Advanced Quantification of Uncertainties In Fusion modelling at the Exascale with model order Reduction (AQUIFER)

UCL

Work done to date

NEPTUNE Workshop, 5-6 September 2022, Abingdon, UK

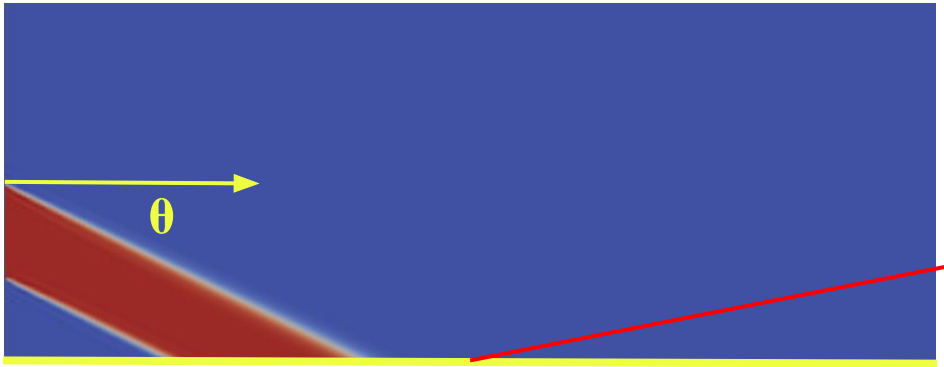
Surrogates plus Model Order Reduction

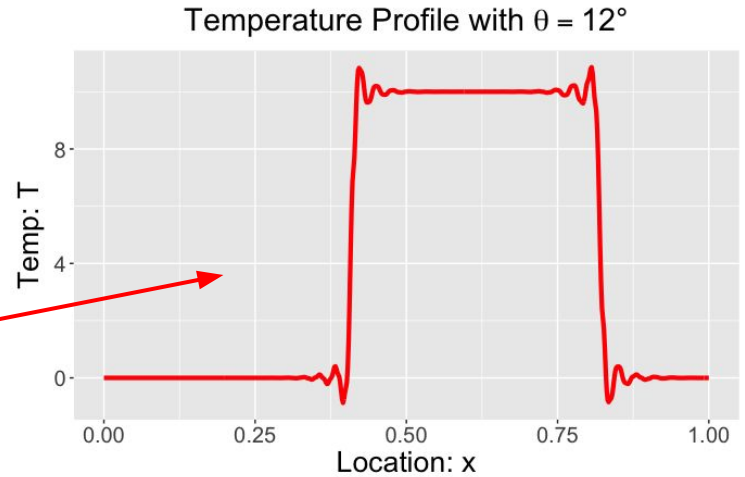
Functional Emulation for temperature profile over spatial domain

Test Case: 2D Anisotropic Heat Transportation

The 2D model describes the thermal diffusion in a plasma confined by a magnetic field. We construct a emulator to explore the relationship between direction of magnetic field θ and temperature profile T_x over the bottom boundary at steady state.

$$f_{\text{emulator}} : \theta \rightarrow T_x,\ \theta \in \left[0,\ \frac{\pi}{2}\right],\ T_x \in C[0,\ 1]$$





Surrogates plus Model Order Reduction

Outer Product Emulator (OPE):

Formulation:

The OPE creates one emulator for all the simulation outputs over the whole domain and simplifies the representation of fitted functions by products of component functions. In general, the OPE has the form:

$$f_i(r) = \sum_{j=1}^{v} \beta_j g_j(r, s_i) + \epsilon(r, s_i)$$

where $f_i(r)$ is the ith simulation output with input r, the g_j are the regression functions, the β_j are unknown coefficients and ϵ is the residual assumed to be a Gaussian Process (GP) such that $\epsilon \sim GP(0, \kappa_\lambda(\cdot))$.

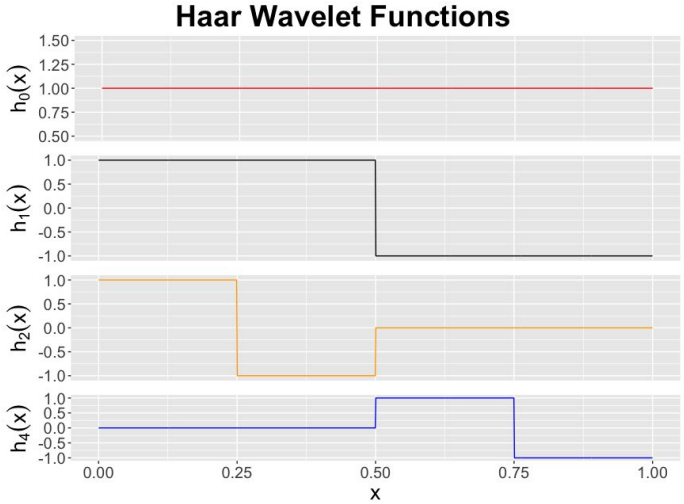
Meta-Level Choices:

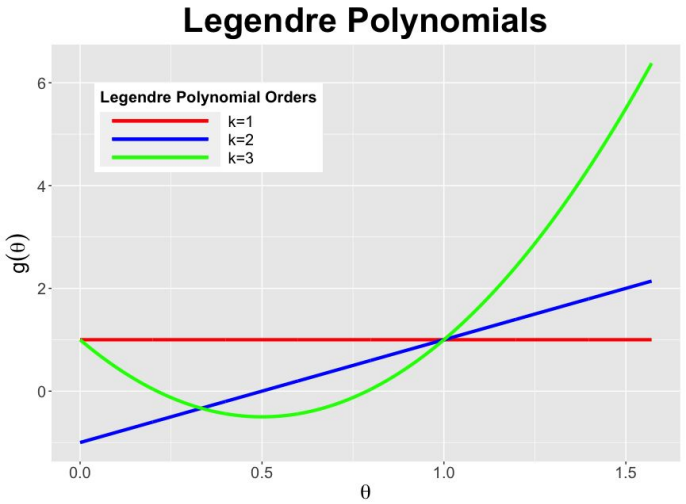
1. **Regression Functions:**
   - **Input Regressors:** $g_j^r(\theta) = \{1, 6\theta - 1, 6\theta^2 - 6\}$ which is the legendre polynomials up to second order
   - **Output Regressors:** $g^s(x) = \{h_j(x)\}_{j=0}^{3}$ where h_j is the jth haar wavelet function

2. **Covariance Functions:**

$$\kappa(x, \theta, x', \theta') = \exp\left(-\left(\frac{|\theta - \theta'|}{\lambda_r}\right)^{\frac{3}{2}}\right) \times \exp\left(-\left(\frac{|x - x'|}{\lambda_s}\right)^{\frac{3}{2}}\right)$$

