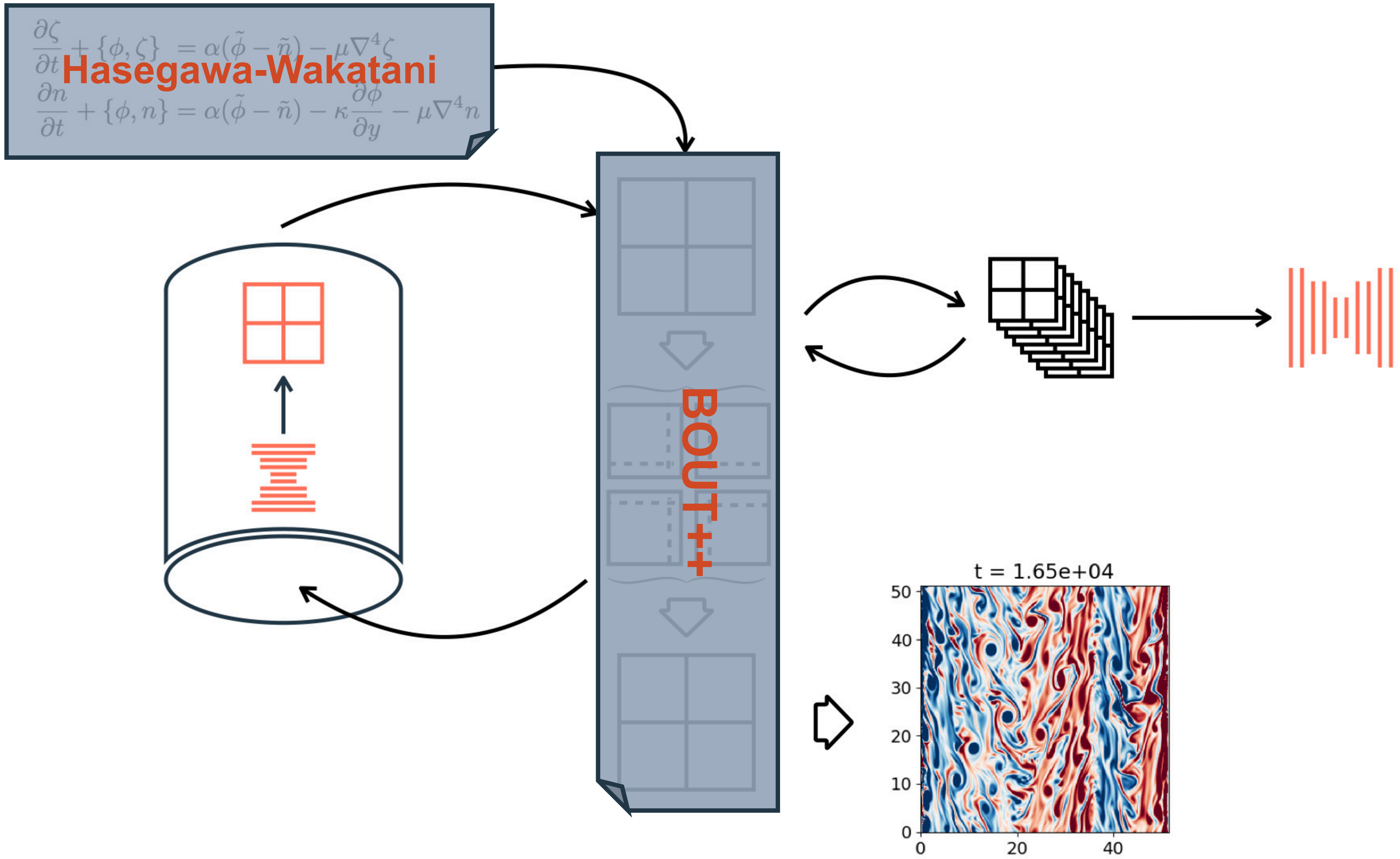
$$\frac{\partial \zeta}{\partial t} + \{\phi, \zeta\} = \alpha(\tilde{\phi} - \tilde{n}) - \mu \nabla^4 \zeta$$
$$\frac{\partial n}{\partial t} + \{\phi, n\} = \alpha(\tilde{\phi} - \tilde{n}) - \kappa \frac{\partial \phi}{\partial y} - \mu \nabla^4 n$$



