

UKAEA – Jan 30-31 2024

ReMKiT1D Workshop January 2024

Introduction to ReMKiT1D

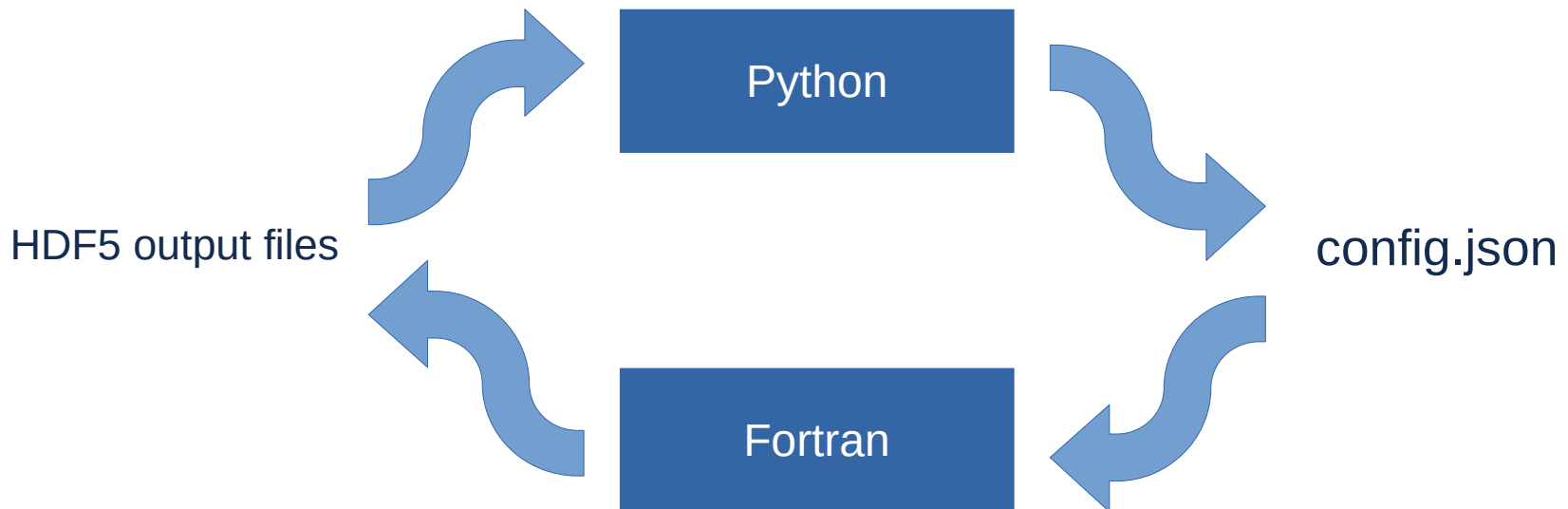
Imperial College
London

What is ReMKiT1D?

Reactive Multifluid and Kinetic Transport in 1D

A flexible framework for solving systems of differential equations, focused on applications in 1D Scrape-Off Layer physics

Written in Modern Fortran with a Python library for configuring runs



What is ReMKiT1D?

Which types of equations do we solve?

General nonlinear systems of ODEs $\frac{d\vec{v}}{dt} = \mathbf{M}(\vec{v})\vec{v} + \vec{\Gamma}$

1D PDEs $\frac{\partial X(\vec{x})}{\partial t} + \nabla \cdot \vec{\Gamma}_X(\vec{x}) = S_X, \quad \Delta \varphi = f$


Electron kinetic equations (in a Legendre basis) $\frac{\partial f}{\partial t} + v_x \frac{\partial f}{\partial x} - \frac{eE}{m_e} \frac{\partial f}{\partial v_x} = \left(\frac{\delta f}{\delta t} \right)_c,$

$$\frac{\partial f_l}{\partial t} = A_l + E_l + C_l$$


Surface level concepts

Which concepts are central to describing our problems?