Surrogates plus Model Order Reduction





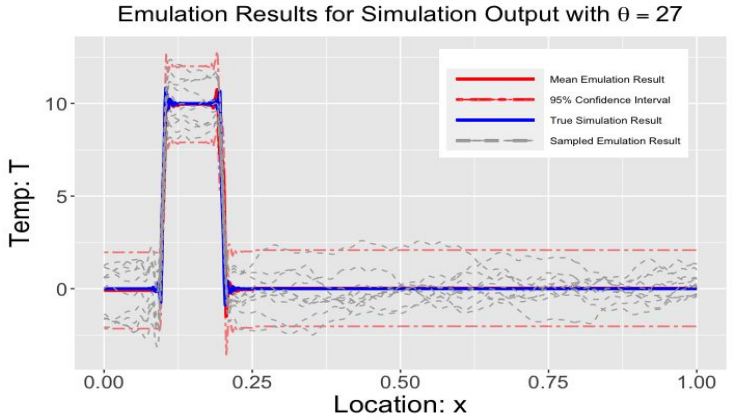
Experiment Settings:

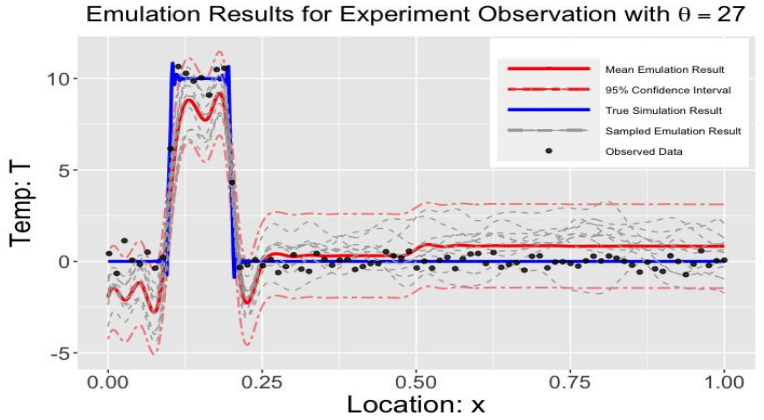
1. **Emulation for Simulation Output**: In this case, we fit the OPE with all the discretized values over bottom boundaries without noise.

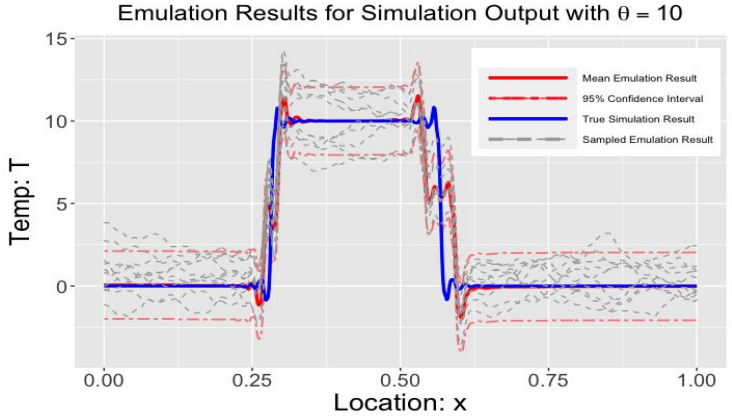
2. **Emulation for Experimental Data**: By contrast, we aimed at reconstructing the temperature profile with limited experimental observations. Here, we fit the OPE with evenly sampled 10% data from all discretized values of simulation added wite noise.

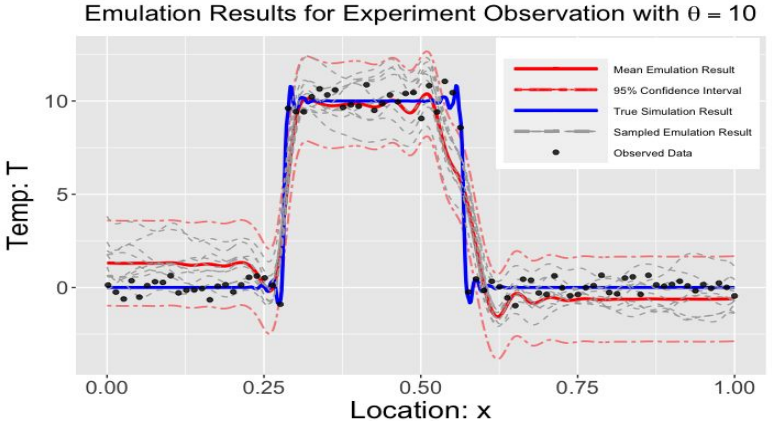
Surrogates plus Model Order Reduction

Results







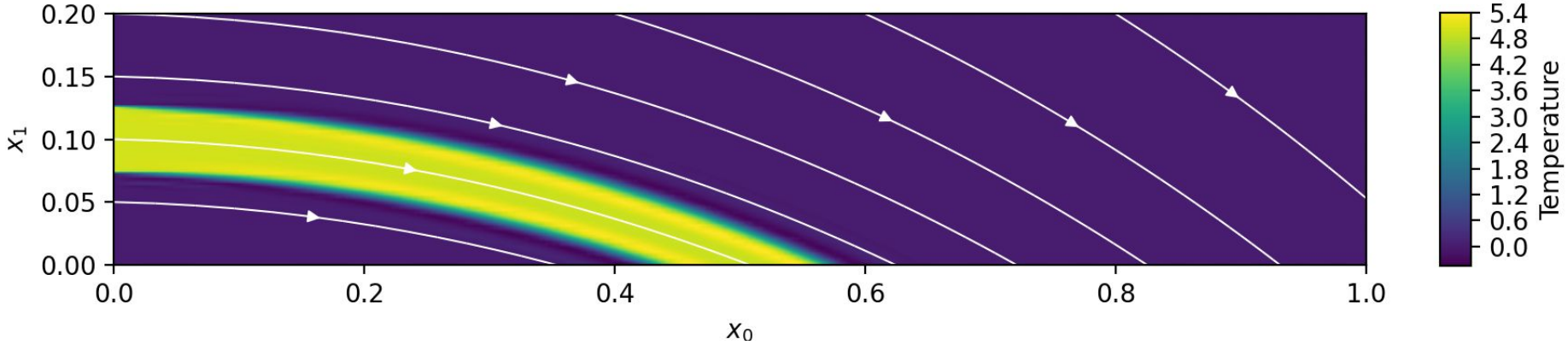


Surrogates plus Model Order Reduction

Summary:

- OPE is an efficient and universal statistical emulator for both high-resolution simulation models and actual experiments
- Domain experts could use their domain-specific knowledge to tailor OPE for interested problems by suitable meta-level choices like regression functions.
- The correlations between input and output are taken into account by the outer products between kernels of input parameter and output values.
- Functional representation for spatial data performs better in capturing the inherent smoothness and variation than discretisation.

Bayesian calibration of anisotropic heat transport



Use 2D anisotropic heat transport model as a test case for calibration.

Consider steady-state solution for temperature of magnetized plasma under a magnetic field with field angle linearly varying from left to right of domain.

Finite-element discretisation on a 80×16 quadrilateral mesh with degree-4 Lagrange elements using FEniCSx.

