D4.3: Report on the implementation of non-intrusive and intrusive VVUQ techniques

Due Date	14 June 2021
Delivery	8 June 2021
Submission of updated version	N/A
Lead Partner	MPG
Dissemination Level	Public
Status	Final
Approved	Executive Board
Version	V1.0

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 800925.

DOCUMENT INFO

Date and version number	Author	Comments
01.03.2021 v0.1	Onnie Luk	Skeleton
31-03-2021 v0.1	Wouter Edeling	First draft surrogate section
31.03.2021 v0.1	Yehor Yudin	Drafts on Gaussian Processes and metamodeling in Fusion
31.03.2021 v0.1	Dongwei Ye	Drafts on surrogate with convolutional neural network
08-04-2021 v0.2	Wouter Edeling	First draft UQP1 section
12.04.2021 v0.2	Onnie Luk	Update VVP4 fusion example
15.04.2021 v0.3	All	restructure draft and insert additional texts
30.04.2021 v0.4	All	Drafts on VECMAtk scaling and insert additional texts
11.05.2021 v0.5	All	Finalize draft to be submitted for internal review
28.05.2021 v0.6	Onnie Luk	Revise after receiving internal reviewers' feedbacks
08.06.2021 v1.0	Edeling & Luk	Final revision

CONTRIBUTORS

- Onnie Luk (MPG) – Main Author, Editor
- Yehor Yudin (MPG) - Author
- Jalal Lakhlili (MPG) - Author
- David Coster (MPG) - Author
- Wouter Edeling (CWI) - Author
- Federica Gugole (CWI) - Author
- Derek Groen (UBRU) - Author
- Diana Suleimenova (UBRU) - Author
- Hamid Arabnejad (UBRU) - Author
- Imran Mahmood (UBRU) - Author
- Maxime Vassaux (UCL) - Author
- Dongwei Ye (UvA) - Author, Reviewer
- Bartosz Bosak (PSNC) - Author
- Piotr Kopta (PSNC) - Author
- Tomasz Piontek (PSNC) - Author
- Daan Crommelin (CWI) – Reviewer
- Peter Coveney (UCL) – Approver, Editor

Disclaimer

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

1 Executive summary

In this deliverable, Work Package 4 (WP4) reports on the experience on implementing verification, validation, and uncertainty quantification (VVUQ) techniques from 9 multiscale applications (6 applications from within the VECMA portfolio and 3 newly-introduced external applications). Some of these experiences include fast-track implementation of non-intrusive uncertainty quantification (UQ) methods (Task 4.2) that were not yet available at the time when deliverable 4.2 (D4.2) was written. Deep-track implementation of more complex, semi-intrusive UQ techniques are widely explored in this deliverable (Task 4.3). Here applications apply the algorithms and methodology introduced in D2.2. Some of these techniques include optimization (cost-reduction) of UQ techniques: sampling efficiency by subsampling, replacement of expensive models with surrogate models (metamodeling), and sensitivity analysis (SA) based dimension reduction of multiscale models. It is worthwhile to mention that the metamodeling technique is classified as semi-intrusive at the multiscale level. However, it is an intrusive approach at the single-scale level, since the methods we utilize replace the single-scale component entirely with a surrogate. 5 of the 9 applications that we showcase in this deliverable have already introduced metamodeling into their workflows. Many of the metamodeling techniques can be found in the EasySurrogate library within the VECMA toolkit developed by WP3. SA based dimension reduction is demonstrated in this deliverable to help decrease the application uncertainty in multiple applications (Task 4.4). Verification and validation (V&V) techniques in the forms of V&V patterns (VVP) were introduced in D2.2, and they are also explored by several applications (Task 4.3). One of these VVPs also has the potential to help lessen the uncertainty in simulation models. We collaborate with WP3 to optimize the scalability of the VECMA toolkit (Task 3.4). Such effort is also documented in this deliverable in which we discuss the scalability of three of the components: EasyVVUQ, QCG-PJ, and FabSim3. The expansive uncertain parameter space in many multiscale applications requires petascale, and emerging exascale, resources. Working together with WP5, a theoretical estimate of the computational performance needs of the fusion application on exascale resources is provided.

2 Implementation of complex VVUQ techniques

Various verification, validation and uncertainty quantification (VVUQ) techniques are developed, both in terms of algorithms (see D2.1 and D2.2) and software readiness. They are tested and implemented in a wide range of application examples. Previously in D4.2, we showcased the fast-track implementation of the VVUQ techniques in the VECMA applications and 1 external application. These implementations were non-intrusive to the workflow, and the UQ methods used include Stochastic Collocation, Polynomial Chaos Expansion, quasi-Monte Carlo, parameter sweep and ensemble bootstrapping. Here, we report on our experience in the deep-track implementation of these UQ methods, not only from the applications in the VECMA portfolio (Multiscale Migration Prediction, Climate, Fusion, In-Stent Restenosis 3D, Materials and Binding Affinity Calculator), but also from three new external applications: CovidSim, Flu and Coronavirus Simulator (FACS) and Dutch-Covid Model. The deep-track implementations utilized the VECMA UQ and V&V patterns in either a semi-intrusive acyclic (UQP2) or a semi-intrusive cyclic (UQP3) workflow, with optimization technique options such as sampling efficiency (UQP-A) and metamodel (UQP-B). In addition, V&V techniques such as Stable Intermediate Forms, Level of Refinement, Ensemble Output Validation, and Quantities of Interest Distribution Comparison are also formulated into V&V patterns, e.g. VVP1 - VVP4.

In addition, we explore the performance of the VECMA toolkit in several application examples. We studied the scaling of EasyVVUQ, QCG-PJ, and FabSim3 in the VECMA toolkit [1-3]. Depending on the number of uncertain parameters in the application, the UQ study can easily require petascale and emerging exascale computing, which is demonstrated in the BAC and Fusion examples. An estimate of the computational performance needed from a theoretical exascale machine based on the characteristics of Altair cluster at PSNC is presented in Section 2.4 with help from WP5.

Two VECMA applications we do not include in this deliverable are the Continuum Blood Flow application (HemeLB) and In-Stent Restenosis 2D (ISR2D). It is important to emphasize that ISR2D is a fast-track application (see D4.2). We took what we learned from the ISR2D UQ implementation to help advance the UQ study of the In-Stent Restenosis 3D (ISR3D) model (a deep-track application).

2.1 Complex UQ techniques in multiscale workflow

Deep-track implementations of more complex, non-intrusive and semi-intrusive (at the multiscale level) UQ techniques are described in detail in this section. The non-intrusive techniques include dimension-adaptive methods to tackle the high-dimensional input parameter space. Some of these semi-intrusive techniques include optimization (cost-reducing) UQ techniques: UQP-A: sampling efficiency by subsampling, UQP-B: replacement of expensive models with surrogate models

(metamodeling), and sensitivity analysis (SA) based dimension reduction of multiscale models. The metamodeling technique we use is semi-intrusive at the multiscale level. However, when we consider the level of intrusiveness towards the single-scale model, it is indeed an intrusive approach.

2.1.1 Non-intrusive uncertainty propagation (UQP1)

We define UQP1 as standard parametric uncertainty quantification (UQ), where the entire (multiscale) code is considered as a black box, see Figure 1 below for a schematic. Here, an input distribution is *assumed* (e.g. based on expert knowledge) which is then propagated forward through the model in order to investigate the corresponding distribution of the output. The variance of the output distribution is a measure of the predictive uncertainty caused by a lack of knowledge in the most suitable values for the inputs. Our main tool for this is EasyVVUQ. Both EasyVVUQ and UQP1 have been extensively described in earlier deliverables. Hence, here we will only focus on recent additions to, and applications of EasyVVUQ, which revolved around the idea of dimension reduction.

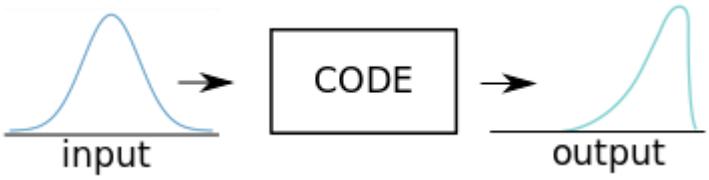


Figure 1: Non-intrusive uncertainty propagation.

Sampling high-dimensional input spaces with EasyVVUQ

Until recently, the methods for UQP1 available in EasyVVUQ were Monte Carlo, Quasi Monte Carlo, Polynomial Chaos Expansions (PCE) and Stochastic Collocation (SC). The first two can be applied to high-dimensional input spaces, but suffer from a low convergence rate, whereas the PCE and SC methods suffer from the curse of dimensionality. This is caused by the fact that their sampling plans are based on simple tensor products of 1-dimensional quadrature-rule nodes, which results in an exponential increase in the required number of code evaluations with an increasing number of uncertain inputs. Practically speaking, this limits their application to less than around 20 uncertain inputs.

To make EasyVVUQ scale to higher-dimensional input spaces, we implemented a dimension-adaptive version of the SC sampler, which banks on the existence of a relatively low *effective dimension.* It is reasonable to assume that the predictions are not equally sensitive to every input, and that therefore a smaller subset of them explain most of the output variance. The dimension-adaptive SC sampler starts with a single sample, and iteratively refines the sampling plan in the directions of the stochastic input space which are found to be more important. Whichever directions are more important than

others is based on the computation of a suitable error metric. We have implemented error metrics based interpolation and those based on quadrature. The former looks at a normed difference between the SC polynomial surrogate (see the metamodelling section below) and code samples in new directions. The latter metric compares the change in the output variance in different directions, and places new samples in the direction with the largest change.

While this approach is still based on tensor products, it now writes the total SC expansion as a *combination* of different tensor products. This makes the sampling plan anisotropic, which allows us to ignore those directions of the input space which do not significantly contribute to the output. This does not break the curse of dimensionality, but rather postpones it to higher dimensions.

2.1.2 Sampling Efficiency (UQP-A)

UQP2 and UQP3, as discussed in D2.1 and D2.2, are based on semi-intrusive methods and apply to acyclic and cyclic multiscale models, respectively. One way to optimize the computational efficiency is by efficient sampling (A). Efficient sampling refers to sampling techniques more efficient than basic methods and hence reduces the number of samples required to perform UQ for the given set of input parameters.

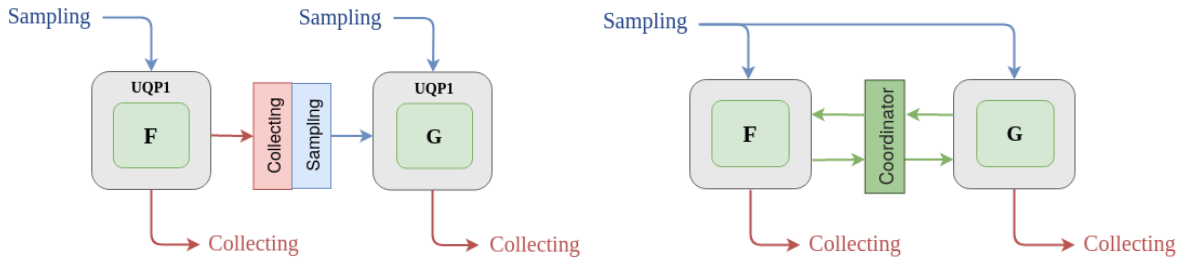


Figure 2: Semi-intrusive acyclic pattern (UQP2, left) and semi-intrusive cyclic pattern (UQP3, right). This figure is taken from [4].

There are two important advantages of applying UQP2 (Figure 2, left): transparency and efficiency. UQP2 makes it possible to investigate how uncertainty propagates and amplifies within each component of the model. UQP2 can be realized as a sequential application of UQP1. After applying UQP1 to submodel F, the data to be sent to submodel G has now turned into an uncertain output, which is then converted into uncertain input for submodel G. Therefore, UQP2 makes uncertainty propagation more transparent and provides additional information on uncertainty as it propagates between submodel levels.