$$\frac{d\vec{v}}{dt} = M(\vec{v}) \cdot \vec{v} + \vec{\Gamma}$$

 Variables – these are the values we are evolving/calculating

 Terms – these will determine the evolution of our variables

 Time integration – how do we march forward in time in our simulation?

$$\frac{\partial X(\vec{x})}{\partial t} + \nabla \cdot \vec{\Gamma}_X(\vec{x}) = S_X$$

 Grid – Some terms might need grid information, and variables need to live somewhere

ReMKiT1D – Variables

Variables represent discretized data

Requirements on the variable data structure:

- We want to be able to treat fluid variables, distributions, and scalars
- We want to be able to use implicit time integration
- We want to be able to calculate variables using non-integration rules

Variable categorisation

Dimensionality

- Fluid (1D)
- Distribution (1D2V)
- Scalar (0D)

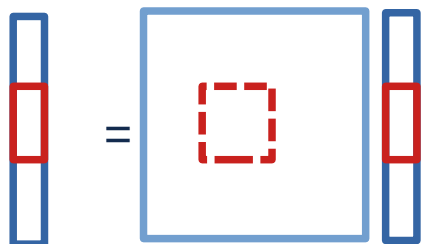
Implicitness

- Implicit
- Derived

ReMKiT1D – Variables

Implicit vs derived variables

Implicit – know how to index themselves into a global vector for PETSc matrix solves



$$\frac{d\vec{v}}{dt} = \mathbf{M}(\vec{v}) \vec{v}$$

Derived – instead of indexing into the global vector they can have a **derivation rule** attached to them

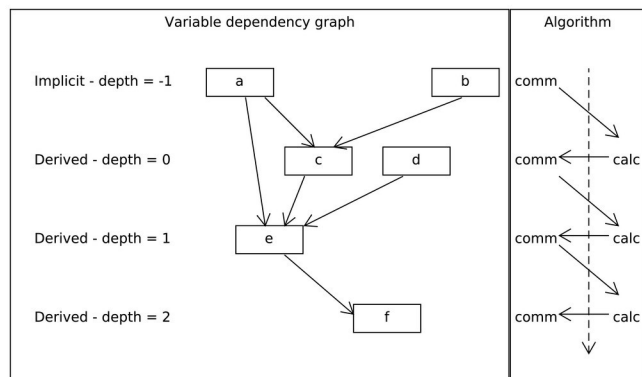
Effectively function wrappers

Derived Example:

Let n and f be density and flux – any (implicit or derived) variables
Then u (a flow speed) can be derived with the derivation $\text{var}_1/\text{var}_2$ with $\text{var}_1=f$ and $\text{var}_2=n$

Communication-safe derivation algorithm

The code will make sure any variables required in a derivation are calculated and MPI-communicated in time