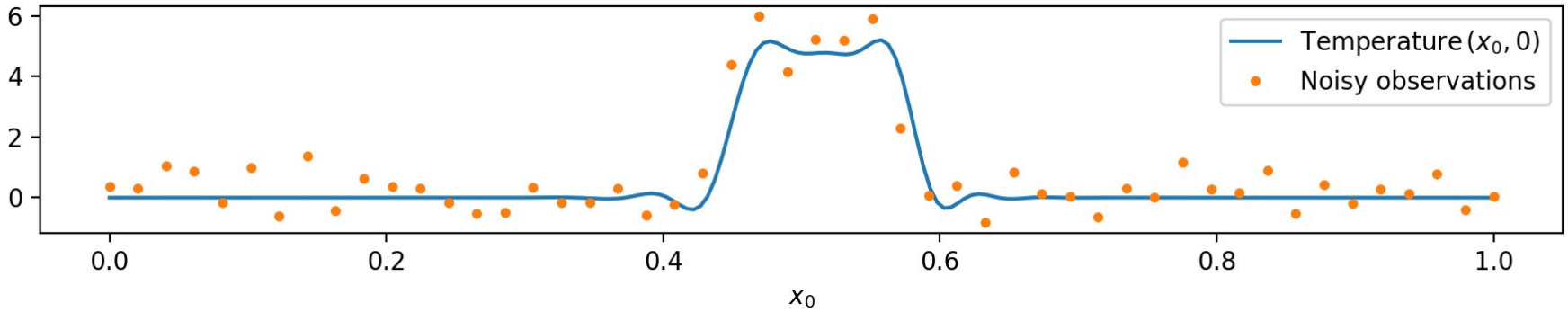
Bayesian formulation of problem

Parameters to infer - field angles at left θ_L and right θ_R boundaries, temperature pulse magnitude on left boundary η, observation noise scale σ, with priors

θ_L ~ Uniform(-π/3, 0), θ_R ~ Uniform(-π/3, 0), η ~ Uniform(0.1, 10), σ ~ LogNormal(0, 1).

Aim: infer posterior distribution on (θ_L, θ_R, η, σ) given N = 50 noisy observations

y_i ~ Normal(Temperature((i−1) / (N−1), 0), σ) $\quad \forall\ i \in 1{:}N$



Approximating posterior using MCMC

To simplify inference, reparameterize in terms of $\boldsymbol{q}$ ~ MultivariateNormal($\mathbf{0}$, $\mathbf{I}$)

θ_L = -(π/3)Φ(q_0), θ_R= -(π/3)Φ(q_1), η = 0.1+ 9.9 Φ(q_2), σ = exp(q_3)

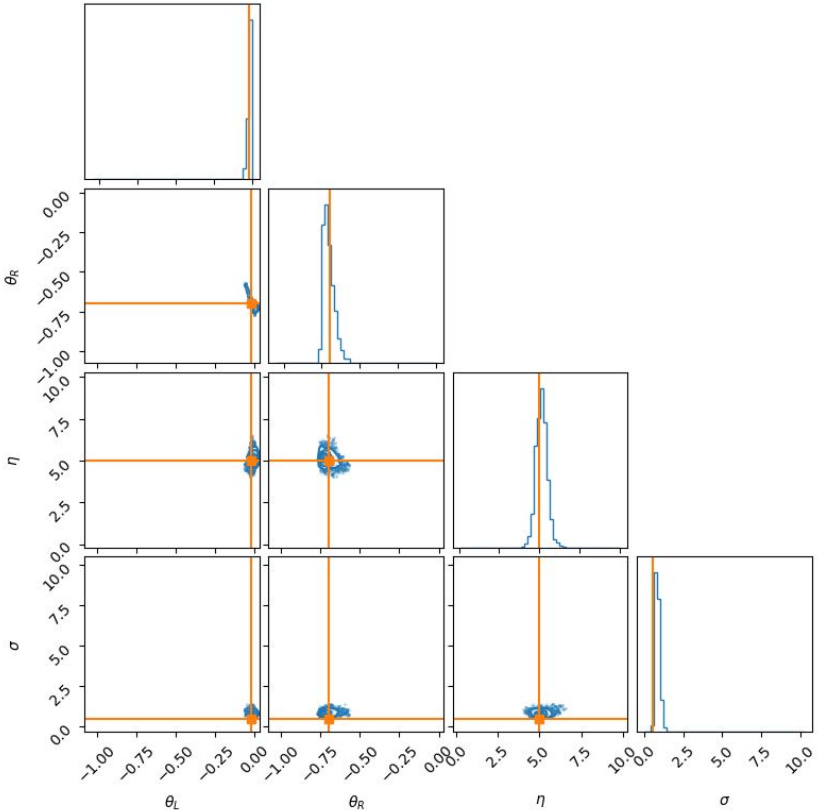
Use a simple random-walk Metropolis algorithm with isotropic Gaussian proposals

$\boldsymbol{q}'$ ~ MultivariateNormal($\boldsymbol{q}$, $\epsilon^2\mathbf{I}$)

with step size ϵ > 0 adaptively tuned to give average acceptance rate 0.234.

Markov chains simulated using Python package *Mici*

MCMC estimates of posterior distribution



Estimated posterior (blue) concentrates around parameters used to simulate observations (orange).

Two chains of 1000 warm-up and 5000 main iterations take ~ 60 minutes to compute as solving model each iteration.

Potential for large performance gains by using emulator to reduce number of model solves needed.

Software implementation. HPC deployment

FabSIM for management of runs

Hackathons, webinars to disseminate

Data Assimilation deployment: initial scalability

Motivation

FabSim3 aims to:

1. Reduce manual effort by providing flexible automation.
2. Reduce human error by introducing consistency across different activities.
3. Automatically curate a comprehensive environment for each simulation run.
4. Curate essential machine- and application-specific knowledge across the community.
5. And, as a result of 1-4: enable efficient remote access to HPC and other resources using a user-side local tool.

Design Overview

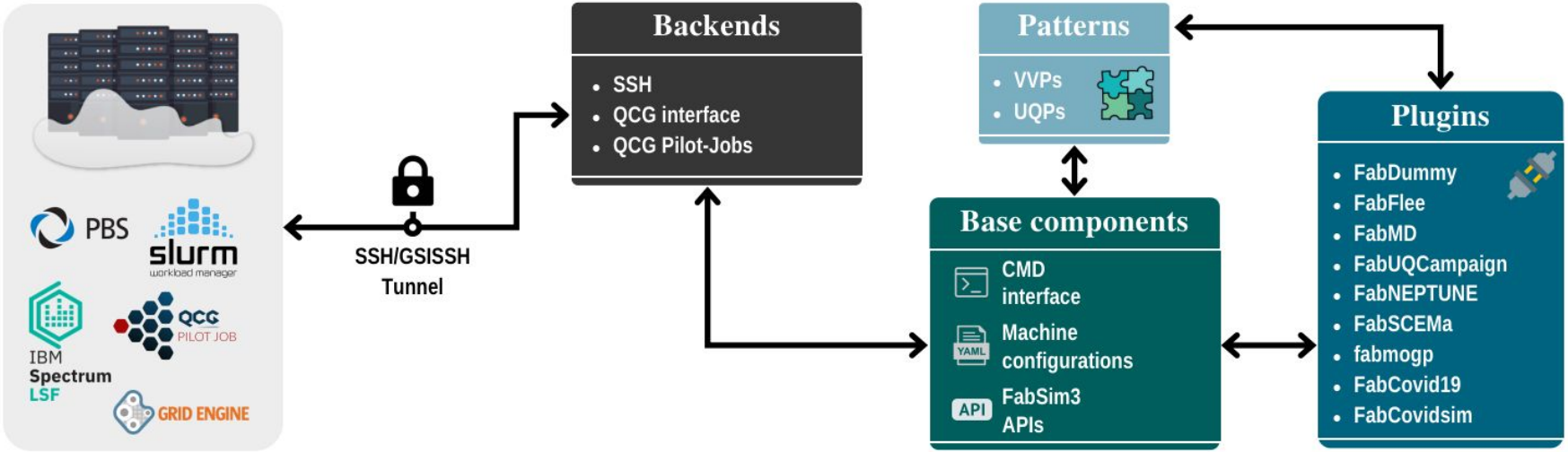
- Generic automation/curation toolkit for multiscale computing.
- Relies on Python3 + Fabric2.
- Supports range of backends:
  - Localhost.
  - Range of supercomputers.
  - QCG broker & Pilot Jobs.
- Focus on bash one-liners, e.g.:
- Supports domain-specific plugins

```
fabsim localhost run_amazingly_complex_app
fabsim eagle validate_flee:mali,cores=24,replicas=5
fabsim localhost install_plugin:FabUQCampaign
```