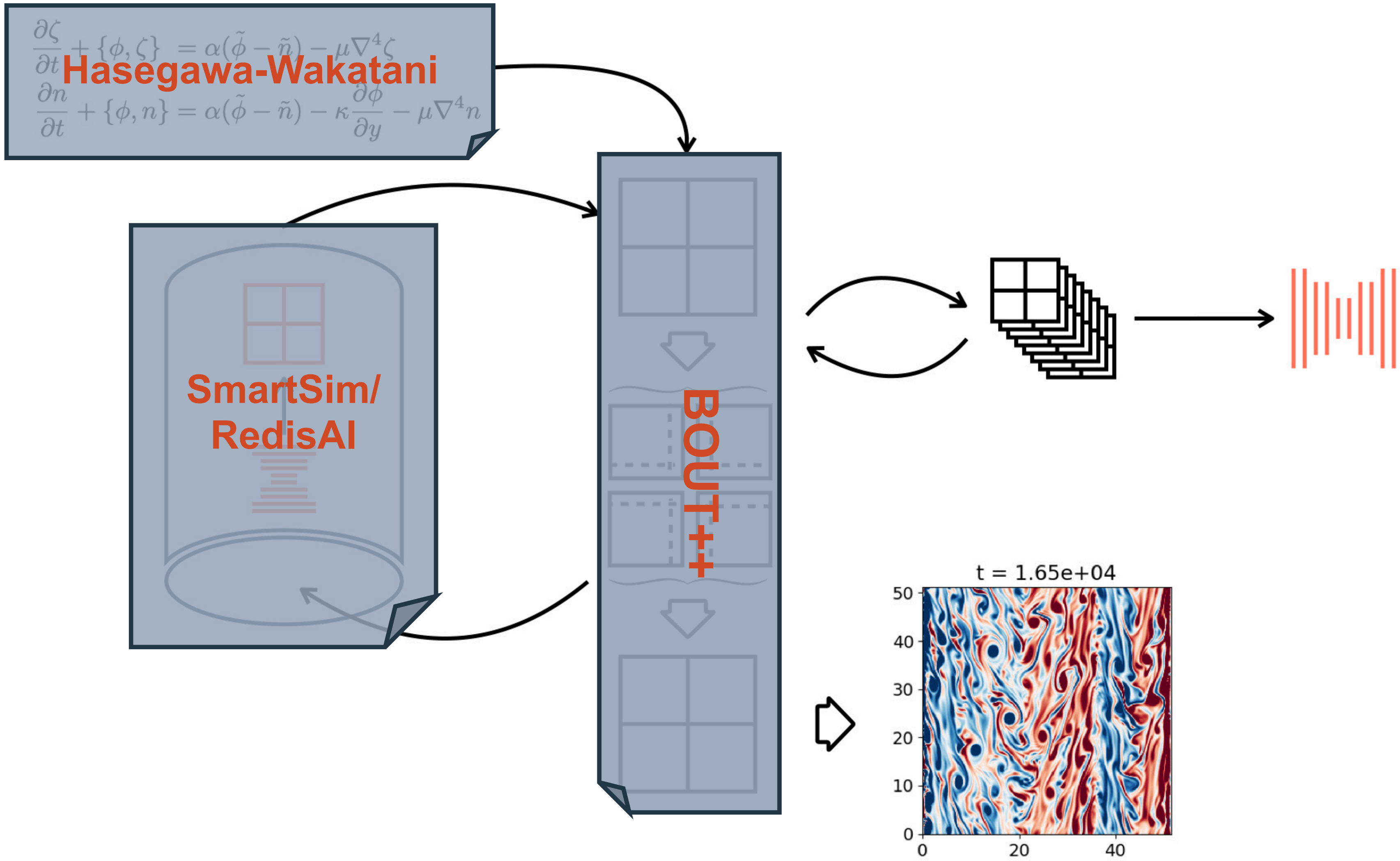
Identified tools/technologies



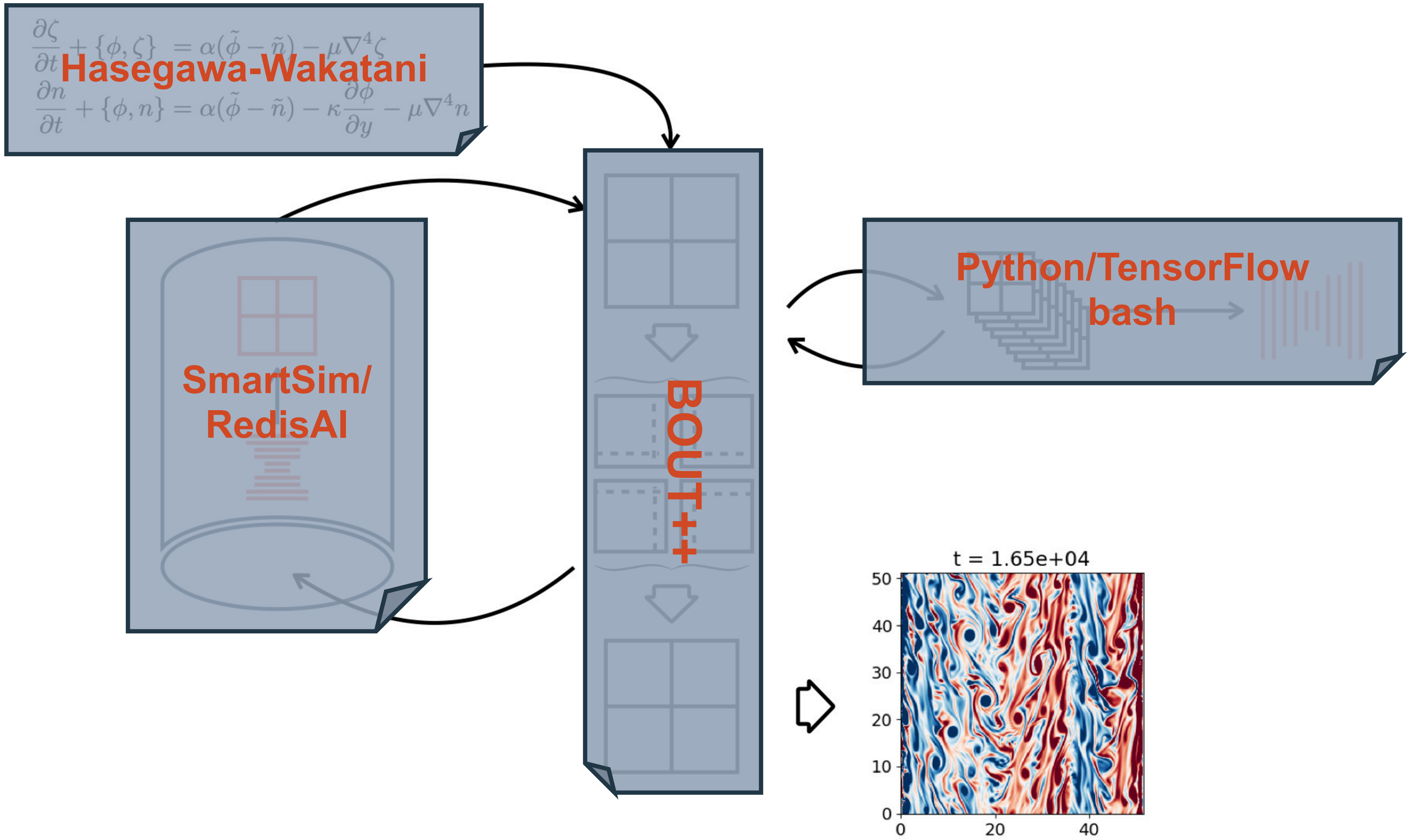
Identified tools/technologies



Identified tools/technologies



Identified tools/technologies



SmartSim

