Tools and techniques for modular, portable, (machine learning) parameterisations

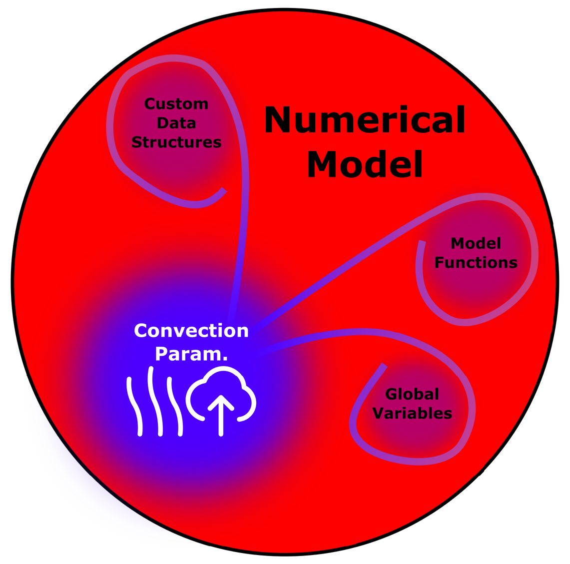
Jack Atkinson[1], Dominic Orchard[1,2], Elliott Kasoar[1,3], Tom Meltzer[1]

1) ICCS, University of Cambridge 2) University of Kent 3) STFC

Current State

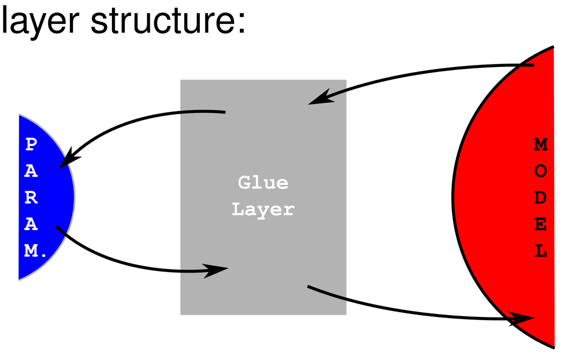
Current implementations of parameterisations in numerical models have common attributes that form a barrier to interoperability:

- Use of custom data structures
  - Will not exist in other models
  - May contain unnecessary variables/data
- Use of global variables
  - No reference from another model
  - Hard to track down within other models
  - Generally a bad idea in software design
- Use of functions from elsewhere in the model
  - now need to be ported alongside the parameterisation



Proposed structure/rules

Our proposed solution to the above issues centres around parameterisations being implemented with a multi-layer structure:

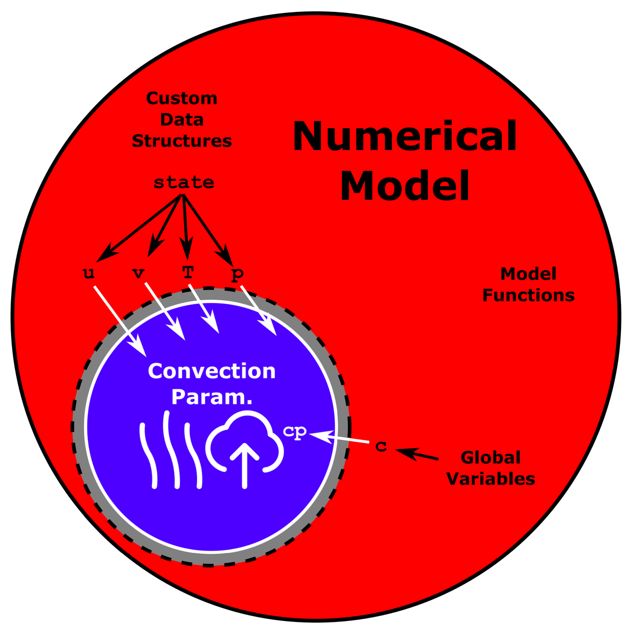


Rules for authors

- Do not assume any data structures at the interface to the numerical model
  - Take all variables as single inputs.
  - These could be used to create data structures *inside* the parameterisation.
- Do not assume any global variables
  - all variables used in the parameterisation must be explicitly passed across the parameterisation-model interface.
- Provide a clear API
  - specify all input and output variables
  - specify units for all input and outputs
  - specify appropriate ranges for inputs and outputs e.g. resolution
- Publish code
  - Version controlled, ideally open-source
  - Provide tests for users/developers