The second important advantage of UQP2 is that it introduces different ways of improving the computational efficiency of corresponding UQ analysis. One way to obtain better efficiency is to apply

resampling: samples of the output of one submodel can be used to approximate the probability density function (PDF) of this output. Then, this output can be considered along with other uncertain inputs of the next submodels and, therefore, uncertainty propagation of this submodel can be performed independently of the analysis of the previous one. This allows application of the most efficient uncertainty propagation methods for each of the single-scale models. It also permits a more flexible implementation of UQP2-A, in which sampling for each submodel can apply a different advanced sampling method. Another way to improve the efficiency for acyclic multiscale models is to build a metamodel of the most expensive single-scale model (see next section).

For cyclic multiscale models, UQP3 again performs non-intrusive UQ on individual single components, as shown on the right side in Figure 2. Compared to the UQP2, in UQP3 we add the Coordinator module between the submodels, to orchestrate the data flows. For UQP3, resampling cannot be applied as it can be for acyclic UQP2 workflows, because the models alternate to modify the model's state, which causes their states and parameters to become correlated. The output of the first model must be sampled conditionally on the value of the shared parameters for the target second model instance. Therefore, instead of resampling, the semi-intrusive Monte Carlo (SIMC) method can be applied to improve the efficiency (UQP3-A). Using the Coordinator between the submodels, one can deploy the corresponding interpolation and the statistical validation. For cyclic multiscale models, the states of the submodels become correlated even if their parameters are separated, through the repeated communication between the submodels. UQP3-A can be applied in this situation as well. Additionally, it is important to note that the implementation of UQP3-A is not limited to Monte Carlo methods. Other UQ methods, such as PCE or SC, can be applied to reduce its computational cost and improve the efficiency of the multiscale UQ analysis. [4]

2.1.3 Metamodeling (UQP-B)

Metamodelling, also denoted as surrogate modelling, is a class of techniques meant to replace an expensive model by a cheaper surrogate, from which many evaluations can easily be drawn at low computational cost. They are trained on supervised data, i.e. input-output samples from the original model. A very large variety of techniques exist, which include Gaussian Processes, polynomial-based surrogates, and machine-learning techniques. Of the polynomial techniques well-known examples include PCE and SC methods. These have already been implemented in EasyVVUQ, our toolkit for UQP1-type problems. For the conceptual basis of the different UQPs, we refer again to VECMA deliverable D2.1. While the primary purpose of UQP1 is to propagate an assumed input distribution through the computational model, in the process a polynomial approximation of the input-output relation is created. This approximation can be used to interpolate the code output to unsampled points

in the input space. Under certain regularity assumptions on the code output, and with a limited number of input parameters, these types of surrogates can be very accurate, and they are orders of magnitude faster than numerical models they approximate. This opens up the possibility of using them in, for instance, inverse problems where the assumed input distributions act as the prior distributions. The posterior distributions of the inputs, conditioned on some observed data, can be obtained with Markov Chain Monte Carlo (MCMC) methods, where the expensive model in the likelihood function is replaced by a polynomial surrogate. Recently, MCMC methods have also been implemented in EasyVVUQ, which opens up the possibility of using inverse problem approaches to expensive simulations, provided we are able to create a suitable surrogate with Stochastic Collocation or Polynomial Chaos.

The preceding all assumes we can view the simulation as one big black box. Multiscale simulations however, consist of coupled models operating at different spatial and temporal scales. A more advanced use of surrogate methods is to replace a sub-model, and keep the coupling with other `real' models intact. In practise, the lion's share of the computational burden is spent at computing the small, microscopic scales of the simulation, which is why we focus on replacing these by a surrogate.

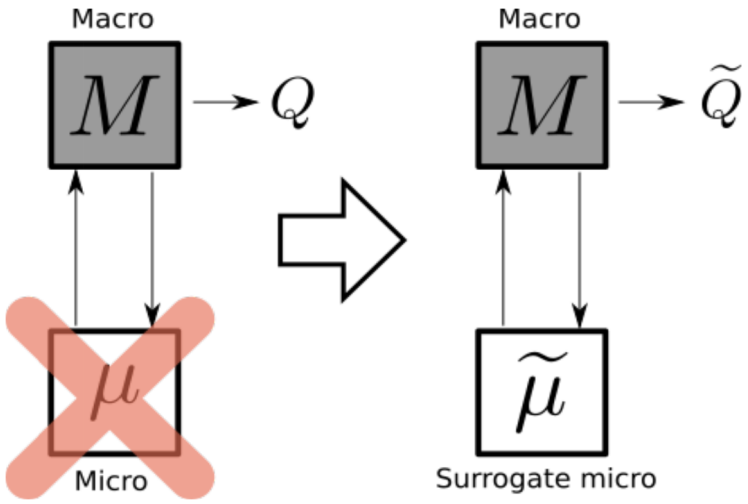


Figure 3: Replacing a micro model with a surrogate in a coupled, cyclic multiscale model.

These types of surrogate methods are clearly different from those implemented in EasyVVUQ, which deal with forward uncertainty propagation. For this reason we added a new component to the VECMA toolkit, namely EasySurrogate. It is designed to provide a library of surrogate methods which can be applied to situations as depicted in Figure 3, with a two-way coupling between the surrogate and the macroscopic solver. For the sake of consistency, we have designed this toolkit with similar syntax to that used in EasyVVUQ. Thus far we have included surrogate methods in the following categories:

1. **Stochastic surrogates.** When the response of micro model μ that feeds back to macroscopic model M is not uniquely determined by the instantaneous input from M into μ (e.g. because of internal chaotic behaviour of μ), the uncertainty of the response can be accounted for by a stochastic representation, i.e. a stochastic surrogate. Further descriptions can be found in D2.2. Some example results are displayed below. At the time of writing, EasySurrogate contains two such methods, namely quantized softmax networks and kernel mixture networks.
2. **Standard regression surrogate.** We have also implemented a standard feed-forward neural network, mainly for comparison purposes. Also, these types of networks can also be used for UQP1 type problems, see the ongoing work described below.

Ongoing work:

1. **Reduced surrogates.** The problem of learning a surrogate for μ is that the state of the multi-scale model can be very large. This necessarily leads to large neural networks, and the need for large training data sets. However, often the quantities of interest are some integrated functions of the full multiscale state. We are developing a new class of dimension reduction techniques that take advantage of such a massive size difference between the multiscale state and the target quantities of interest. This method compresses the training data by several orders of magnitude, which results in smaller machine learning surrogates. The description of this type of surrogate was first introduced in D2.2. The idea is that these can be more easily applied to large-scale problems. Thus far we have demonstrated this approach on the 2D Navier Stokes equations, and the 2D Gray-Scott reaction diffusion equations, see the examples below. This method is also implemented in EasySurrogate, although thus far it is geared towards spectral solvers only.
2. **Online learning**. We previously mentioned the difficulty of two-way coupling between M and μ, and that error accumulation can lead to biased output statistics. The problem is that training a machine-learning surrogate for μ on data alone is not sufficient. We need a surrogate that performs well in the two-way coupled *modelling environment*, not a surrogate that can reproduce *data* well. Some recent work has focused on this topic in the context of simple climate models. We have reproduced these results in EasySurrogate, and are now in the process of applying this method in combination with the reduced surrogates mentioned above.
3. **Machine-learning based UQP1.** Neural networks can also be trained on supervised training data composed of input-output samples, which makes it a UQP1-type of surrogate. We are

currently developing an interface between EasyVVUQ and EasySurrogate such that an EasyVVUQ dataframe can be read by EasySurrogate, after which a standard neural network is trained to represent the input-output relation. In addition to standard neural networks, we also have a working initial implementation of so-called Deep Active Subspace surrogates. These are similar to vanilla neural networks, except that the initial layer implements a dimension reduction where only a small number of important linear combinations of input variables are passed through the rest of the network. An example is given below, using the CovidSim model as a test case. Finally, since neural networks offer rapid access to gradients using back propagation, we are also investigating computing global, derivative-based SA estimates.

4. **Convolutional neural network**. Apart from the traditional node-based neural network, convolutional neural networks (CNN) can be used to construct the surrogate model for computational models with the inputs and outputs of which can be considered as imaged-like information. Detailed descriptions of CNN and how it is applied as a surrogate model in the UQP3-B type of UQ analysis, within the ISR2D model, were reported in D2.2.
5. **Gaussian Processes.** Another way to build a surrogate for the micromodel μ is to represent it as a Gaussian Process (GP). This surrogate would treat a likelihood of a quantity of interest as a multivariate normal distribution fully defined by a vector of means and a covariance matrix. Once these parameters are defined through a Gaussian Process Regression (GPR) and subsequent maximum likelihood estimation, the model requires only cheap computations to produce gaussian posterior distribution of the quantity of interest of simulation for any new feature data. Such a GP emulator can interpolate the value of response of the simulator and provide an uncertainty measure in form of a variance, and gives a fully probabilistic surrogate for μ. One of the examples of Gaussian Process utilization at VECMA is the fusion application where GP-based surrogates are used to emulate turbulence codes for the purpose of acyclic UQP2-type analysis of transport fluxes in tokamak plasma. This is discussed further below.

2.2 Verification and validation techniques

Verification and validation (V&V) are essential tasks to perform when building robust and relevant simulations. Verification assesses the quality of the numerical treatment of the model used in the predication, while validation assesses the correspondence of the model results to real-world observations (or ground truth benchmarks).

V&V patterns, or *VVPs*, are intended to encapsulate essential V&V procedural elements in an application-agnostic way. Many such patterns can be conceivably defined, but as a starting point within

VECMA we focus on four VVPs that have a widespread applicability, both in a single scale and a multiscale context. These patterns were introduced and described in D2.2, and they are: VVP1-Stable Intermediate Forms (SIF), VVP2-Level of Refinement (LoR), VVP3-Ensemble Output Validation (EoV), and VVP4-Quantities of Interest Distribution Comparison (QDC). QDC was previously referred to as the Quantity of Interest pattern in D2.2. Two additional forms of similarity measures have been added into QDC since the dissemination of D2.2: Z-test, or the compatibility measure. Currently, VVP2 is available within FabSim3, while VVP4 is available within both FabSim3 and EasyVVUQ.

2.3 Application Examples

The tables below summarize the VVUQ techniques (Table 1) along with surrogate modelling techniques (bottom) utilized by the example applications. In the rest of the section, we report on each application's VVUQ implementation experience. While the tables show certain techniques are applied in only one or very few applications, all of these methods are meant to be generic enough for any application to use.

Table 1: VVUQ (exclude metamodeling) techniques employed by individual applications.

	SC	PCE	qMC	MC	SIF	LoR	EoV	QDC
CovidSim	**x**							
Migration	**x**	**x**				**x**	**x**	
FACS	**x**	**x**			**x**^	**x**	**x**	
Dutch-Covid				**x**				
Climate	**x**							
Fusion		**x**	**x**					**x**
ISR3D			**x**					
Materials	**x**					**x**		
BAC /NAMD	**x**							

^ FACS intends to apply the SIF technique in the near future

Table 2: Metamodeling techniques employed by individual applications.

	Stochastic surrogate	Standard regression surrogate	Reduced surrogate	ML-based UQP1	CNN	GP
CovidSim				**x**		
Climate	**x**	**x**	**x**			
Fusion						**x**
ISR3D*				**x**	**(x)**	**x**
Materials						**x**

* ISR2D (not ISR3D) uses CNN metamodeling method

2.3.1 CovidSim

CovidSim is a well-known and influential epidemiological code, that has been used to advise the UK government on the effects of various non-pharmaceutical interventions, e.g. social distancing and school closures. We analysed the effect of parametric uncertainty on the predictions of this code, and found it to contain substantial uncertainty. Due to the large number of input parameters, we implemented a dimension-adaptive sampler in EasyVVUQ, enhancing the scalability of the toolkit to high-dimensional input spaces. This work was published in *Nature Computational Science*, and has received considerable attention [5]. Please refer to Appendix 4.1: CovidSim for a more detailed description of this work.