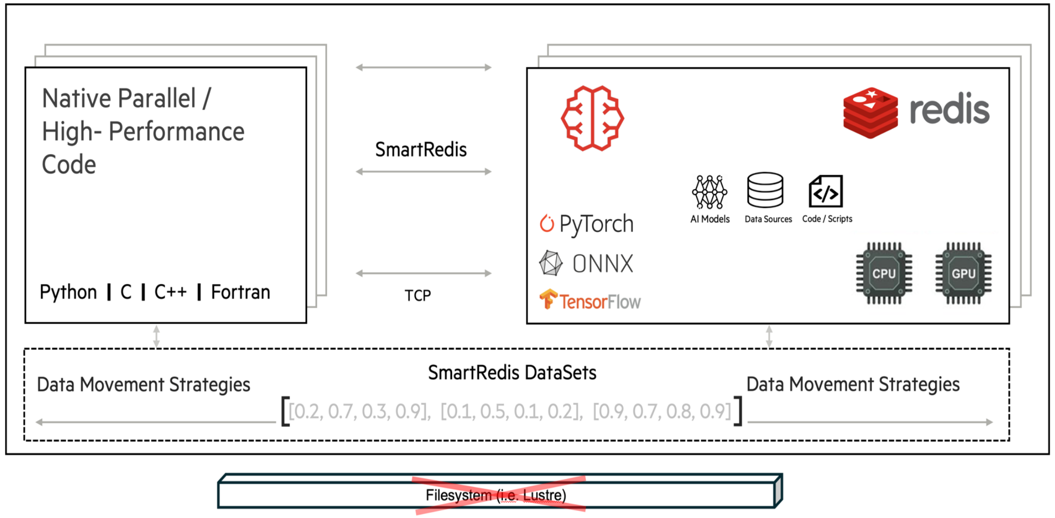
github.com/CrayLabs/SmartSim

- A library to facilitate data exchange between HPC simulations and Machine Learning workflows

- Two main components:

  1. **SmartSim**: The infrastructure to call Python ML models from Fortran, C and C++ (and Python)

  2. **SmartRedis**: A collection of clients to exchange data (tensors, models) with a distributed, in-memory database (Redis cluster) across multiple compute nodes