http://www.github.com/djgroen/FabSim3

Groen et al., *Comp. Phys. Comm. (2016) vol. 207, pp. 375--385.*
(new publication under review)

Design Overview



Key features

- Facilitate remote job management.
  - Provide shared definitions for machine-specific deployments.
- Curate local and remote environment
  - for consistent execution and easier debugging.
- Enable the use of pattern-based VVUQ (+SA)
- Provide shared definitions for application-specific deployments.

What can FabNeptune do?

- Simple **Neptune** simulation:
  - fabsim machine **Neptune**:**Neptune**_test*

- Ensemble **Neptune** simulation (input script with different topology):
  - fabsim machine **Neptune**_ensemble:**Neptune**_ensemble_example*

- **EasyVVUQ + FabNeptune** simulation:

  - Using **Dask**, Dynamic task scheduling optimized for parallel computing (https://docs.dask.org/en/stable/)
    - fabsim machine **Neptune**_init_run_analyse_campaign:fab**Neptune**_easyvvuq_InRuAn*_dask

  - Using **QCG-PilotJob**, Dynamic execution of many tasks inside a single allocation (https://qcg-pilotjob.readthedocs.io/en/latest/)
    - fabsim machine **Neptune**_init_run_analyse_campaign:fab**Neptune**_easyvvuq_InRuAn*_QCGPJ

  - Using **ThreadPoolExecutor**, Dynamic execution of each submitted task using one of possibly several pooled threads (https://docs.python.org/3/library/concurrent.futures.html)
    - fabsim machine **Neptune**_init_run_analyse_campaign:fab**Neptune**_easyvvuq_InRuAn*_ThreadPoolExecutor

EasyVVUQ + easysurrogate + FabNeptune simulation current activity

Machine (Localhost or remote)

EasyVVUQ+FabNEPTUNE

Convection 2D

Output values of aerodynamic forces during time-stepping

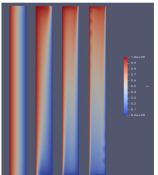
[Convection is triggered by maintaining a constant temperature difference across the space between the two largest faces]

Aero Forces (left hot, Boundary regions**)**

['F1-press_L', 'F1-visc_L', 'F1-total_L', 'F2-press_L', 'F2-visc_L', 'F2-total_L']

Aero Forces (right cold, Boundary regions**)**

['F1-pres_R', 'F1-visc_R', 'F1-total_R', 'F2-press_R', 'F2-visc_R', 'F2total_R']



Variance-based sensitivity analysis

Sobol method

selected model outputs

$$\begin{aligned} \frac{1}{Pr}\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) &= -\nabla p + Ra\, T\, \hat{\mathbf{y}} + \nabla^2\mathbf{u}, \\ \left(\frac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T\right) &= \nabla^2 T, \\ \nabla\cdot\mathbf{u} &= 0. \end{aligned}$$

System of equations describing vertical natural convection is, for fluid velocity u, temperature T and pressure p

Convection 2D

```
<?xml version="1.0" encoding="utf-8" ?>
<NEKTAR>
  <EXPANSIONS>
…

</GLOBALSYSSOLNINFO>
    <PARAMETERS>
      <P> TimeStep      = 0.01          </P>
      <P> T_Final       = 1.0           </P>
      <P> NumSteps      = T_Final/TimeStep   </P>
      <P> IO_infoSteps  = 10            </P>
      <P> Ra            = ${Rayleigh}E2        </P>
      <P> Pr            = ${Prandtl}           </P>
      <P> Kinvis        = Pr            </P>
    </PARAMETERS>
…
    <FUNCTION NAME="DiffusionCoefficient">
      <E VAR="T" VALUE="${DiffusionCoefficient}" />
…
</NEKTAR>
```

Selected model inputs

Sobol method

Rayleigh number:
(proportional to the applied temperature difference)

$$Ra = \frac{\beta g \Delta T L^3}{\kappa \nu}$$

Prandtl number:
(fixed by the choice of fluid in the tank)

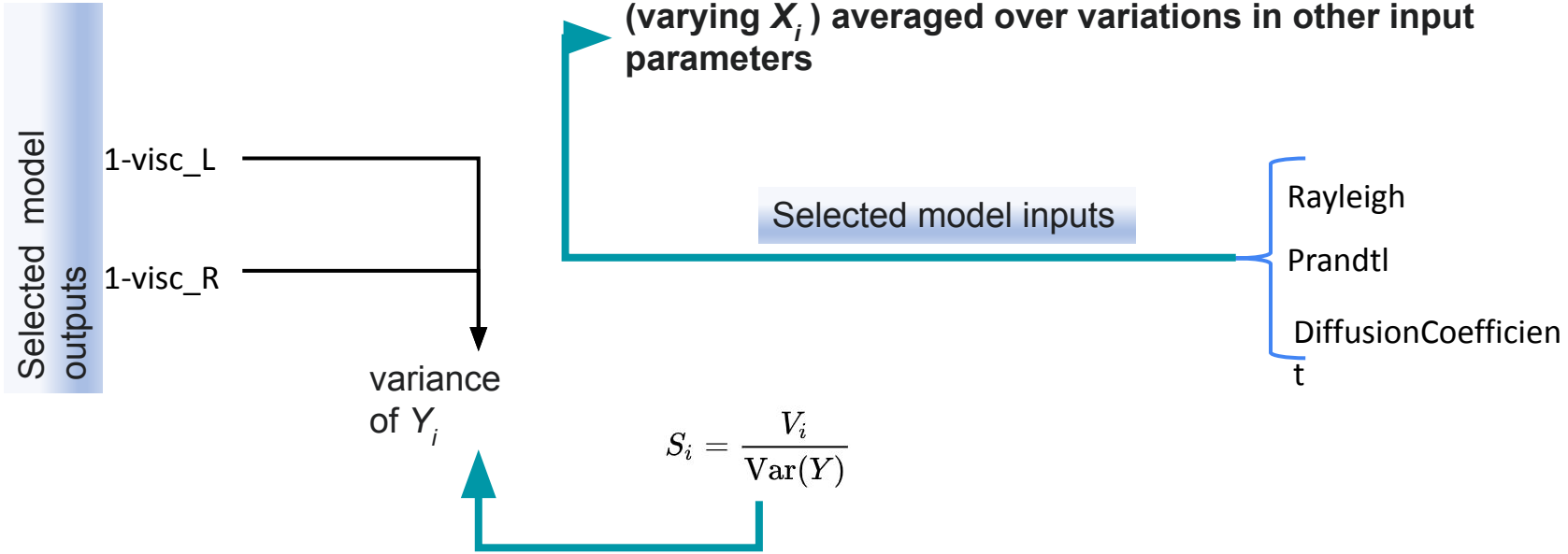
$$Pr = \frac{\nu}{\kappa}.$$

Nusselt number:
amount of heat transfer, mean of the integrals of the heat flux on the heated / cooled boundaries