A more recent addition to the VECMA toolkit is EasySurrogate. Here, we have implemented a neural-network based UQP1 algorithm, capable of handling high-dimensional input spaces. It too was applied to the CovidSim model in ongoing work, see again Appendix 4.1: CovidSim.

2.3.2 FACS : Flu And Coronavirus Simulator

FACS [6] is an agent-based modelling code that models the viral spread at the sub-national level, incorporating geospatial data sources to extract buildings and residential areas within a predefined region. Using FACS, we can model Covid-19 spread on the local level, and provide estimations of the spread of infections and hospital arrivals, given a range of public health interventions. The full simulations considered several scenarios – including that an area takes no measures, that an area enters an extended lockdown period or that an area runs a 'dynamic' lockdown using a variety of changing interventions – with all of the lockdown scenarios showing a marked decrease in local

infections. The simulation allows infection rates to be visualised on a map, run using OpenStreetMaps, letting modellers see where hotspots develop, and how hotspots move as the disease spreads through a population.

2.3.3 Dutch-Covid Model

We analysed the Virsim model [33], an epidemiological model for the spread of Covid-19 in the Netherlands. Non-intrusive uncertainty quantification analysis (UQP1) is presented for the robustness analysis of the impact of four COVID-19 exit strategies under uncertainties in the input parameters and due to the intrinsic stochastic nature of the computational model. The analysis revealed large uncertainties in the model outputs and high probabilities of an overburden on the healthcare system. Furthermore, for each strategy it highlighted the model parameters responsible for large portions of the output variance. These pieces of information can be used either to refine the computational model or to inform policy-makers such that targeted interventions can be made.

Please refer to Appendix 4.2: Dutch-Covid Model for a detailed description of this work.

2.3.4 Multiscale Migration Prediction

Predicting forced migration is crucial as governments and NGOs can save refugees' lives and allocate humanitarian resources efficiently [7]. FLEE [8] is an agent-based simulation approach where a complex system is modelled as a set of autonomous decision-making agents that behave accordingly with their environment based on a set of rules. FLEE forecasts the destinations of people escaping violent conflicts,and has been previously validated for a range of geographically different conflicts [9]. FLEE requires a range of input parameters, which specify the environment and population of agents along with their properties. The Flee code calculates the movement of displaced agents on a daily basis with a total number of simulation days. Flee is optimised for simplicity and flexibility, and supports simulations with 100,000s of agents on a single desktop. The parallel version of FLEE (called pFLEE) is a highly efficient and scalable code that helps easily simulate complex simulation scenarios of forced migration occurring at different parts of the world.

We investigated the sensitivity of input parameters in the FLEE algorithm using the SC sampler and identified the parameters that affect the simulation output. We performed SA for four African conflict situations. We also modified the FLEE algorithm (FLEE 1.9) and changed the agent-based modelling migration ruleset (FLEE 2.0), which improved the average total error across conflict situations. Moreover, we demonstrate the application of UQP-A, VVP2 and VVP3 with FLEE.

Please refer to Appendix 4.3: Multiscale Migration Prediction for a detailed description of this work.

2.3.5 Climate

The climate examples are three-fold:

1. We studied the parametric uncertainty of the (compute-intensive) Dutch Atmospheric Large Eddy Simulation. The novelty of this work was the inclusion of discrete inputs on various modelling choices, as well as the inclusion of the random seeds.
2. We developed stochastic neural-network based surrogate models in EasySurrogate. We expect that the stochastic nature results in improved surrogate models for the unresolved scales of (geophysical) flow problems. Extensive tests on the Lorenz96 system yielded positive results.
3. We developed so-called reduced-surrogate models to represent the effect of the unresolved scales on key QoIs. This is done with the scalability of EasySurrogate in mind. Unlike the 'traditional' sense of scalability, where the method is scaled to large HPC machines, here develop a new form of dimension reduction, thus downscaling the problem instead of upscaling the method.

Please refer to Appendix 4.4: Climate for a detailed description of this work.

2.3.6 Fusion

The multiscale fusion workflow (MFW) is constructed to model the plasma dynamics inside fusion devices that involve highly disparate scales in both space and time. MFW couples 3 single-scale models (transport, equilibrium, and turbulence) together, along with a module that converts fluxes to transport coefficients required by the transport model. Further details on the fusion application and its fast-track implementation of the UQ techniques (UQP1) can be found in D4.2. Three patterns for UQ and V&V are presented for the multiscale fusion simulations: semi-intrusive acyclic UQ based on sampling efficiency improvement (UQP2-A) and surrogate modeling (UQP2-B), as well as validation of quantities of interest (VVP4). For the first case we coupled single-scale deterministic models, namely a transport solver to evolve plasma core profiles, e.g. temperatures, and an equilibrium model to update the plasma geometry, to create an acyclic workflow. By utilizing the cubic B-Spline representation of temperature profile obtained from the transport model, which also serves as inputs to the equilibrium model, we reduced the correlation and dimensionality for the equilibrium model and improved the UQ computational efficiency by almost a factor of 2. Secondly, we applied GP

regression as a surrogate for the plasma turbulence microscale model to analyse the resulting particle and heat fluxes. Then we used this surrogate to implement a sequential optimization scheme that can gradually reduce the uncertainty of the surrogate. For validation, we applied custom skewness-informed compatibility metrics to compare the plasma temperature profiles obtained at a quasi-steady state, from MFW simulations and experimental measurements. Currently, we are developing a rubric to interpret the compatibility results, which would improve our confidence in the simulation results. Please refer to Appendix 4.5: Fusion for a detailed description for this work.

2.3.7 In-stent restenosis 3D (ISR3D)

Restenosis can happen in coronary arteries after stent deployment due to excessive growth of new tissue in the vessel's lumen (neointimal proliferation). In-Stent Restenosis 3D (ISR3D) is a multiscale computational model to simulate this process. It consists of three submodels: an initial deployment model, an agent-based smooth muscle cell (SMC) model and a blood flow model using the Lattice Boltzmann method. To study the process of restenosis, forward UQ is applied to investigate the influence of uncertain biological parameters on the neointima growth (UQP1). The four epistemic uncertainties considered in the forward propagation include endothelium regeneration time, balloon extension area, the threshold strain for smooth muscle cells bond breaking, and the percentage of fenestration in the internal elastic lamina. However due to the expensive computational cost of ISR3D (around 800 core-hour per sample), we adapted UQP1-B and applied surrogate modeling to conduct uncertainty quantification. We applied the Gaussian process regression method to develop a non-intrusive surrogate model for the ISR3D model. To further improve the computational efficiency, an active learning method is used to select the training points effectively and efficiently. The method is based on the Active Learning Cohn [10] but extends the point selection with laplacian regularisation. We are currently working on a combination of semi-intrusive surrogate modeling (UQP3-B) with a multi-fidelity algorithm (multi-fidelity Monte Carlo) in order to minimise the error introduced by the surrogate modelling. Please refer to Appendix 4.6: ISR3D for a detailed description for this work.

2.3.8 Materials

We employ SCEMa (Simulation Coupling Environment for Materials) [11] to simulate a standard dogbone test of an epoxy resin. The dogbone test, which is a standard way to test the properties of a sample dogbone-shaped material under tension, is simulated at the continuum level, coupled with replicas of molecular systems of epoxy. Three uncertain parameters are considered: strain rate, temperature, and length (of the sample). We made use of non-intrusive UQP1 to analyze the

uncertainty associated with SCEMa simulations. We also attempted to build surrogate models of the molecular dynamics models to apply semi-intrusive UQP3B. The non-intrusive UQ relied on the building of a GP surrogate model which showed good agreement with SCEMa's predictions of the force applied on the dogbone sample to strain it.

Further, SCEMa was enhanced by a clustering algorithm enabling to avoid unnecessary redundant MD simulations, based on a redundancy threshold chosen according to the desired accuracy. In turn, we applied the VVP2 (Level of Refinement) to verify the convergence of the clustering algorithm with respect to the similarity threshold parameter.

Please refer to Appendix 4.7: Materials for a detailed description of this work.

2.3.9 Binding Affinity Calculator (BAC)

The BAC application is a workflow tool that runs and analyses simulations designed to assess how well drugs bind to their target proteins and the impact of changes to those proteins. It is a collection of scripts which wrap around common molecular dynamics codes to facilitate free energy calculations. Use of ensemble simulations to robust, accurate and precise free energy computations from both alchemical and end-point analysis methodologies [12]. In the standard BAC protocol, an ensemble of 25 replicas is used for each of the parametric configurations. The starting phase spaces are close to each other for the replicas, differing only in their initial velocities which are generated independently from a Maxwell–Boltzmann distribution. Each molecular system is then virtually heated to a desired temperature, and subsequently maintained at this temperature and a defined pressure (with temperature and pressure coupling constants). Ever since the fast-track implementation reported in D4.2, BAC has used the NAMD code to explore the high-dimensional input parameter space using non-intrusive UQ with adaptive Stochastic Collocation. The dimension reduction and petascale need for UQ in molecular dynamics will be addressed in Scaling for EasyVVUQ.

2.4 The study of VECMA toolkit scalability and performance needs in applications equipped with UQ and V&V

Since we are targeting exascale within VECMA, we need to consider whether the tools are ready for the applications' anticipated exascale needs. In this section, we describe the collaborative effort between WP 3, 4 and 5 to study the performance of three components within the VECMA toolkit, namely EasyVVUQ, QCG-PJ, and FabSim3. Using several application models mentioned in the previous

section as study cases, we explore the scaling of these tools, and address the petascale to theoretical exascale performance.

2.4.1 Scaling for EasyVVUQ

As an example, let us consider the exascale readiness of EasyVVUQ. The first consideration is that we could be using the toolset for a very large number of relatively cheap samples, or at the other extreme, a relatively small number of very expensive calculations. In the latter case, there are no reasons for concern with the EasyVVUQ infrastructure since there is no pressure at all on the toolbox. Therefore let us look at the performance when dealing with a large number of samples.

As a proxy-app for the fusion use case, the fusion tutorial was developed which solves for the steady state temperature profile of a cylindrical plasma. [13] This uses FiPy [14] to solve the differential equation and delivers a solution in about 20 ms, allowing for a large number of cases to be running yet delivering data of the same form as the fusion application (a 1D profile of temperature).

An initial examination of the various phases of the calculation using PCE[1] showed that some phases scaled as N^2 whereas the expected scaling was N^1. A small change in the code fixed this, resulting in O(1) scaling for all phases except for the analysis phase which also has a dependency on the number of polynomials in the PCE expansion.

We have demonstrated the ability to run and analyze cases with more than one million samples (where the QoI had 100 values). This case showed a need for a very large amount of memory in the analysis phase which needs to be addressed (this would be alleviated by parallelising this part of the analysis calculation).

Of course, the goal in any EasyVVUQ analysis is to minimize the number of samples needed, and sparse grids or adaptive sampling can help this -- but the above analysis is important since it is likely that the number of samples will be large for complicated cases.

Dimension reduction and the petascale need for UQ in molecular dynamics

Another type of scaling encountered in UQ problems is the ability to deal with high-dimensional input spaces, which is more mathematical in nature rather than computational. As illustrated by the CovidSim work described above, we are implementing UQ methods in both EasyVVUQ and EasySurrogate along these lines.

More recently, we have also completed similar work involving the NAMD molecular dynamics (MD) code, which is much more compute intensive than CovidSim. In particular, we applied dimension-

[1] Polynomial Chaos Expansion, see https://en.wikipedia.org/wiki/Polynomial_chaos

adaptive stochastic collocation to a 14 dimensional space of selected NAMD input parameters. In addition, we also controlled for the aleatoric uncertainty introduced by the random seeds. To do so, we ran 25 replicas (25 random seed values) for every parametric configuration, which we show is crucial to do in MD simulations. Our computational budget of 2M core hours (on the LRZ SuperMUC-NG machine), was depleted after 123 samples from the input distribution. In comparison, we could run a CovidSim adaptive campaign of 3000 input samples using in the order of 10.000 core hours. While the NAMD campaign provided insight into the code's uncertainty, 123 samples were not enough to get fully converged results. Hence, we expect the peta-scale needs of compute-intensive problems with high-dimensional input spaces to (far) exceed 2M core hours, especially if we also need to control for the stochastic (aleatoric) output variability via replicas over random seeds.