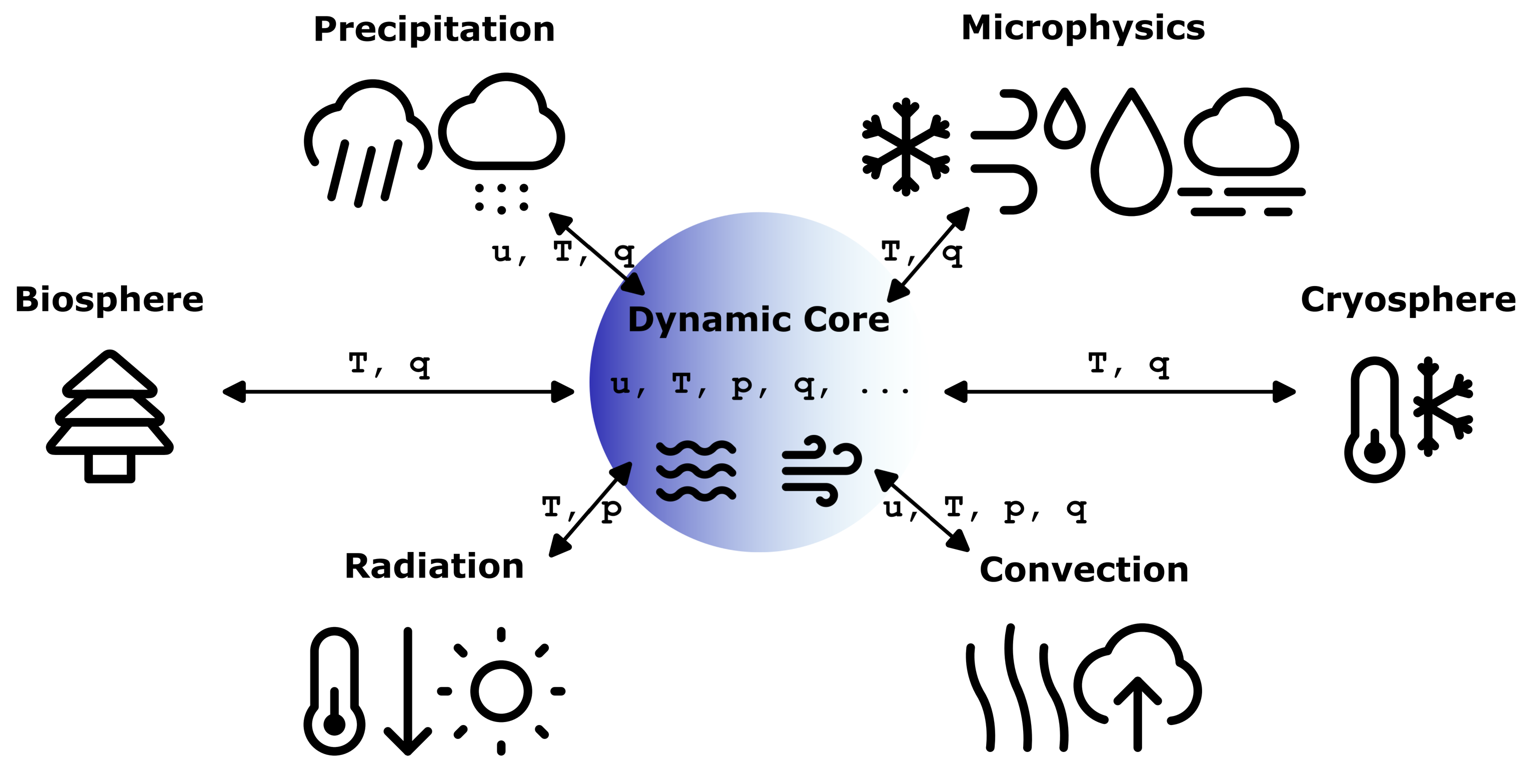
Rules for users

- Provide clear API/documentation for the model variables, their units, and any data structures
- Coupling of models should be done through a **Glue code/interface layer**:
  - Used to join the two defined interfaces (parameterisation and model).
  - Conversion between model and parameterisation variables.
  - Conversion between model and parameterisation units.



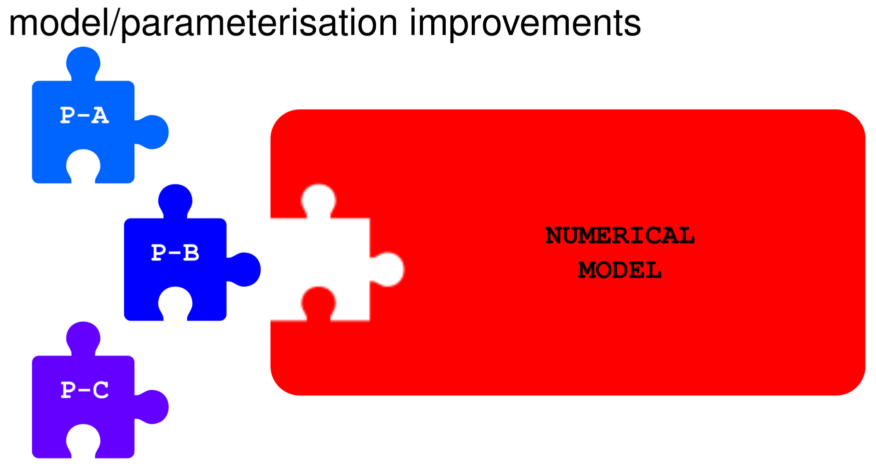
Motivation

Climate models rely on parameterisations for subgrid processes.



These are often published as standalone papers, documented and described with physical equations.
If authors made the code implementation of their parameterisations available this would help with:

- *re-use* and deployment in a variety of models.
- *reproducibility*
- collaboration and model/parameterisation improvements



In reality implementations of parameterisations are tightly coupled (in terms of code/software) to any model they are deployed in. This presents barriers to interoperability, re-use, and scientific progress.

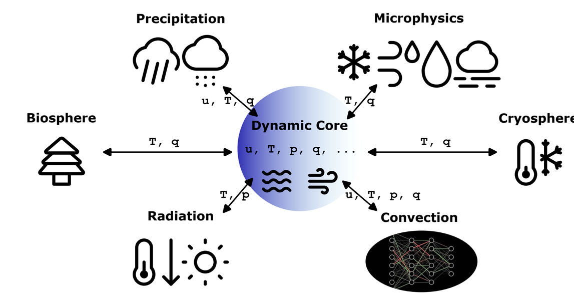
Vision

- Parameterisations should strive to be model-agnostic i.e. interoperable
- This can be aided by sensible software design.
- Along with code authors should provide a clear API for variables in and out.
- Re-deployment to a new model then requires end users to write a model interface layer
- ML parameterisations bring additional challenges.
  We propose adopting a NN core wrapped by a physics layer and model interface later.
- We have developed *FTorch* to facilitate PyTorch-Fortran coupling and integrated it into CESM.

Machine Learning considerations

Machine-learnt (ML) parameterisations provide potential for higher performance and/or accuracy by training using observational data or high-resolution model output.

However, implementation of such parameterisations brings additional challenges beyond those discussed to the left.