$$Nu = \frac{\int_0^H -\nabla_x T(x_0, y) + \nabla_x T(x_1, y) dy}{\int_0^H -\nabla_x T(x_0, y) + \nabla_x T(x_1, y) dy|_{Ra=0}}$$

First-order indices

Sobol method



$$S_i = \frac{V_i}{\mathrm{Var}(Y)}$$

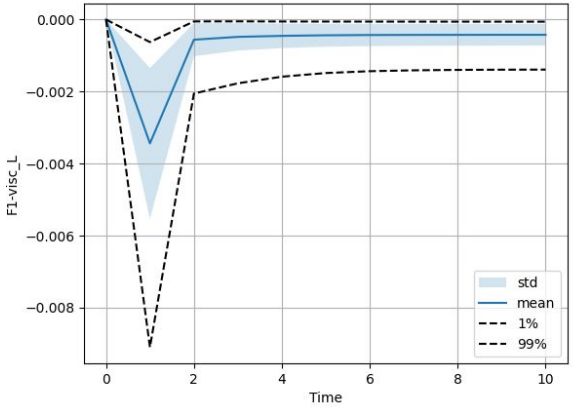
$$\sum_{i=1}^{d} S_i + \sum_{i<j}^{d} S_{ij} + \cdots + S_{12\ldots d} = 1$$