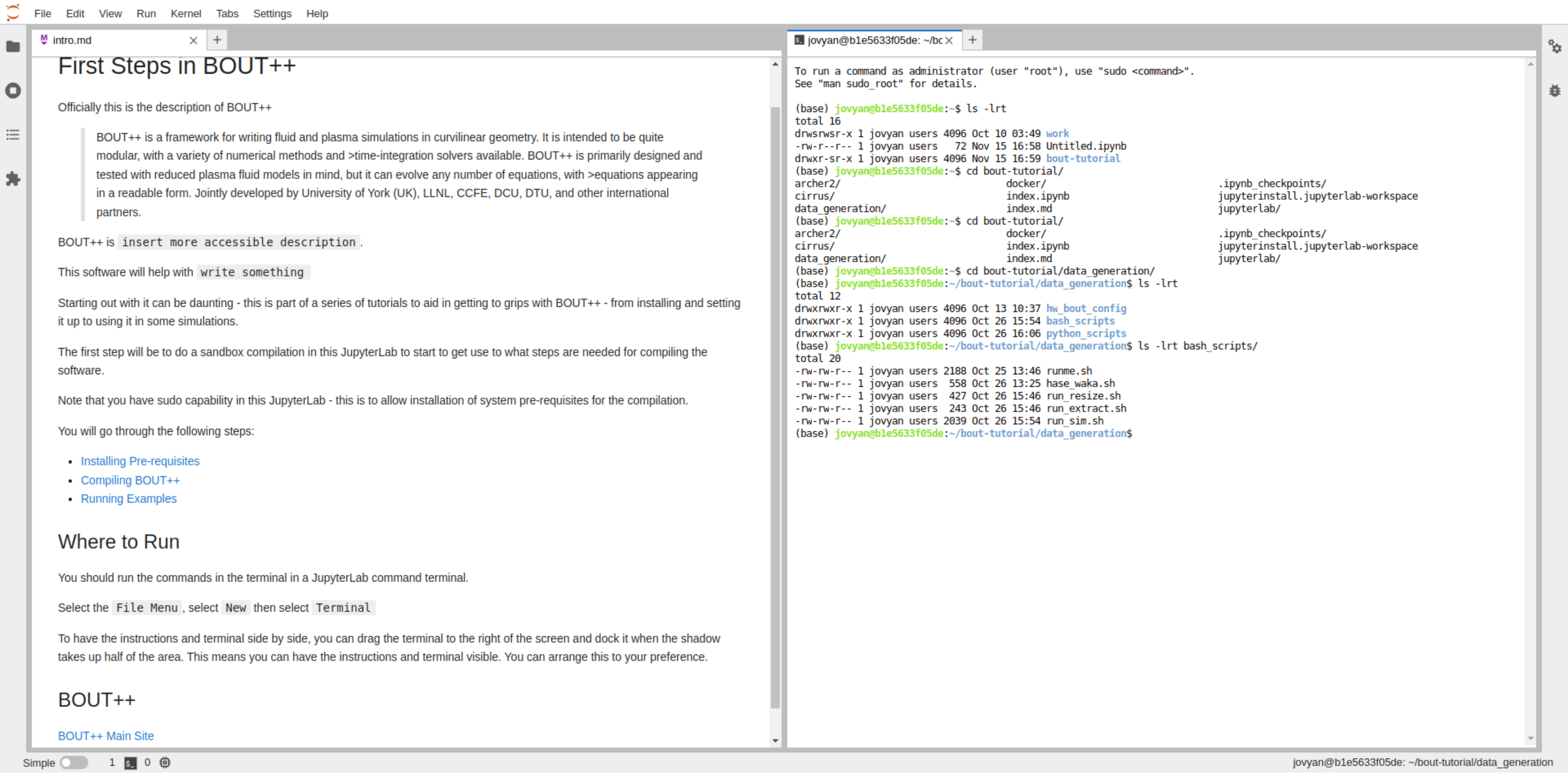
From: https://www.craylabs.org/docs/overview.html

- Provides uniform API for pulling/pushing simulation data from C, C++ and Fortran at run-time

- Orchestration ability to coordinate the runs of simulation and machine learning in distributed environment

Our Tutorials

(local and remote/HPC)