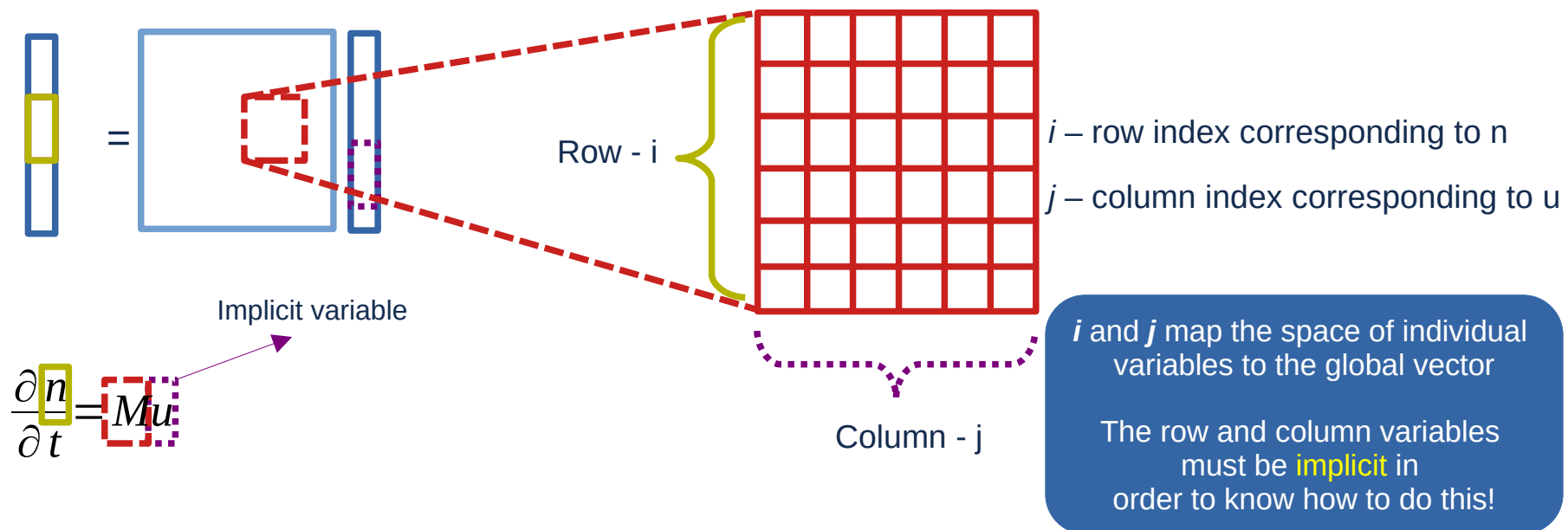
ReMKiT1D – Terms

Term objects represent individual additive terms in equations of the type

$$\frac{\partial n}{\partial t} = S_1 + S_2 + \dots$$

Evolved variable

General matrix terms are currently available to user (general explicit terms in development)



ReMKiT1D – Terms

$$\frac{\partial n}{\partial t} = M u$$

Example (a little convoluted):

$$\frac{\partial n}{\partial t} = 5 \sin(t) x^2 w \nabla (n u)$$

What's M here?

$$M_{ij} = 5 \sin(t) x_i^2 w_i S_{ij} n_j$$

In general:

$$M_{ij} = c X_i(x) H_i(h) V_i(v) T(t) R_i C_j S_{ij}$$

Constants
(normalization)

Coordinate and
time profiles

Variable row and
column functions

Stencil matrix – in general
depends on variables/data

Chosen from a list of
customizable templates

Assuming a central difference
stencil for the differential operator

Row - i

Column - j

ReMKiT1D – Terms

What about something like this? $\frac{d\vec{v}}{dt} = \mathbf{M}(\mathbf{v}) \cdot \vec{v}$

$$\frac{\partial n_1}{\partial t} = f_{11}(n_1, n_2, \dots, n_n) n_1 + f_{12}(n_1, n_2, \dots, n_n) n_2 + \dots + f_{1n}(n_1, n_2, \dots, n_n) n_n$$

$$\frac{\partial n_2}{\partial t} = f_{21}(n_1, n_2, \dots, n_n) n_1 + f_{22}(n_1, n_2, \dots, n_n) n_2 + \dots + f_{2n}(n_1, n_2, \dots, n_n) n_n$$