PCESampler, polynomial_order: 3

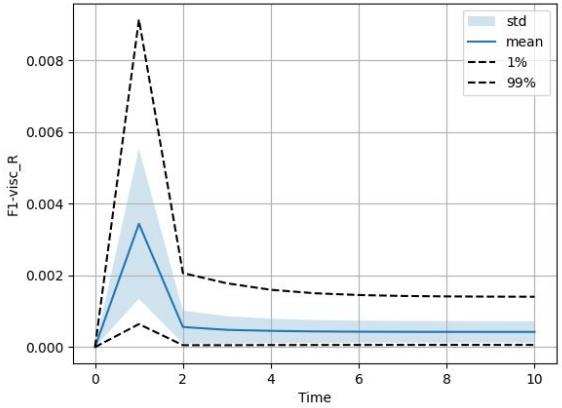
Moments

TimeStep= 0.01
T_Final = 1.0
NumSteps=
T_Final/TimeStep

F1-visc_L

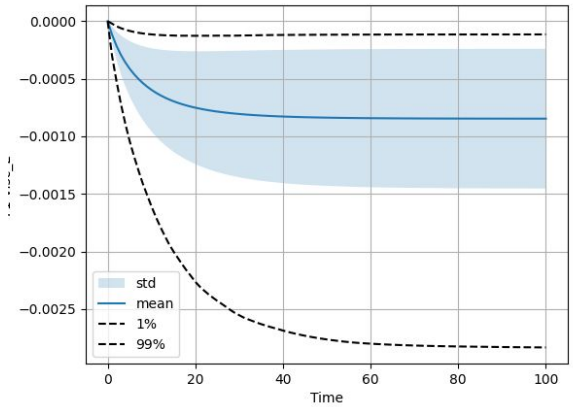


F1-visc_R

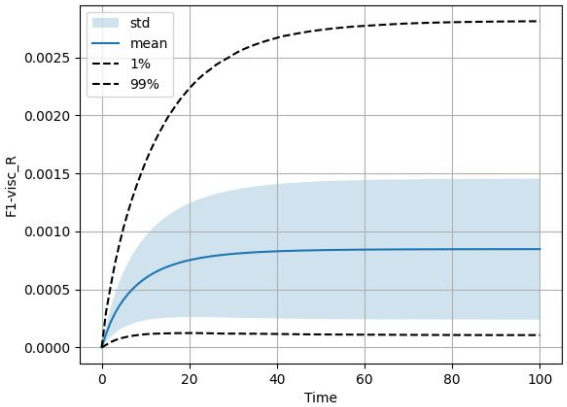


TimeStep= 0.001
T_Final = 1.0
NumSteps=
T_Final/TimeStep

F1-visc_L



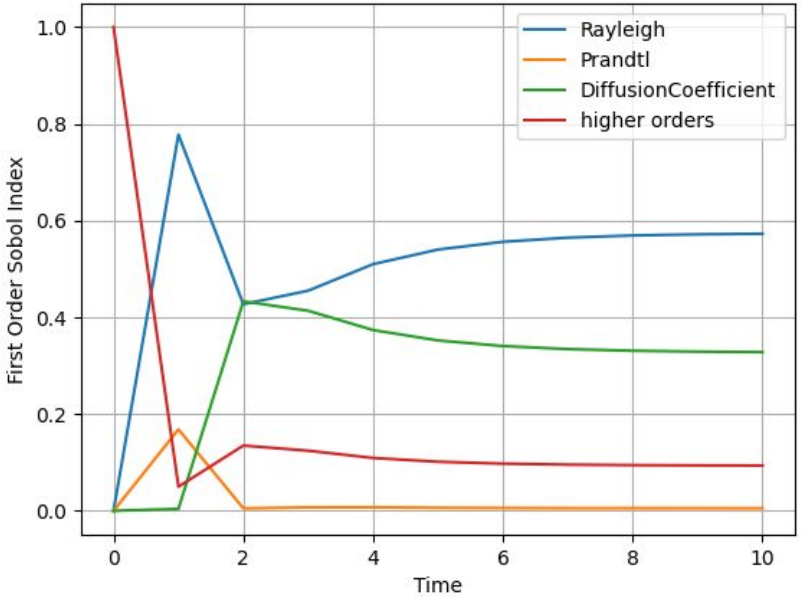
F1-visc_R



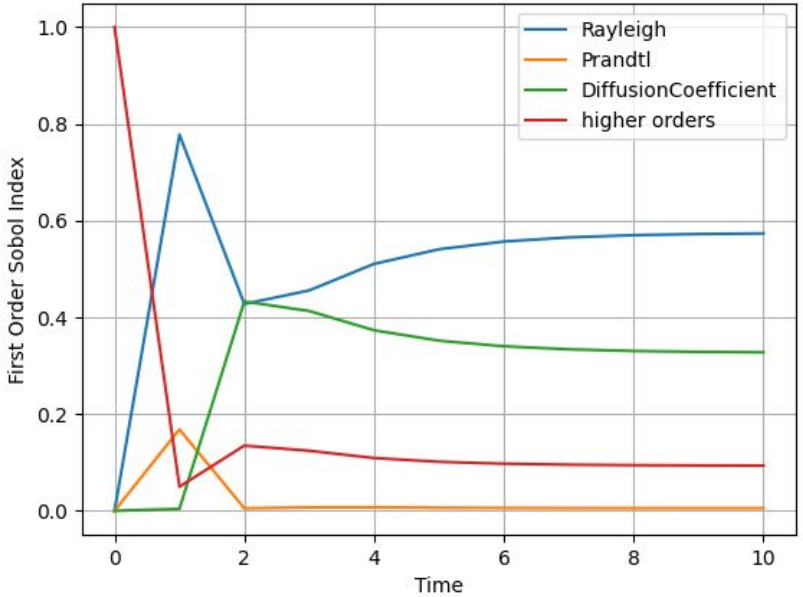
First-order indices

Sobol method

F1-visc_L



F1-visc_R

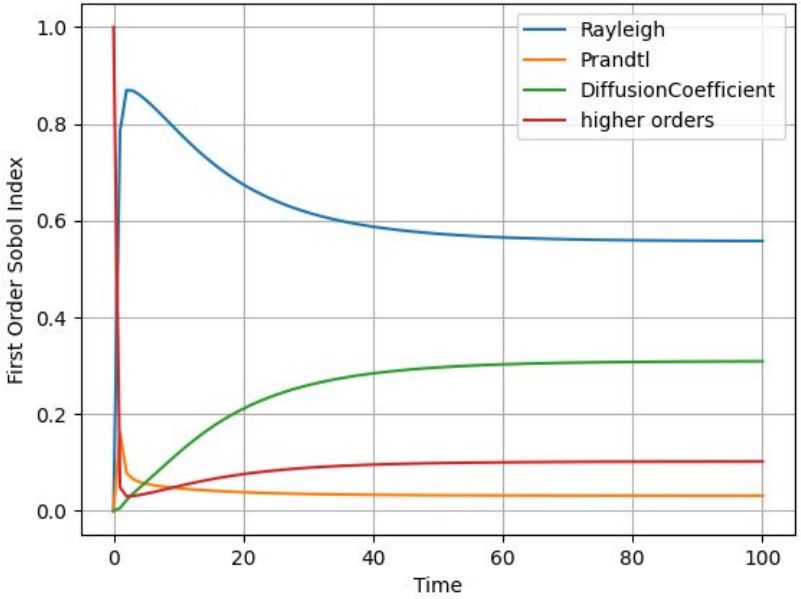


PCESampler, polynomial order: 3

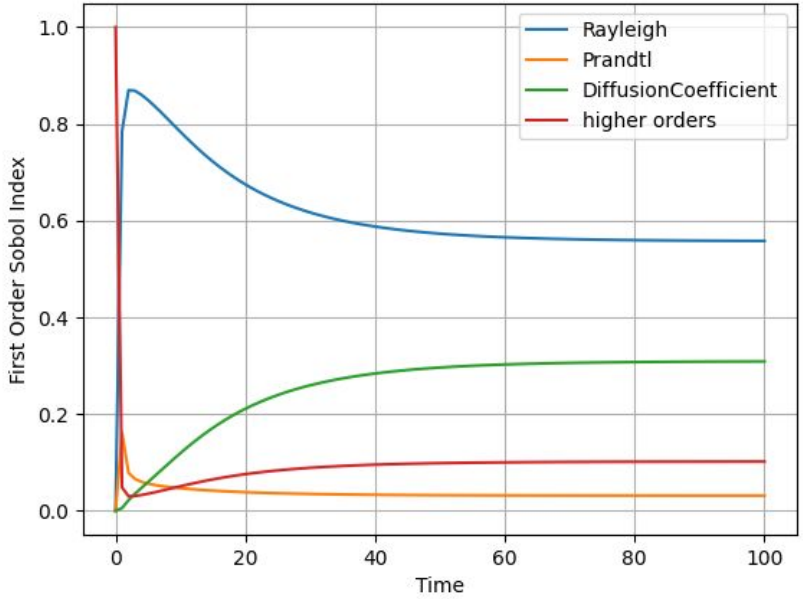