2.4.2 Scaling for QCG-PJ

There is currently no machine available that has computing resources with Exaflop/s performance. The most powerful computer currently on the Top500 list (November 2020), Fugaku (RIKEN Center for Computation Science, Japan), has a theoretical computing power of about 0.5 Eflop/s achieved with more than 7 million computing cores. It is worth noting that the processors used in this machine are based on the ARM architecture, which is different from the currently most widely used architecture in HPC systems, i.e. x86, which makes it necessary to adapt applications to make optimal use of such architecture. In order to show the scale and complexity of the problem of exa-scale computing, in the following text we have carried out considerations on how a machine with a theoretical power of 1 EFlop/s could look like using currently available processors. It should be noted that in our considerations we focused on the theoretical efficiency of floating point calculations by the processors' execution units, while we omitted the impact of several elements, such as:

- energy efficiency - the power needed by the computing resources, data storage as well as the cooling systems of such a machine,
- file system performance,
- efficiency of the nodes interconnect network,
- communication libraries enabling the effective exchange of messages between processes of the running application,
- queuing systems capable of efficiently serving such a large machine.

Since all these elements have a significant impact on the performance of calculations, they should be taken into account in a full analysis.

As a basis for the following considerations, we have adopted the Intel® Xeon® Platinum 8268 (Cascade Lake) processor used in the recently commissioned Altair machine (PSNC Poznań) currently ranked

85th on the Top 500 list (November 2020). This processor has 24 cores, each with 2 AVX-512 units. Each AVX-512 unit is capable of performing 16 double-precision number operations per processor cycle. Assuming that the maximum clock frequency of the Intel® Xeon® Platinum 8268 processor when using AVX-512 instructions by all 24 cores is 2.6 GHz, this gives us the theoretical computational performance of a single processor as 24 * 2 * 16 * 2.6 ~ 2 T DP flop/s. (Hereinafter, when using the flop/s notation, we mean the number of double-precision operations per second.)

Assuming that each node of a typical supercomputer will consist of two such processors, its performance will be about 4 Tflop/s. Using simple arithmetic, we can calculate that in order for a machine to have 1 Pflop/s performance we need about 250 such nodes, and in the case of a machine of 1 Eflop/s, the number of such nodes increases to 250 thousand, giving a total of about 12 million computing cores. This is a theoretical maximum computing performance, without taking into account parameters such as the bandwidth of RAM interfaces. In practice, apart from synthetic tests, real applications are not able to come close to such results.

Knowing the number of available cores of a theoretical machine with computational power of 1 Eflop/s, we can try to calculate how long it would take to perform the most computationally complex simulation in the VECMA project. Such a simulation is the UQ scenario of the fusion model, developed at MPG. This simulation, with 8 uncertain parameters, requires the execution of 390625 samples, where each sample uses about 1024 cores. With 12 million computing cores at our disposal, we are able to run 11737 samples simultaneously. In order to execute all required samples we need as many as 34 such run cycles. Depending on the calculation length of a single sample, from 4 to 40 days, this gives a total calculation time of all samples on a 1 Eflop/s machine ranging from 4 months up to 4 years!

Of course, it is practically impossible to use the entire computing machine for such a long period. One of the solutions to this problem is to split the total computation into smaller groups and then run them on the resources provided by multiple computing machines. The QCG-PilotJob software for running multiple applications in a single queue system allocation would make it possible to run multiple samples on a single computing resource. Preliminary tests of this service have shown that it can support up to 20,000 jobs on 29,000 cores with an efficiency of over 95%. Running multiple instances of the QCG-PilotJob service on multiple resources, e.g. using the FabSim3 tool, would make it possible to run computations on distributed computing systems and thus, at least in theory, would make it much easier to achieve exa-scale performance.

2.4.3 Scaling for FabSim3

Many scientific applications need to run large ensemble simulations (1000+ runs) to perform UQ and SA, which cannot be executed as individual jobs on most supercomputers due to scheduler constraints, and require a pilot job mechanism such as QCG-PJ. QCG-PJ has been shown to efficiently execute 10000 jobs with less than 10% overhead, even if those jobs only last for one second each. Please refer to D5.2 for more details.

To improve the FabSim3 ensemble submission performance, we integrated it with QCG-PJ and enabled multi-processing job submission from the local machine. To demonstrate the benefit of this, we measured the total elapsed time of the job submission to a remote machine for the epidemiology [16] application. Owing to limitation of maximum number of submitted jobs per user, i.e., 5000 jobs both in running and pending at a given time, on the Eagle supercomputer, we use ensembles sizes 500 and 5000 runs, and use up to 6 cores for the submission process. We enabled pilot jobs for the larger ensemble for both large and small cases. Figure 4 presents the required time to submit for each job (to be submitted) to a remote machine.

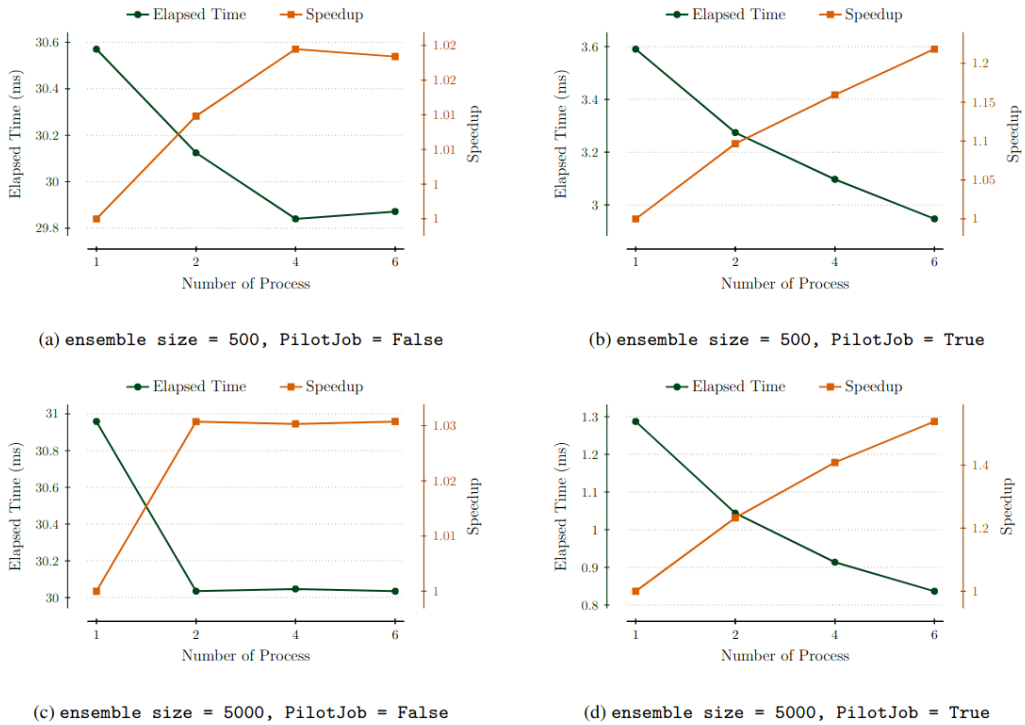


Figure 4: Time required to submit per each job with FabSim3 (with/without QCG-PJ) relative to the number of cores used for job submission. Graph is made using an average of 10 repetition of each ensemble size. Please note that, here we only measure the job submission overhead, so queuing time and job execution on computing nodes are not considered in our test.

As it shows, for a single core, by enabling QCG-PJ during job submission, the job submission overhead is around 30 milliseconds. When using FabSim3 with QCG-PJ, the job submission per job reduces to around 1.7 milliseconds. Compared to previous releases, the job submission, using a single core, without and with QCG-PJ was about 2466 milliseconds and 80 milliseconds, respectively. This significant reduction on job submission obtained by the new design implementation in the latest release of FabSim3 toolkit. This benefits users by saving them time when a large ensemble of runs needs to be submitted to the remote machine.

2.4.4 Summary of the scaling

The project has demonstrated the ability to run up to one million tasks in EasyVVUQ, the ability to launch 1000s of jobs in FabSim3 and to run more than 20 000 jobs with QCG-PJ. The limitations in the testing for bigger jobs is not with the VECMAtk tools, but with the computational resources that are currently available to the project. The initial assessment is that the tools are ready for exa-scale, but it could happen that new challenges will emerge when testing at that scale.

3 Conclusions

In this report, WP4 illustrates the use of complex, non-intrusive and semi-intrusive UQ techniques, as well as V&V techniques in 9 multiscale applications: CovidSim, FACS, Dutch-Covid Model, Multiscale Migration Prediction model, Climate application, Fusion application, In-Stent Restenosis 3D, Binding Affinity Calculator using NAMD, and Materials application.

The non-intrusive UQ techniques include utilizing the dimension-adaptive SC sampler available in the EasyVVUQ library to tackle the higher-dimensional uncertain input parameter space. In addition, sensitivity analysis was used to perform dimension reduction in multiple applications. This technique helps reduce the uncertainty in the multiscale models. For example, the Dutch-Covid Model and Multiscale Migration Prediction models use sensitivity analysis to identify the input parameters that have significant effect on the output uncertainties, and therefore increase their confidence in drafting future strategies or allocating resources accordingly.

The advanced VVUQ techniques introduced in D2.1 and D2.2 are also explored by the applications. These advanced techniques include subsampling (UQP-A) and replacing the most expensive submodel with a surrogate (UQP-B). The goal of these techniques, including the SA based dimension reduction method, is to reduce the cost of the UQ simulations. The UQP-A methods include BSpline, sparse grids, and dimension-adaptive sampling. Several metamodeling techniques are also introduced with some

already available in the EasySurrogate library: stochastic surrogates, standard regression surrogates, online learning, reduced surrogates, machine-learning based UQP1, convolutional neural network, and Gaussian processes. Five out of nine of application examples we showcase in this deliverable have incorporated at least one of the above metamodeling techniques into their multiscale models. The applications use these techniques in different settings, including semi-intrusive acyclic (UQP2) and semi-intrusive cyclic (UQP3) workflows. They have demonstrated that these optimization techniques indeed improve the performance of the UQ simulations with qualitatively the same results. Four V&V techniques are available in the forms of VVPs: Stable Intermediate Forms (SIF), Level of Refinement, Ensemble Output Validation, and Quantities of Interest Distribution Comparison (QDC). Three out of four of these VVPs are applied in several applications, while the SIF will soon be implemented in the FACS application. They have the potential to improve the confidence we have (reliability) in simulation models. For example, the QDC method (VVP4) can be used to provide quantitative comparison between data collected from experiments and simulation results, hence helping decrease the uncertainty we have in the simulation model.

Last but not least, with the help from WP3 and WP5, we have explored the scalability of three components from the VECMAtk, including EasyVVUQ, QCG-PJ, and FabSim3. With BAC (using NAMD), fusion, and FACS applications as test cases, the high-dimensionality of uncertain input parameters from these applications require petascale, and eventually exascale, resources. A theoretical estimation of the computational performance needs to exascale resources (projection based on Altair cluster at PSNC) on the fusion application is provided.

Overall, the experiences reported by applications on implementing VVUQ techniques (algorithm developed by WP2 and ready to be accessed from several components within the VECMAtk developed by WP3) have proved helpful in driving the development of these VVUQ methods. Together with WP2, 3 and 5, we are preparing the software, applications, and infrastructure for runs on emerging exascale architectures, when such resources become available.