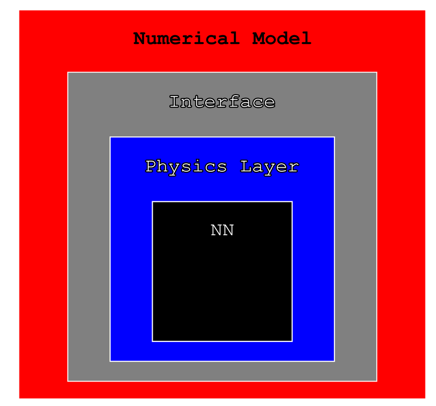


- Grid restrictions
  ML parameterisations are bound to the grid on which they are trained.
  Users will now need to perform a grid interpolation as part of the interface.
- Hybrid architectures
  It is desirable to run ML on GPU, whilst most numerical models are CPU-based so offloading will be required.
  Efficient data transfer and GPU usage will require MPI Gather.
- Language interoperation
  Most ML models are trained in Python frameworks e.g. PyTorch whilst many numerical models use Fortran or similar compiled languages.
  Joining the two is not straightforward.

Structure

We propose that ML parameterisations adopt a nested structure:

- A pure neural net (NN) core
  - to allow easy substitution of re-trained/different architecture nets
- A physics wrapper
  - to pre/post-process variables for the NN
  - to enforce physical constraints e.g. mass and energy conservation



FTorch Coupling Library [1]

To tackle some of these issues we have developed *FTorch*, a library for coupling PyTorch models to Fortran code.
Key features include:

- A "pythonic" Fortran interface for users
- Multi-GPU support using Torch offloading
- Efficient, no-copy data transfer
- No requirement for Python at runtime

FTorch is available on GitHub:

https://github.com/Cambridge-ICCS/FTorch

Examples and applications

We have applied these approaches to implement ML parameterisations in CAM (the Community Atmosphere Model) for:

- Deep convection and precipitation, trained using the high-resolution (LES) model SAM (System for Atmospheric Modeling) [4] re-deployed in CAM.
- Atmospheric gravity waves, trained using the high-resolution WRF (Weather Research and Forecasting) Model [2] re-deployed in CAM.

Both of these have been run on multiple High Performance Computing (HPC) systems.
FTorch has been implemented as part of the CIME framework to facilitate use by wider users of CESM (the Community Earth System Model).

KEY REFERENCES

[1] ICCS. FTorch. https://github.com/Cambridge-ICCS/FTorch, 2022.

[2] Y Qiang Sun, Pedram Hassanzadeh, M Joan Alexander, and Christopher G Kruse. Quantifying 3d gravity wave drag in a library of tropical convection-permitting simulations for data-driven parameterizations. *Journal of Advances in Modeling Earth Systems*, 15(5):e2022MS003585, 2023.

[3] Y Qiang Sun, Hamid A Pahlavan, Ashesh Chattopadhyay, Pedram Hassanzadeh, Sandro W Lubis, M Joan Alexander, Edwin Gerber, Aditi Sheshadri, and Yifei Guan. Data imbalance, uncertainty quantification, and generalization via transfer learning in data-driven parameterizations: Lessons from the emulation of gravity wave momentum transport in waccm. *arXiv preprint arXiv:2311.17078*, 2023.

[4] Janni Yuval, Paul A O'Gorman, and Chris N Hill. Use of neural networks for stable, accurate and physically consistent parameterization of subgrid atmospheric processes with good performance at reduced precision. *Geophysical Research Letters*, 48(6):e2020GL091363, 2021.

ACKNOWLEDGEMENT AND PARTNERS

SCHMIDT SCIENCES

EGU General Assembly #EGU24 14-19 April 2024

MORE INFORMATION

Jack Atkinson
Senior Research Software Engineer
Research Software Group
Institute of Computing for Climate Science
University of Cambridge
jwa34@cam.ac.uk

Sharing is encouraged