First-order indices

Sobol method

F1-visc_L



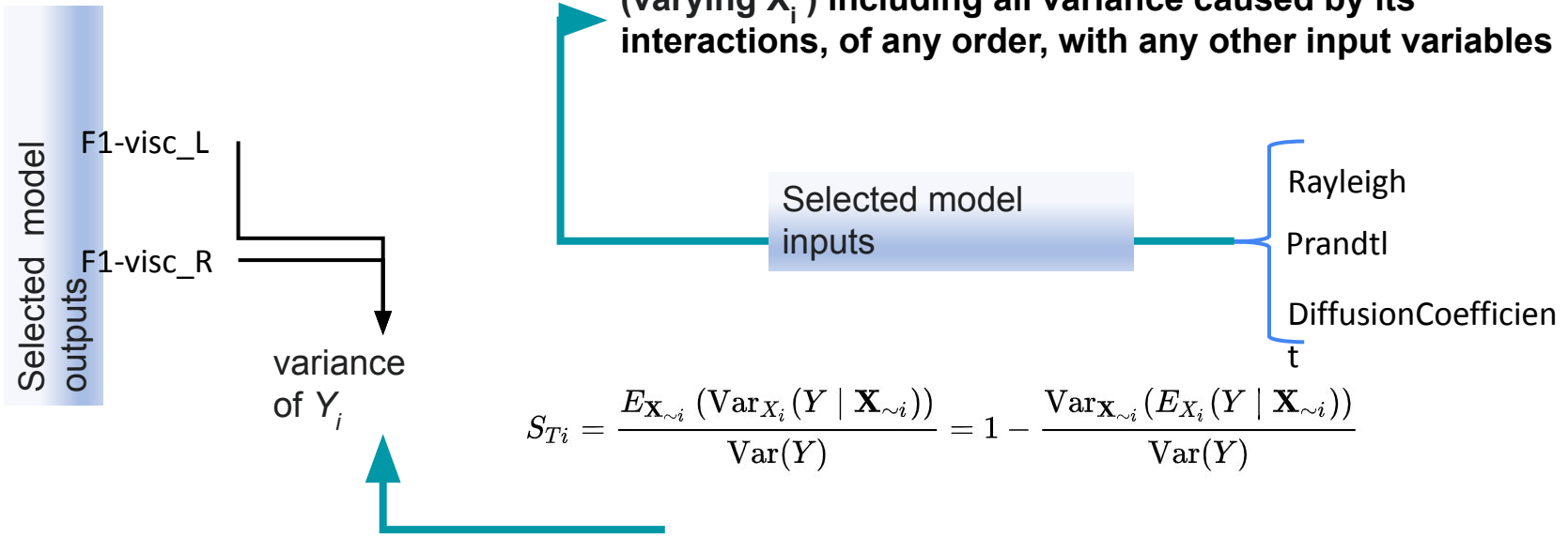
F1-visc_R



PCESampler, polynomial order: 3

Total indices

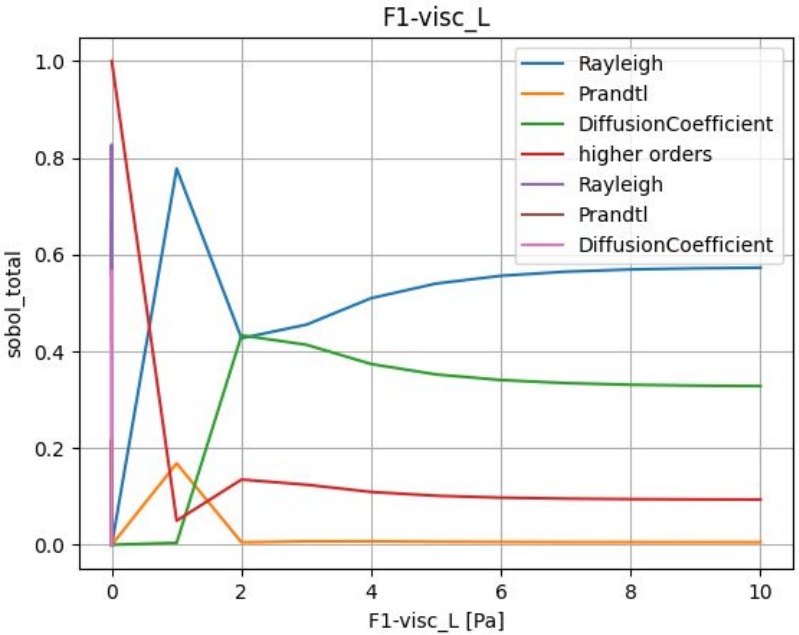
Sobol method

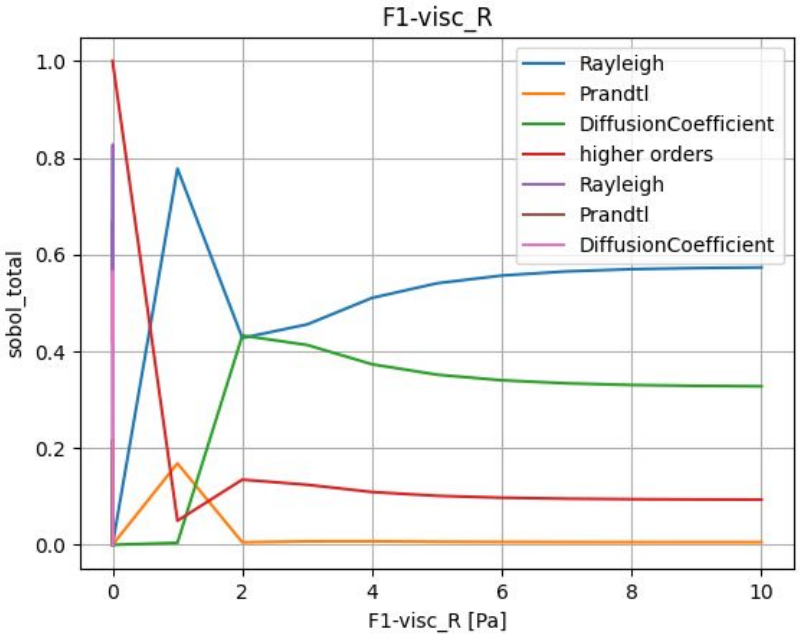


PCESampler, polynomial_order: 3

Total indices

Sobol method

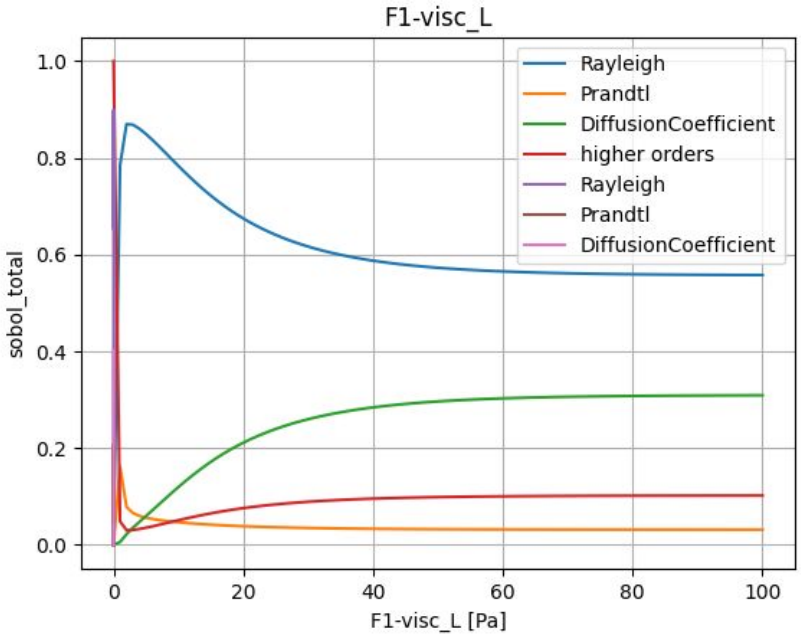


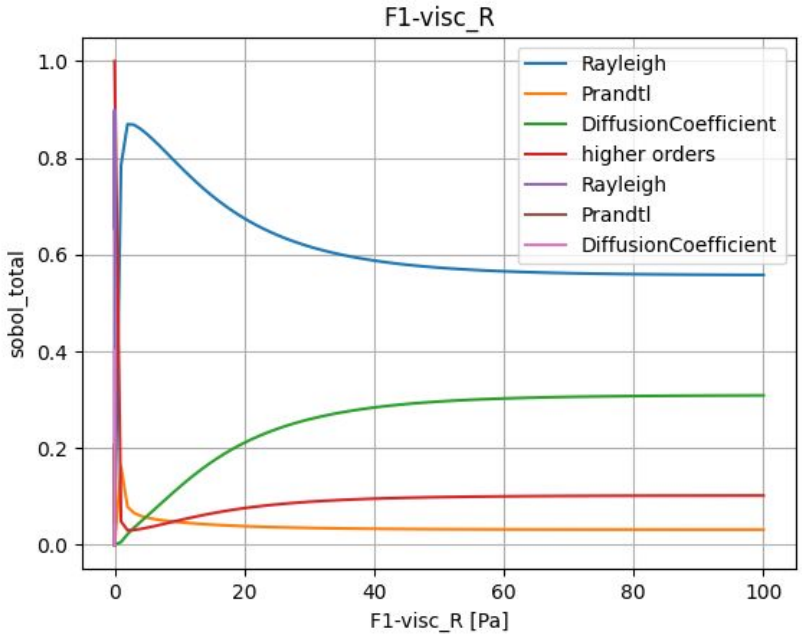


PCESampler, polynomial order: 3

Total indices

Sobol method





PCESampler, polynomial order: 3

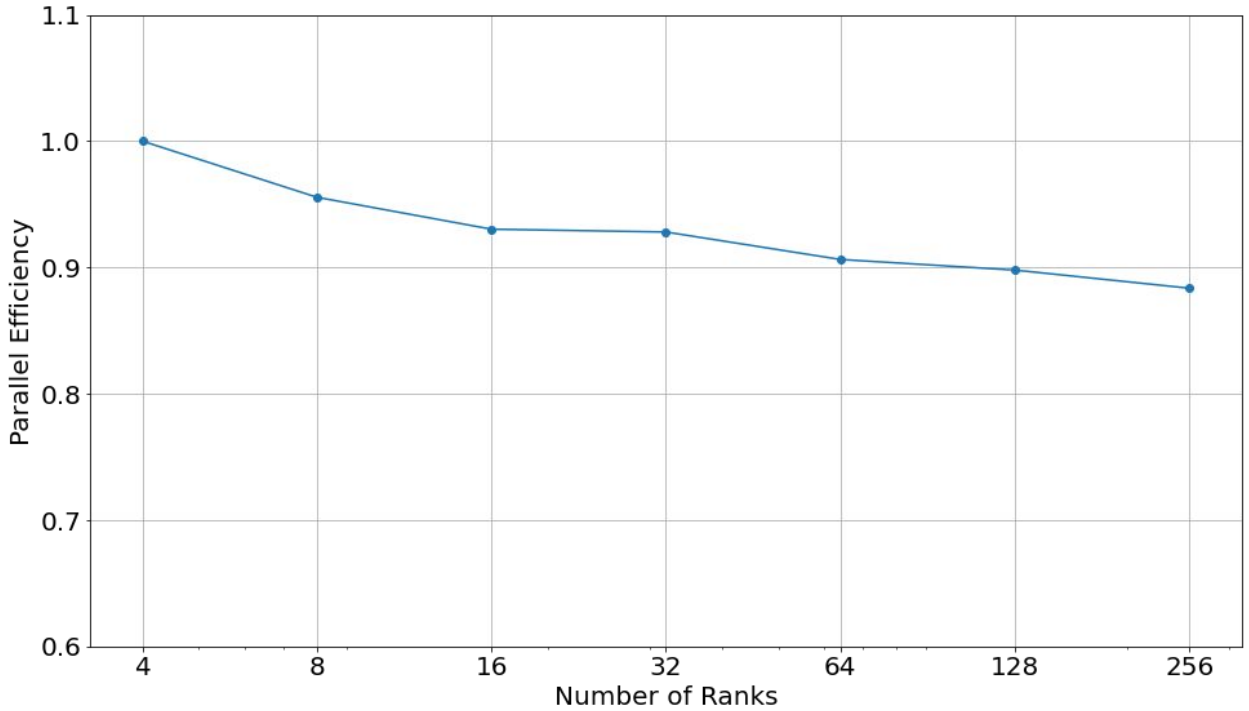
Data Assimilation: scaling