4 Appendix: Application Examples in Detail

4.1 CovidSim

Non-intrusive UQ (UQP1) for the CovidSim epidemiological code

We applied the UQP1 approach to the well-known COVID19 epidemiological model CovidSim from Imperial College, which has 940 input parameters. Many of these were obviously not relevant. After eliminating these, we were left with 60 inputs. We ran 3 separate dimension-adaptive campaigns on subsets of 20 inputs, which identified important inputs. These were all combined in a final campaign with 19 inputs. Doing so with standard SC or PCE is completely impossible. We also tested our software on the full 60 dimensional input space at once. It can handle this dimension, although we do expect slower convergence, hence the extra step of 3 separate preliminary campaigns. When ran CovidSim in parallel, using 28 cores per code evaluation. The adaptive ensemble execution workflow was performed using the VECMA automation toolkit FabSim3, and the HPC machine we used was the PSNC supercomputer Eagle. Some results are shown below. We found CovidSim to be quite sensitive to perturbations in the input parameters, and it amplifies uncertainty from the input to the output by roughly a factor of 3.

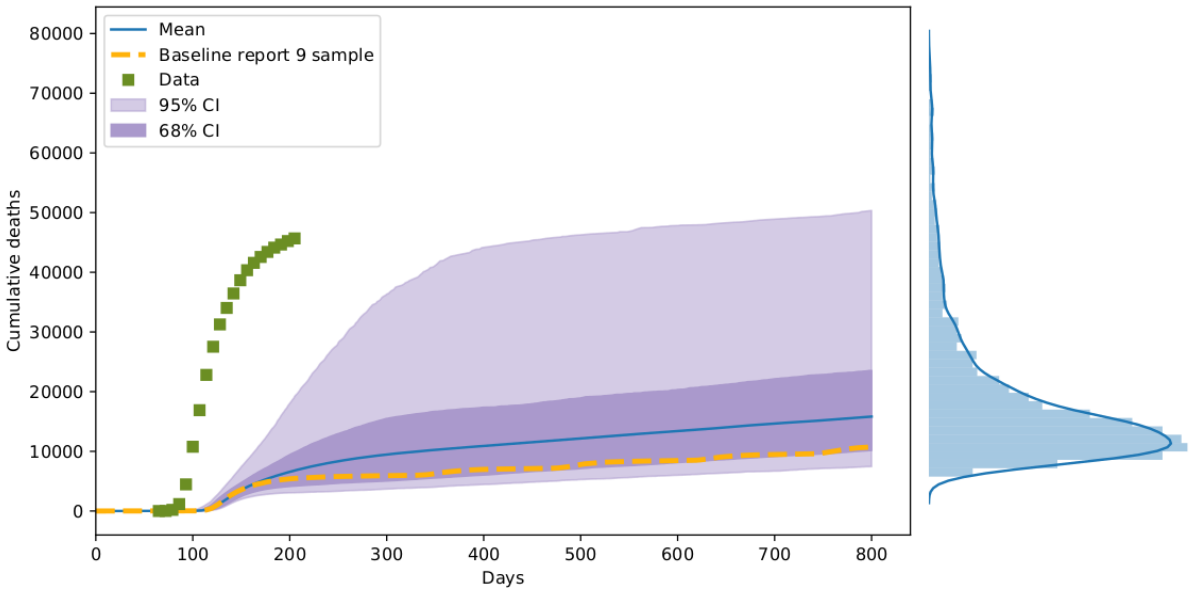


Figure 5: Results from the adaptive ensemble execution CovidSim workflow. Figure is taken from [5].

Machine-learning based UQP1 example for the CovdiSim epidemiological code

Here we will use EasySurrogate as a UQP1-type surrogate for the well-known CovidSim epidemiological code. We chose 19 different inputs to be uncertain, which can be considered as a high-dimensional input space. We published a UQP1 study on this problem, where we utilized the dimension-adaptive

version of the EasyVVUQ Stochastic Collocation sampler, in order to deal with the high-dimensional nature of the problem. Standard SC / PCE methods cannot deal with this number of inputs. However, neural networks can easily handle a 19 dimensional input vector. Here, we therefore first used EasyVVUQ to sample CovidSim 1500 times using a simple Monte Carlo approach. We used FabSim3 to submit the samples to the PSNC Eagle supercomputer. The resulting data frame of input-output samples was then fed to the Deep Active Subspace implementation of EasySurrogate. The surrogate first reduced the 19 dimensional input space down to an active subspace of 7 dimensions, before propagating further through the network. The confidence intervals of both the 1500 CovidSim code samples, and the intervals computed from samples of the surrogate are shown below. They show good qualitative agreement, except the confidence intervals of the code samples came with a computational cost of approximately 7000 core hours, while the intervals of the surrogate were computed in a matter of seconds on a laptop.

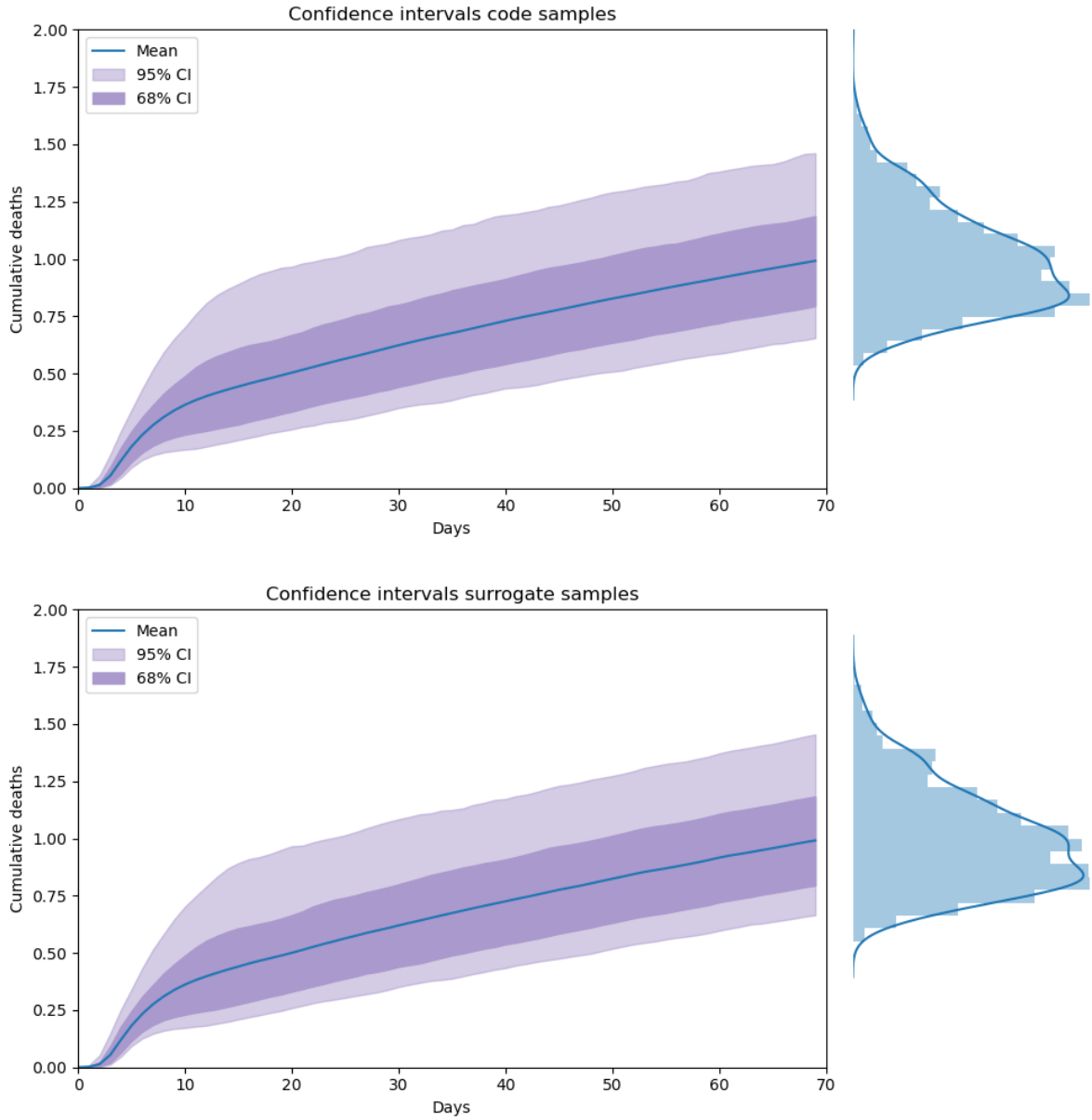


Figure 6: The confidence intervals of both the 1500 CovidSim code samples (top), and the intervals computed from samples of the surrogate (below).

4.2 Dutch-Covid Model

SA-based dimension reduction: an application to COVID-19 exit strategies (UQP1)

We used EasyVVUQ (in combination with FabSim and QCG-PJ) to assess uncertainties and sensitivities of four exit strategies implemented in an individual-based transmission model. The study is focused on the COVID-19 pandemic in the Netherlands, but it can be extended also to other countries and to other similar diseases. As intensive care (IC) capacity is limited, we analyzed the question whether the considered exit strategies can avoid overburden of the healthcare system. We therefore looked at the

distributions of two quantities of interest (QoIs): the maximum of the moving average of the prevalent cases in IC and the total number of IC patient-days in excess of IC bed capacity. The first QoI gives an indication of the intensity of an outbreak, while the second quantifies by how much the total IC capacity is overburdened.

We used Monte Carlo sampling to evaluate how uncertainties in model parameters values or due to the inherent stochastic structure of the model can affect the outcome of a given strategy, therefore assessing the robustness of such strategy. Our results showed that for all the considered strategies there are big uncertainties in the model outcome and all strategies have low probabilities to avoid healthcare overburden, with only Contact Tracing getting close to 50% chances. Furthermore the shape of the cumulative distribution functions of the QoIs is mostly determined by the parameters directly related to the strategy itself and only weakly by uncertainties in other parameters such as, for instance, the basic reproduction number of the disease.

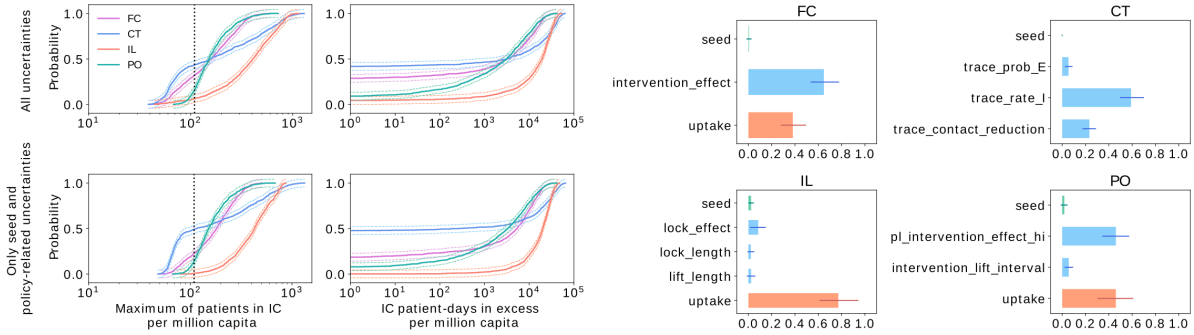


Figure 7: SA-based dimension reduction to Dutch-Covid Model. This figure is taken from [17].

Although important when setting, e.g., a minimum level of intervention, these non-policy-related parameters might be kept fixed, thus strongly reducing the computational burden, in sensitivity studies meant to determine which parameters of a specific strategy induce more variance in the model output. A sensitivity analysis based on the first order Sobol indices of the policy-related parameters revealed the crucial role of the effect of interventions, of the intervention uptake by the population and of the capability to trace infected individuals. The intrinsic stochasticity of the model instead does not induce much variability in the QoIs. This kind of information can be used either to decompose a model parameter (e.g. the effect of intervention) in a set of smaller processes (e.g. working from home, closure of schools), or to inform policy makers such that resources are disposed in an optimal way or such that target-specific interventions are introduced [17].