1. Training container (docker) with a toy example to run locally, on personal machines

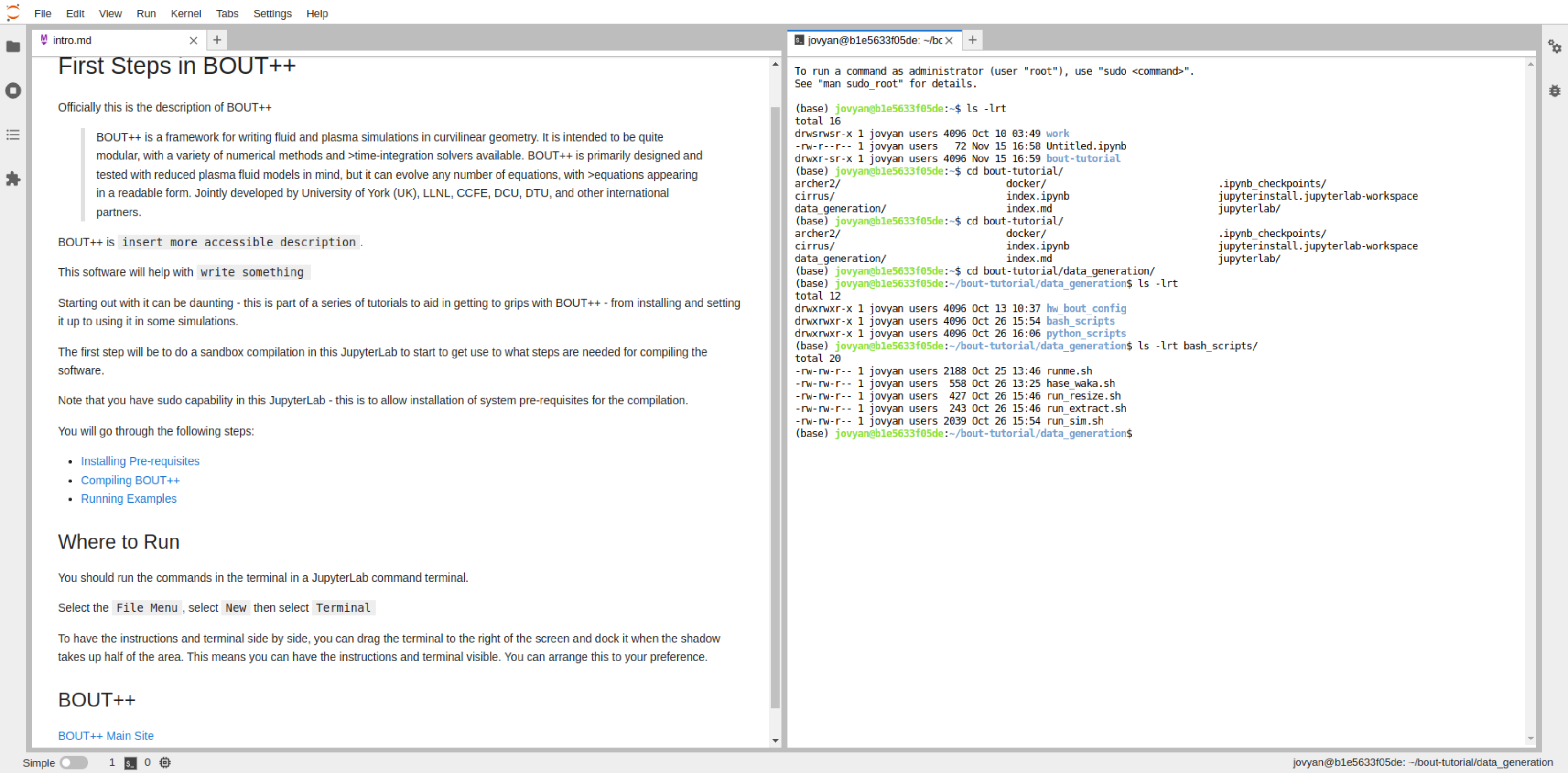
1. Remote visual environment, backed by UK HPC services

1. Prototype and a hands-on workshop in 2023

Our Tutorials

(local and remote/HPC)

Trial available on request



1. Training container (docker) with a toy example to run locally, on personal machines

1. Remote visual environment, backed by UK HPC services

1. Prototype and a hands-on workshop in 2023

Collaborations and links

1. BOUT++ users with simulations

1. Similar problems/simulations, other numerical solvers

1. Machine Learning specialists and Data Scientists
   – errors, visualisations, uncertainty, verification/validation of models, …

simlint@mlist.is.ed.ac.uk

Thanks for listening!

Questions?

References

- D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner and S. Hoyer: **Machine learning–accelerated computational fluid dynamics.** Proceedings of the National Academy of Sciences, 118(21), 2021

- Maulik, R., San, O., Rasheed, A., & Vedula, P. (2019). **Subgrid modelling for two-dimensional turbulence using neural networks.** Journal of Fluid Mechanics, 858, 122-144. doi:10.1017/jfm.2018.770

- P. Watson: **Applying machine learning to improve simulations of a chaotic dynamical system using empirical error correction.** Journal of Advances in Modeling Earth Systems, May 2019.

- M. Wakatani and A. Hasegawa: **A collisional drift wave description of plasma edge turbulence.** The Physics of fluids, 27(3):611–618, 1984

- A. Hasegawa and M. Wakatani. **Self-organization of electrostatic turbulence in a cylindrical plasma.** Phys. Rev. Lett., 59:1581–1584, Oct 1987.

- TensorFlow: `tensorflow.org`
  Keras: `keras.io`

- S. Partee, M. Ellis, A. Rigazzi, S. Bachman, G. Marques, A. Shao, and B. Robbins: **Using machine learning at scale in HPC simulations with smartsim: An application to ocean climate modeling.** CoRR, abs/2104.09355, 2021.
  `github.com/CrayLabs/SmartSim`