⋮

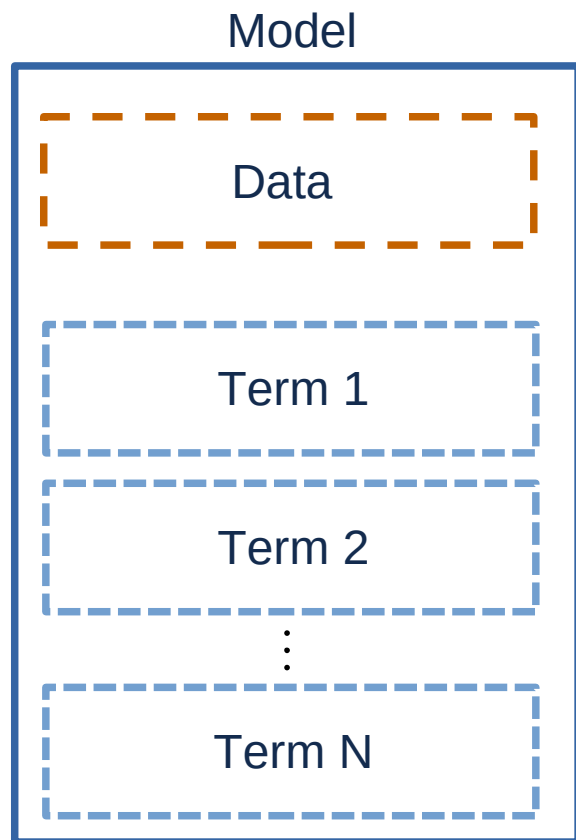
$$\frac{\partial n_n}{\partial t} = f_{n1}(n_1, n_2, \dots, n_n) n_1 + f_{n2}(n_1, n_2, \dots, n_n) n_2 + \dots + f_{nn}(n_1, n_2, \dots, n_n) n_n$$

That's a lot of terms and data needed to calculate them!

The above situation is very common in Collisional-Radiative **Models**

ReMKiT1D – Models

Models are collections of (related) terms and (optional) data



From the Python interface:

We can select individual term components to build terms

We collect terms into models

We can attach data to models

Models and terms are used to specify the time integration

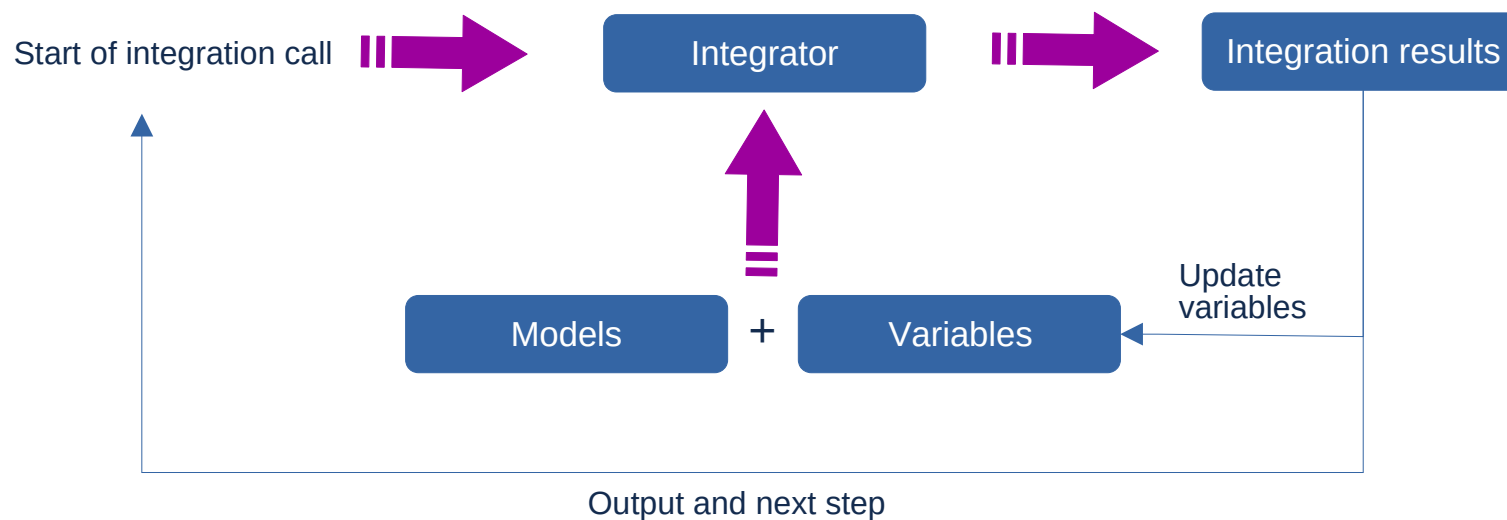
ReMKiT1D – Time integration

With variables and terms/models defined, how do we use them?