4.3 Multiscale Migration Prediction

SA-based dimension reduction: An application to migration prediction

For forced migration sensitivity analysis (SA), we use an automated execution environment where we combine the Fabric for Flee Simulation (FabFlee) plugin [18], the EasyVVUQ toolkit [19] and QCG-PilotJob [20]. FabFlee is an integration of Fabric for Simulation (FabSim3) [21] toolkit and the Flee agent-based simulation code. To identify the influential (or pivotal) parameters in the Flee algorithm, we perform SA on parameters presented in Table 3. We executed EasyVVUQ script, which has the stochastic collocation sampler with Sobol's indices and obtained simulation results for four African conflict scenarios with different simulation periods. We simulated Mali for 300 days from the 29th February 2012, the Burundi Conflict for 396 days from the 1st May 2015, the South Sudan instance for 604 days from the 15th December 2013 and the Central African Republic (CAR) for 820 days from the 1st December 2013. We chose these simulation periods based on conflict progressions that forced people to flee from their origins to neighbouring countries.

Table 3. Defining parameter space for the uncertain parameters of forced migration simulation.

Input parameters	**Type**	**Min value**	**Max value**	**Default value**	**Uniform range**
max_move_speed	float	0.0	40 000	200 km/day	(20, 500)
camp_move_chance	float	0.0	1.0	0.001	(0.0, 0.1)
conflict_move_chance	float	0.0	1.0	1.0	(0.1, 1.0)
default_move_chance	float	0.0	1.0	0.3	(0.1, 1.0)
camp_weight	float	1.0	10.0	2.0	(1.0, 10.0)
conflict_weight	float	0.1	1.0	0.25	(0.1, 1.0)

EasyVVUQ performs SA and provides the Sobol indices results for six input parameters and we calculated the average difference for each parameter presented in Table 4. We observed that the most influential parameters are *default_move_chance* and *camp_move_chance* and highly sensitive for all four conflict simulations. The *max_move_speed* parameter is also pivotal as the Sobol indices for Mali, South Sudan and CAR are 0.2249, 0.1788 and 0.1617 respectively and greater than 0.05 contributing

to the simulation output. While *conflict_move_chance* and *camp_weight* have slight impact compared to earlier mentioned parameters with the average differences of 0.0829 and 0.0628. The only input parameter with the lowest values is *conflict_weight* that is a non-influential variable for all simulation instances. The Sobol indices for the combination of parameters are non-influential to the simulation output as they are below 0.01.

Table 4: FabFlee and EasyVVUQ input parameter exploration results for six parameters of forced migration.

Input parameters	**Mali**	**Burundi**	**South Sudan**	**CAR**	**Average**
max_move_speed	0.2249	0.0485	0.1788	0.1617	0.1534
camp_move_chance	0.3427	0.4404	0.0766	0.1945	0.2635
conflict_move_chance	0.0739	0.0233	0.0477	0.1869	0.0829
default_move_chance	0.2273	0.4121	0.4891	0.2503	0.3447
camp_weight	0.0396	0.0526	0.0835	0.0758	0.0628
conflict_weight	0.0288	0.0022	0.005	0.0315	0.0168

According to the obtained sensitivity results, we refined *default_move_chance*, *camp_move_chance* and *max_move_speed* to improve the reliability and validity of our simulations. We changed the default value of *max_move_speed* to 420 km/day in the EasyVVUQ script. We included a new parameter *max_walk_speed*, which has an agent's speed movement of 35 km/day as a default value. The uniform distribution range is also modified for the *max_move_speed* parameter, which starts from 100 km/day and specifies the range for *max_walk_speed* varying from 10-100 km/day. The remaining four parameters are unchanged but we improved the algorithm ruleset of Flee.

We executed the refined parameters and obtained the second iteration results for the distribution of forced migrants. It provided new Sobol indices results for seven input parameters and we calculated the average difference for each parameter presented in Table 5. We found that the Sobol indices values for the new parameter, namely *max_walk_speed*, is 0.0 for all four countries and, thus, are not influential in the simulation output. However, the Sobol indices for *max_move_speed* are lower than the indices in the previous results, but still have an impact on output. The movement chance of locations vary differently across conflict scenarios and are highly sensitive compared to other

parameters. Importantly, the average difference for *default_move_chance* in the second iteration (0.0868) is much lower compared to the initial results (0.3447). Contrarily, the *camp_move_chance* and *conflict_move_chance* parameters have higher sensitivity to highlight their importance as we have not refined these assumptions. Similarly, the Sobol indices values for *camp_weight* and *conflict_weight* have slightly increased for all four conflict simulations. Nonetheless, the *conflict_weight* parameter was the only parameter with the lowest sensitivity in Table 4, which still holds for Burundi, South Sudan and CAR, but has changed for the Mali conflict. The combination of parameters in the second iteration of sensitivity analysis is still non-influential. We observed that all variations of parameter combinations have the Sobol indices below 0.01 and thus, they do not strongly correlate with each other.

Table 5: FabFlee and EasyVVUQ input parameter exploration results for seven parameters of forced migration using the updated algorithm ruleset.

Input parameters	**Mali**	**Burundi**	**South Sudan**	**CAR**	**Average**
max_move_speed	0.1367	0.0556	0.1326	0.0837	0.1021
max_walk_speed	0.0000	0.0000	0.0000	0.0000	0.0000
camp_move_chance	0.3356	0.7242	0.0261	0.1524	0.3095
conflict_move_chance	0.2133	0.0482	0.1968	0.4591	0.2293
default_move_chance	0.0929	0.0447	0.1619	0.0476	0.0868
camp_weight	0.0667	0.0829	0.1558	0.0811	0.0966
conflict_weight	0.0835	0.0071	0.0066	0.0444	0.0354

We reduced the relative sensitivity of two parameters (*default_move_chance* and *max_move_speed*) by respectively 75% and 33%. We attain these reductions by adding a range of new assumptions and logical mechanisms to our simulation algorithm, and including a single additional parameter (the maximum walking speed). We also calculated the mean total error for two sensitivity iterations and another simulation set with the refined parameters retaining their default values across four African countries. The mean total error for sensitivity iterations have decreased after the changes in the logical structure of the algorithm and parameter refinements. For the simulation sets with default values, the

total errors are considerably higher for Burundi, South Sudan and CAR, but lower for Mali in comparison with two sensitivity iterations.

Table 6: The mean total error for sensitivity iterations after changes in logical structure of algorithm and parameter refinements.

Simulations	Mali	Burundi	South Sudan	CAR
SA iteration 1	0.4914	0.3832	0.4926	0.3809
SA iteration 2	0.4101	0.3361	0.4753	0.3257
Default values	0.3122	0.2571	0.5234	0.3378

Example: UQP-A with Flee

In migration prediction simulations, we require a conflict evolution model to predict where violent events are likely to occur next, given a historical occurrence of violence. Bearing this in mind, we implemented a model, namely Flare, which aims to use a range of data sources and algorithms to construct an artificial conflict evolution forecast. It relies on generic growth parameters driven by high-level heuristics since creating accurate conflict simulations raise significant ethical issues.

To exemplify our approach, we simulated the 2012 North Mali Conflict using Flee, which is an agent-based modelling algorithm, and Flare that relies on a stochastic network-based algorithm (Groen et al. 2019). The workflow for the Mali conflict simulation has been automated using a specifically developed FabFlee plugin for the FabSim3 automation toolkit (Groen et al. 2016). It consists of the following steps:

- Twenty executions of the Flare code, each generating a different type of conflict progression. Each run takes around 10 seconds, requires one core, and executions can be done concurrently.
- Conversion of Flare output to Flee input.
- 20 executions of the Flee code, one for each Flare output. Currently, these Flee runs take around 20 minutes using one core, and can be done in parallel using up to 24 cores (with an efficiency of around 70%).
- Collation of the data into a single representation.
- Graph visualization of arrivals, including the uncertainty introduced by the Flare conflict code.

The initial conflict locations in the Mali conflict are Kidal, Timbuktu and Menaka. During each day of the simulation, any adjacent location has a y% probability to become a conflict location, where y is equal to the number of adjacent conflict locations, divided by two. There is a 0% chance of conflict locations appearing if conflict does not take place in an adjacent location, and any conflict location will retain its conflict status for the remainder of the simulation.

We perform our Flee simulations in two locations: on a local desktop, which we use for code development and testing, but also on the 1.3 Pflop/s Eagle supercomputer in Poznan (Top500 2019). We use the latter resource to show that our FabSim3-based workflow implementation can seamlessly switch between local and remote execution, and that we can maintain this workflow in the case that our simulations become more detailed (and expensive).

We present example forecasts of camp population growth for the two largest camps in the conflicts, Fassala-Mbera and Abala, respectively in Figures 8 and 9. In these forecasts, we account for the uncertainty introduced by the stochasticity of our simple conflict evolution model, which is represented using 95% confidence intervals. In our results, we found that the introduction of different conflict evolutions results in a very large uncertainty in the growth of camp populations, particularly in the middle segment of our simulations. This behaviour occurs even though our conflict evolution model has quite restrictive assumptions (e.g., conflicts can only spread to neighbouring locations). In the start-up phase of our simulations, the uncertainty is very small, as new agents require some time to arrive in camps.

In addition, in the case of Abala, the uncertainty gradually decreases after the camp has reached full capacity, as new agents are deterred from entering the camp. When we run simulations repeatedly with the same conflict progressions, camp populations may also vary due to the stochastic nature of the Flee code. However, this 95% confidence interval of these variations is substantially smaller, and always within a few hundred agents of the mean.

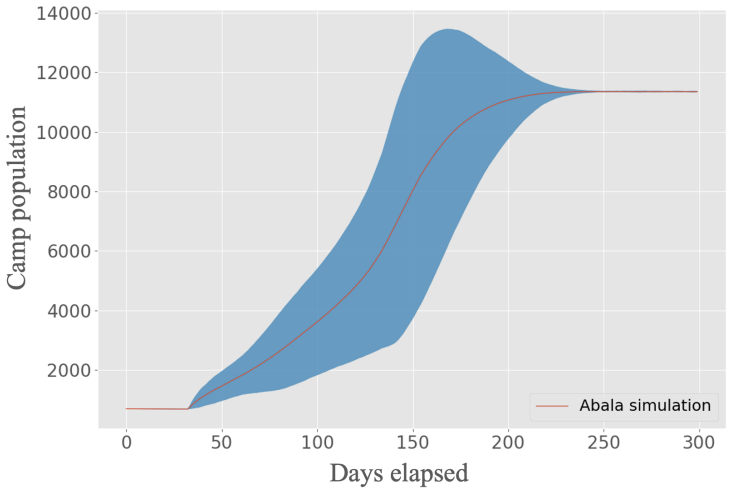


Figure 8: Camp population (value on y-axis) in Fassala and Mbera, the largest camp, as a function of the number of days elapsed. The mean forecast is indicated with a red line, and a 95% confidence interval (two standard deviations) accounting for different conflict progressions is indicated with the blue region.

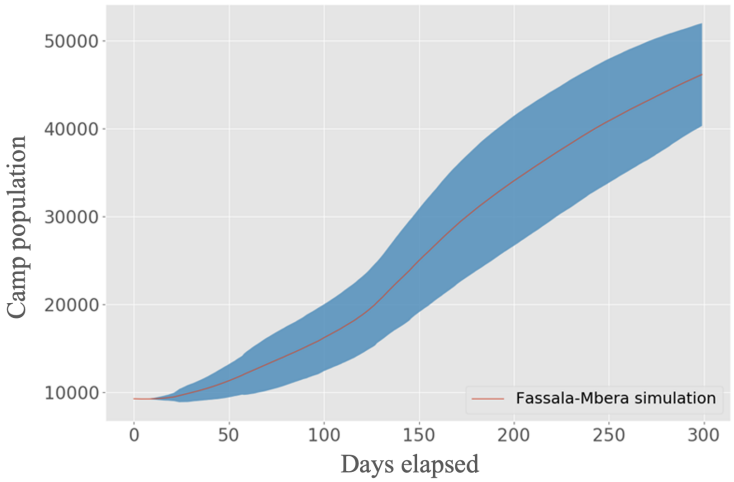


Figure 9: Camp population (value on y-axis) in Abala, the second largest camp, as a function of the number of days elapsed. See Figure 8 for an explanation of the lines and symbols.