Initial steps towards using ParticleDA.jl for Nektar++

Collaboration with Leeds (velocity measurements) on observations

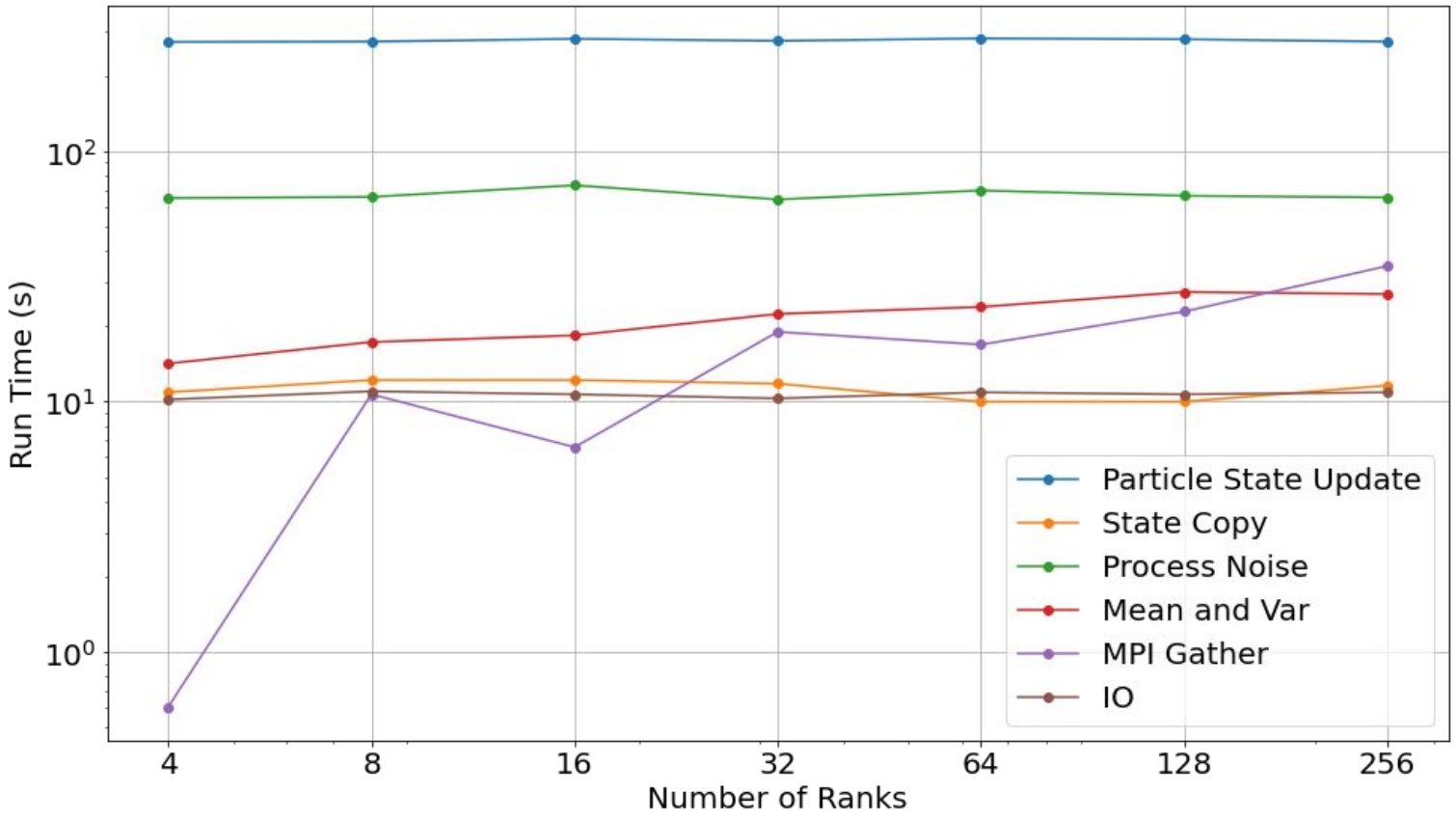
Satisfactory Weak Scaling on CSD3

- Tier2 CSD3 Peta4 KNL partition
- julia 1.4
- 16 threads/rank
- 64 particles/thread
- 4 ranks/node
- No hyperthreading
- 100x100x3 state array per particle



At 256 ranks, we are using 4096 cores and simulating 260 000 particles

Loss of Scaling Is Due to MPI Collectives in Computing Mean and Variance



On Archer2 the non-scaling variance calculation dominates the run time