We provide a discretization in time through defining an integrator

Integrators are highly flexible (more on that later) and perform time-stepping for us – think RK4 or Backwards Euler

Simplified integration loop



ReMKiT1D – Time integration

In this workshop we will be using the Backwards Euler integrator

$$\frac{v_i^{k+1} - v_i^k}{\Delta t_k} = M_{ij}^{k'} v_j^{k+1}$$

Evaluated at previous non-linear iteration

Non-linear terms are handled using fixed point iterations

Iterations are repeated until some selected variables converge

More on this in the hands-on sessions

ReMKiT1D – Spatial grid

We need a grid for variables to live on and in order to consistently define some of the differential operators!

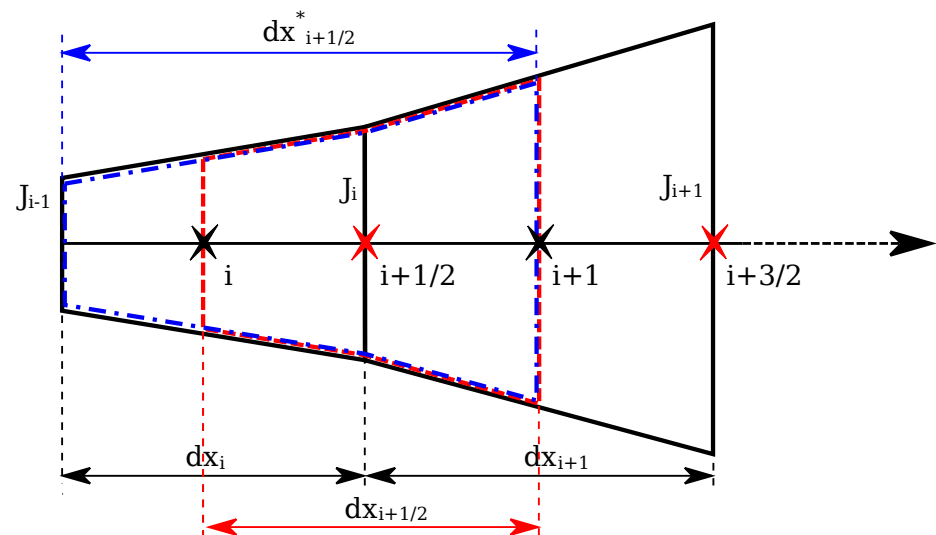
By default ReMKiT1D supports Finite Volume methods using staggered grids

Variables live on either the regular or the dual/staggered grid and can be interpolated on the other grid

Regular grid – cell centres

Dual/staggered grid – cell edges

Cells can have faces of different areas
(natural implementation of flux expansion)

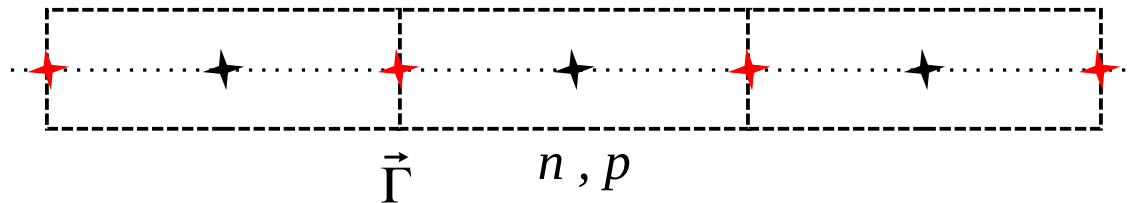


ReMKiT1D – Spatial grid

Staggered grid example – Finite Volume reminder:

$$\frac{\partial n}{\partial t} + \nabla \vec{\Gamma}(x) = 0, \quad m \frac{\partial \vec{\Gamma}}{\partial t} + \nabla p = 0$$

Where do we put our variables?