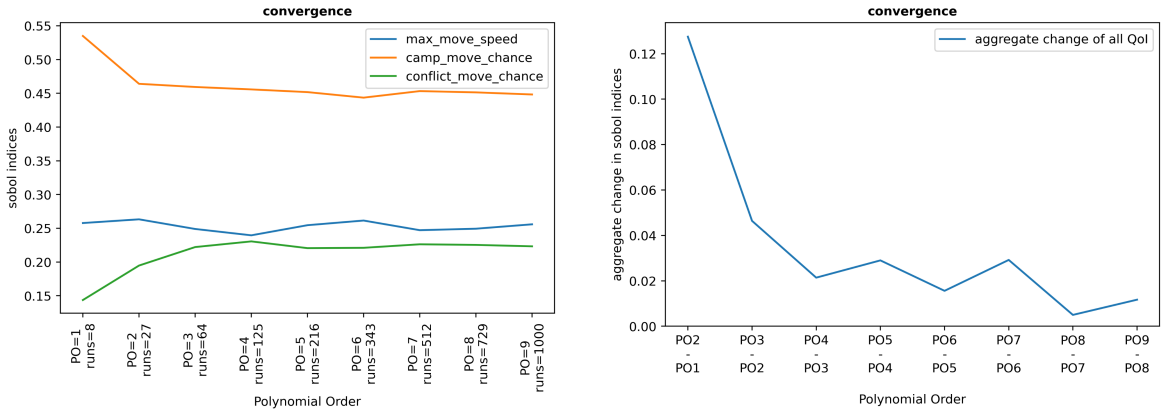
We can also measure the uncertainty caused by a stochastic model for evolving violent events on the output of the agent-based model (populations in camps). We found that a random variation in conflict evolutions has a severe effect on the agent-based model output. In our opinion, this proves that any errors or uncertainty in the forecasting of conflict evolutions will have to be accurately propagated to any forced migration model that is derived from it. It also underlines that we need accurate and robust conflict evolution algorithms, if we want to make actionable migration forecasts in such a setting.

Example: VVP2 with Flee

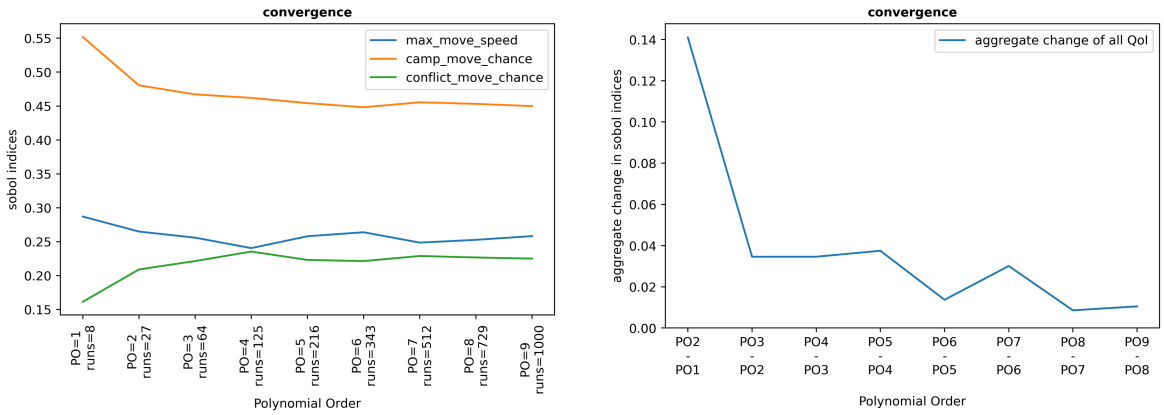
We ran VVP2 with Flee code, within these configurations:

- vary parameters : [max_move_speed, camp_move_chance, conflict_move_chance]
- Polynomial order (PO) range : [1,9]
- QoI : Total Error
- Samplers : SCSampler (Stochastic Collocation sampler) and PCESampler (Polynomial Chaos Expansion)

Based on the PO value and sampler name, a different range of runs are generated. In Figure below, you can see the sobol indices for each vary parameters for different PO numbers (on the left side). Also, the difference between each two PO values are represented as the aggregate change function (on the right side).



Sampler = Stochastic Collocation sampler



Sampler = Polynomial Chaos Expansion

Figure 10: The first-order Sobol indices for each of the uncertain parameters of FLEE.

Example: VVP3 with Flee

The EoV pattern is implemented within the FabSim3 and FabFlee plugin. To demonstrate this pattern, we run an ensemble simulation of 9 runs. After finishing the simulation and fetching the output results, we ran the validation pattern on the output result file per each simulation run and calculated the validation metric and mean score of ensemble runs. The output of this function pattern for FabFlee would be :

~/FabSim3/results/mali_training_hidalgo_1/RUNS/run_5
Validation run_5: 0.2867322888404355
~/FabSim3/results/mali_training_hidalgo_1/RUNS/run_3
Validation run_3: 0.3270998723280654
~/FabSim3/results/mali_training_hidalgo_1/RUNS/run_4
Validation run_4: 0.299040750515153
~/FabSim3/results/mali_training_hidalgo_1/RUNS/run_2
Validation run_2: 0.327737961791057
~/FabSim3/results/mali_training_hidalgo_1/RUNS/run_8
Validation run_8: 0.26853918820856126
~/FabSim3/results/mali_training_hidalgo_1/RUNS/run_9
Validation run_9: 0.26692681315534805
~/FabSim3/results/mali_training_hidalgo_1/RUNS/run_1
Validation run_1: 0.3392997798547688
~/FabSim3/results/mali_training_hidalgo_1/RUNS/run_7
Validation run_7: 0.2797604227928289
~/FabSim3/results/mali_training_hidalgo_1/RUNS/run_6
Validation run_6: 0.2841636513726309

Mean score: 0.29770008098431655

4.4 Climate

Assessing the impact of parameter uncertainty and model switches on the Dutch Atmospheric Large-Eddy Simulation (DALES) code (UQP1)

In this study, we investigated uncertainties in a Large Eddy Simulation of the atmosphere, employing stochastic collocation, which has hardly been used yet in this context. When analyzing the uncertainty

of model results, one can distinguish between uncertainty related to physical parameters whose values are not exactly known, and uncertainty related to modelling choices such as the selection of numerical discretization methods, of the spatial domain size and resolution, and the use of different model formulations. While the former kind is commonly studied e.g. with forward uncertainty propagation, we explore the use of such techniques to also assess the latter kind. From a climate modelling perspective, uncertainties in the convective response and cloud formation are of particular interest, since these affect the cloud-climate feedback, one of the dominant sources of uncertainty in current climate models. Therefore we analyze the DALES model in the RICO case, a well-studied convection benchmark. We use the VECMA toolkit for uncertainty propagation, assessing uncertainties stemming from physical parameters as well as from modelling choices. We find substantial uncertainties due to small random initial state perturbations, and that the choice of advection scheme is the most influential of the modeling choices we assessed [22].

Metamodelling (UQP-B) example with stochastic surrogates for the Lorenz 96 atmospheric model

Below we show the results of two simulations involving the aforementioned stochastic Quantized Softmax Network (QSN) surrogates. Shown are the probability density function (pdf) of the macroscopic state, for both the reference Lorenz 96 system (L96), as well as the system with a QSN surrogate for μ. The left plot shows the results for the case when no time-lagged input features were used, i.e. where the QSN surrogate was Markovian. These results clearly do not match the reference well. The right subplot shows the results for the QSN surrogate with time-lagged feature vectors, showing a clear improvement [23].

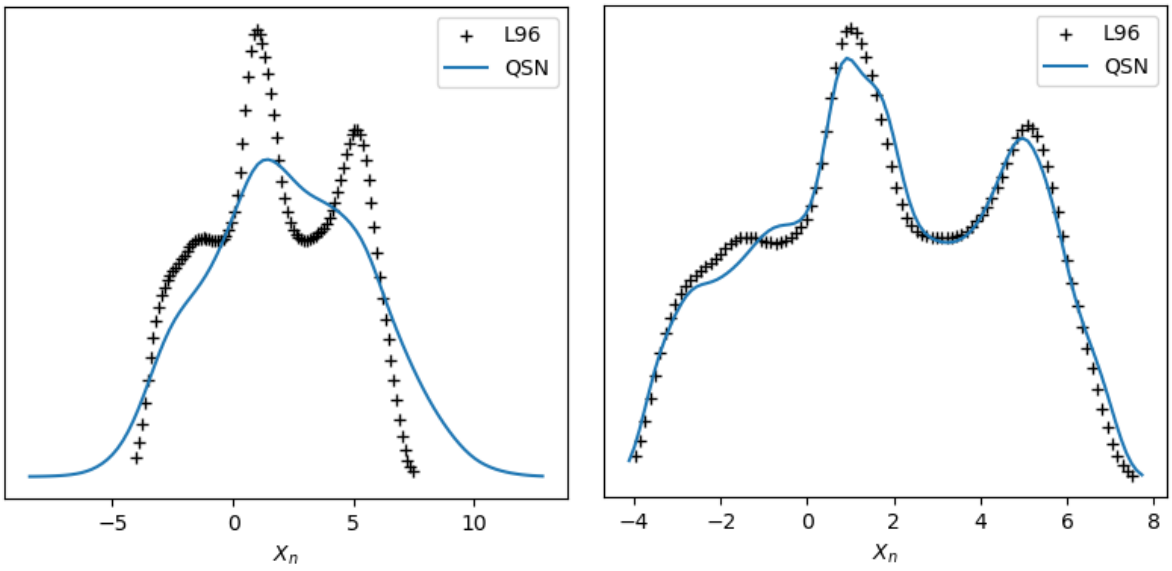


Figure 11: the probability density function (pdf) of the macroscopic state, for both the reference Lorenz 96 system (L96), as well as the system with a QSN surrogate for μ. The left plot shows the results for the case when no time-lagged input features were used, i.e. where the QSN surrogate was Markovian. The right subplot shows the results for the QSN surrogate with time-lagged feature vectors.

Dimension reduction example with reduced surrogates (UQP-B) for the 2D Navier Stokes equations

Applying a surrogate directly to the full state, as we did for the Lorenz 96 case, is more difficult when this state becomes larger. The size of the L96 macroscopic state was an 18 dimensional vector, while the 2D Navier Stokes case we investigated had a state consisting of 64 x 64 points. However, in the results below we decided to focus on pre-defined quantities of interest, in this case the global energy E and enstrophy Z of the system. When we apply the reduced surrogate method, we can compress the training data from 64 x 64 points per time step, down to 2 data points per time step. This is a reduction of 3 orders of magnitude. To show that the reduced training data does not result in a loss of accuracy compared to `perfect' full-field training data, consider the figure below. It shows the pdfs of E and Z for the reference system, and those of the system where the reduced training data was used to inform μ. The pdfs show a near perfect overlap, and the pdfs of a standard eddy viscosity surrogate are also shown as a reference. Next steps would include training a small machine-learning based surrogate on the reduced training data, and extrapolating the system in time, possibly retraining this surrogate in an online fashion as described above [24].