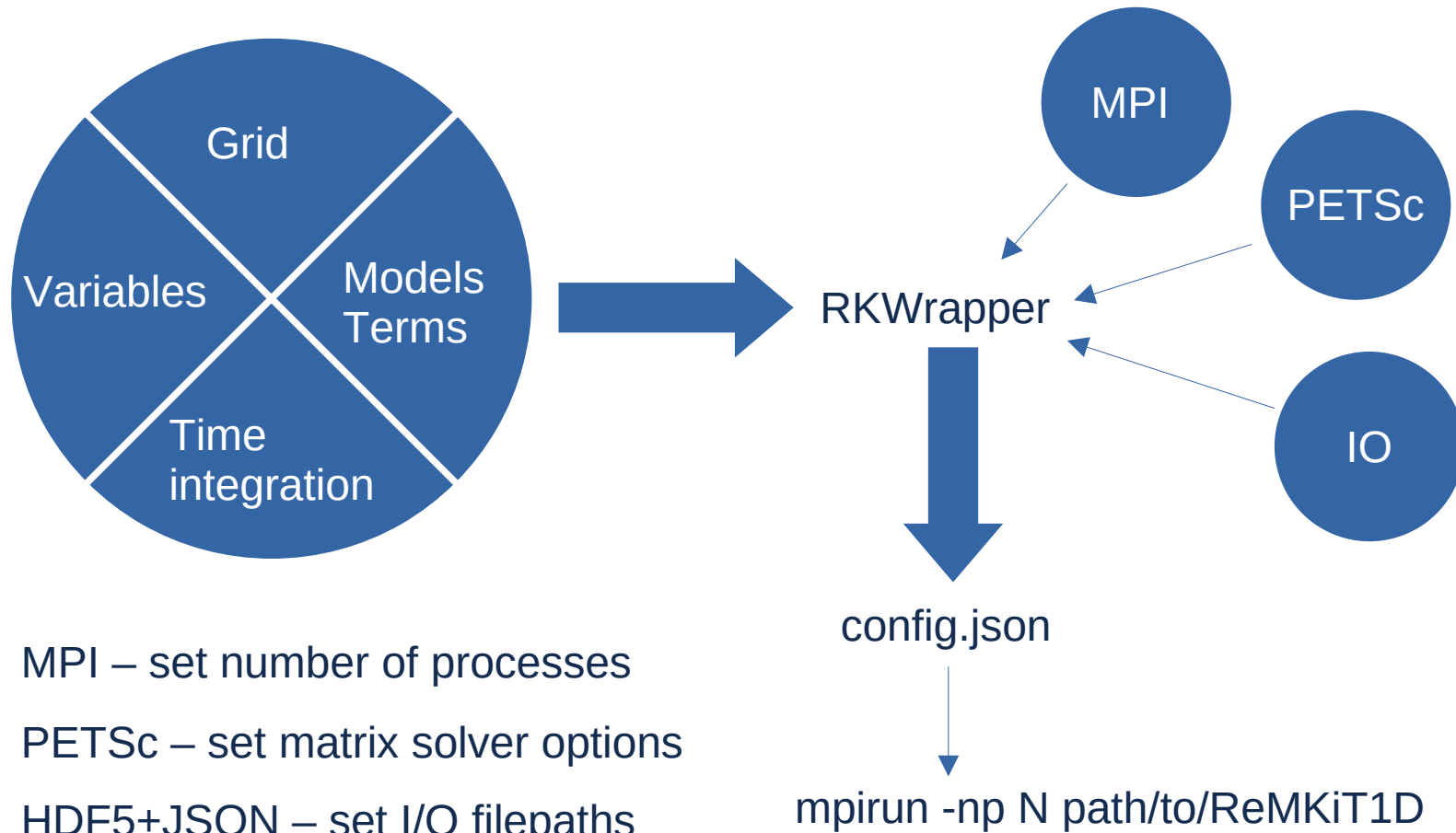


A good rule of thumb:

- Scalar-like variables live in cell centers
- Vector-like variables live on cell edges

ReMKiT1D – Configuration and I/O

We set the 4 main shallow concepts using the Python package RMK_Support



Hands-on session