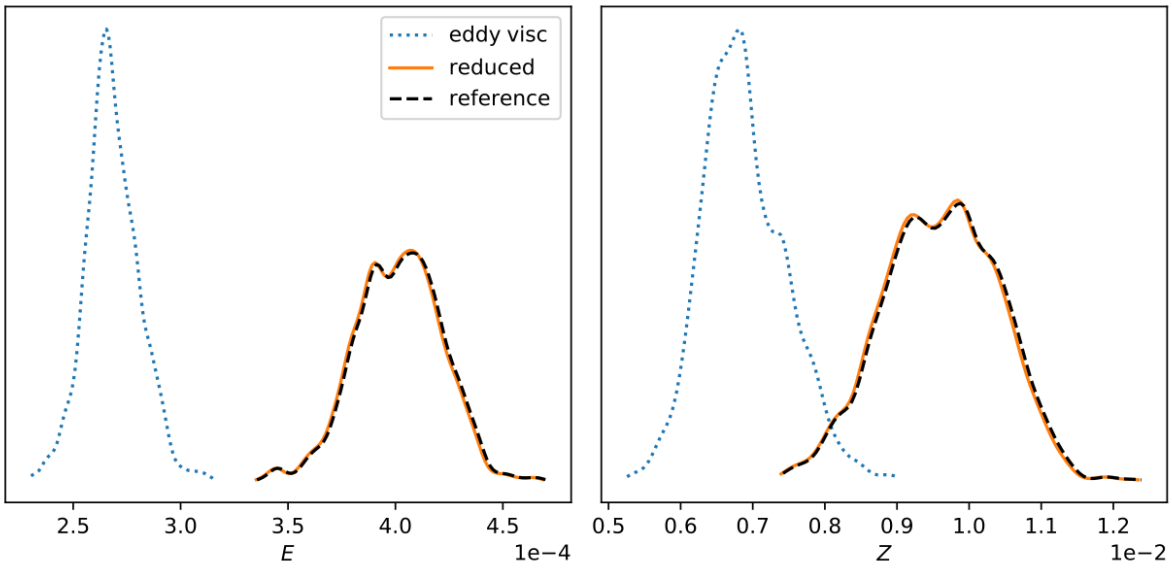


Figure 12: the pdfs of E (left) and Z (right) are plotted as the reference system, and those of the system where the reduced training data was used to inform μ. The pdfs of a standard eddy viscosity surrogate are also shown as a reference.

4.5 Fusion

The plasma dynamics inside a thermonuclear fusion device is very intricate, and it is essential to understand this dynamics before one could predict the device's performance. In particular, the plasma turbulence driven by microinstabilities affects the overall plasma transport, and that could lead to loss of plasma confinement in fusion devices. This is a difficult problem because the spatiotemporal scales involved between turbulence and transport are highly disparate. To help us understand the dynamics, the multiscale fusion workflow (MFW) couples 3 single-scale models (transport, equilibrium, and turbulence) together, along with a module that converts fluxes coming from the turbulence model to transport coefficients required by the transport model.

Further details on the fusion application and its fast-track implementation of the UQ techniques (UQP1) can be found in D4.2.

Efficient sampling (UQP2-A) in acyclic fusion workflow example

To demonstrate the acyclic multiscale case, we present a plasma fusion physics model comprising two single-scale deterministic models: (i) a transport solver, which evolves temperature profiles of the plasma at the macro time scale, and (ii) a 2D equilibrium model, which updates the plasma geometry [25]. These two submodels are performed using serial codes that take respectively 10 ms and 1–5 s per run. Thus the equilibrium model is much more expensive to run computationally.

In the previous work, we have carried out UQ with non-intrusive methods (UQP1) like the quasi-Monte Carlo method and PCE [4,26,27]. With a small number of uncertain parameters we can run each code with a low number of samples. To further improve the computational efficiency of the UQ, here we use UQP2-A by applying UQP1 to each submodel with PCE and exploit the resampling in between submodels to achieve extra speed-up.

We consider six uncertain parameters coming from external heating sources and boundary conditions (4096 samples using PCE with a cubic polynomial [26]). The first UQP1 box (1D transport model) calculates the distribution of electron temperatures T_e as an output. The second UQP1 box (2D equilibrium model) takes T_e as input and calculates the plasma pressure P. In this case, the quantity T_e is a smooth profile evaluated across the radial grid coordinates ρ_{tor}. Thus, the output of the first UQP1 box is a very high-dimensional, strongly (serially) correlated distribution.

In order to reduce both correlation and dimensionality, we approximate each T_e sample with a cubic BSpline [28]:

$$T_e(\rho_{tor}) = \sum_{i=1}^{4} C_i P_i(\rho_{tor})$$

where P_i is a piecewise polynomial of degree 3 and C_i are the spline coefficients. We then resample C_i, allowing us to perform UQ for the equilibrium with less cost and sufficient accuracy.

The descriptive statistics as well as the complete range of the output P with respect to the radial coordinate ρ_{tor} are shown in figure below, and they correspond to the expected values [26,29]. The results between UQP1 and UQP2-A are qualitatively the same: the differences between output means of each UQP are in the order of $O(10^{-5})$. The main advantage in this case study is the flexibility given by the usage of UQP2-A, where we can easily perform a size reduction for an expensive code without affecting the other submodels. In this case, this improved performance by almost a factor of 2 compared to the UQP1 with PCE, which is a fairly good result since the most expensive model (the equilibrium) needs 3 uncertain parameters C_1, C_3 and C_4 ($C_2 = C_1$ to force the derivative to be zero on ρ_{tor} =0) instead of the initial six parameters.

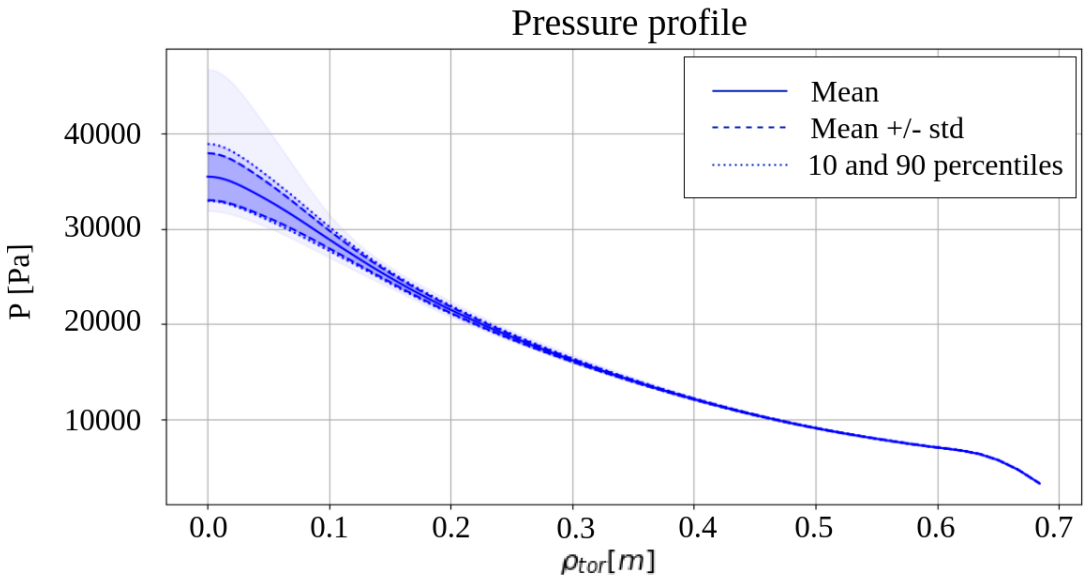


Figure 13: Descriptive statistics and complete range of the pressure P. This figure is taken from [4].

Metamodelling example in Fusion application

For the fusion simulation, we applied metamodeling to analyse the uncertainties in plasma transport introduced by the stochastic effects of turbulence. Instead of calculation of the uncertain intervals for energy transport fluxes based on quadrature schemes with multiple evaluation points, each corresponding for a single run of an expensive turbulence micromodel for certain point of parameter space, we utilised few results of simulations to fit a surrogate model based on Gaussian Processes. These surrogates allow us to compute the statistics of the output quantities of interest, including the sensitivity indices with respect to different input parameters. In case of fusion application, we are interested in measuring how much variance in transport fluxes and transport coefficients are explained by radial core profiles of a tokamak, especially temperature and its gradient for different plasma species. Furthermore, Gaussian Process surrogates are utilised to implement a sequential optimisation scheme which allows us to decide which simulations instances have to be run next to improve the performance of the surrogate. Such optimization leads to gradual reduction of the uncertainty of the surrogate model, as well as a controllable decrease of the error in calculation of the output statistics. The figure below demonstrates a result of such sequential optimisation for a surrogate of GEM0 analytic turbulence code and the dependency of energy flux values on electron temperature gradient for a single plasma flux tube.

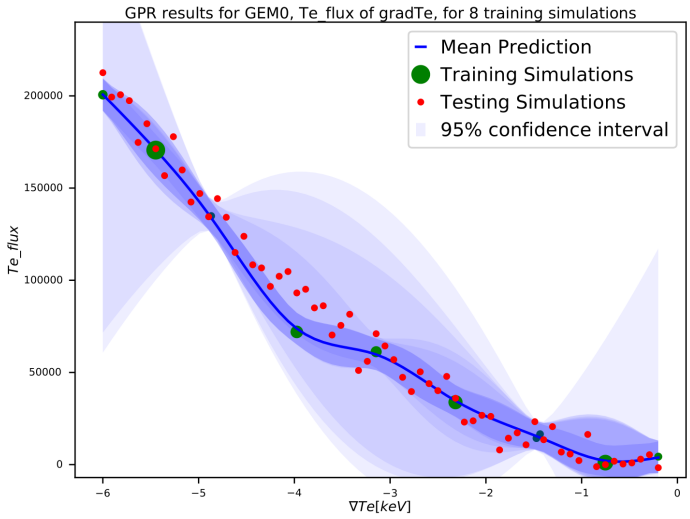


Figure 14: result of sequential optimisation for a surrogate of GEM0 analytic turbulence code and the dependency of energy flux values on electron temperature gradient for a single plasma flux tube.

Temperature distributions comparison (VVP4) example in fusion application

In fusion simulation, similarity measures were utilized to compare two distributions obtained from simulation and experiment data. [30] The MFW runs for multiple iterations, usually at the order of 10^4 - 10^5, until the plasma reaches a quasi-steady state. One way to use the similarity measures is to compare the probability density functions of a quantity of interest, namely, the temperature of one species obtained from the transport single-scale model output, at quasi-steady state to measurements from experiment. The distribution from the simulation is obtained by taking the moments of several hundred data points from a range of simulation time. Measurements of an experiment shot from the ASDEX Upgrade tokamak [31] include electron and ion temperatures, T_e and T_i respectively, across ρ_{tor}. The more conventional similarity metrics such as Hellinger distance, Jensen-Shannon distance, and Wasserstein metrics were utilized initially. The measured distances between experiment and simulation distributions are shown below.

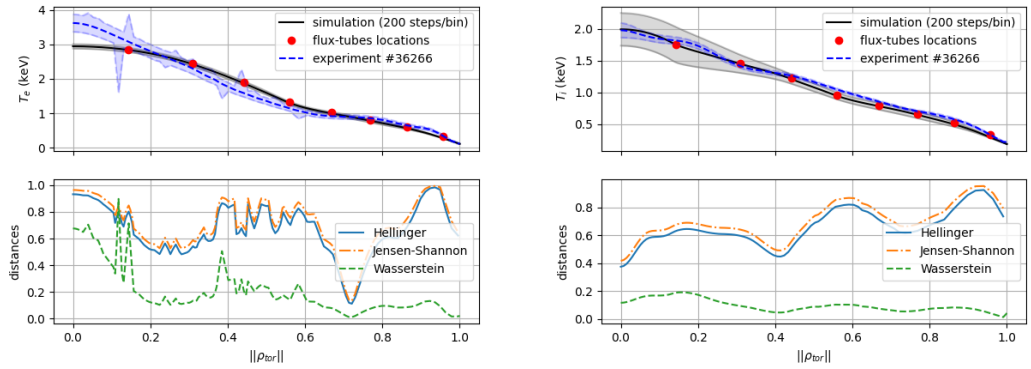


Figure 15: Comparison between experimental and simulated electron (top left) and ion (top right) temperature profiles in ASDEX Upgrade tokamak. The circle markers represent Te at each flux-tube. The shaded regions represent average temperature ± standard deviation. The experiment (dashed trace) is labeled with shot no. 36266. The simulated quasi-steady state T_e and T_i are plotted in solid lines. This figure is taken from [30].

However, the results do not match with what was visually observed between the two distributions. This is due to the metrics being very sensitive to the variance coming from both distributions. Considering both distributions are only approximations, a compatibility measure was developed with adjustable weighting factor (w) that determines the balance between lower and higher moments:

$$w_{tot} = (1-w)\frac{(\mu_2-\mu_1)^2}{2({\sigma_1}^2+{\sigma_2}^2)+(\mu_2-\mu_1)^2} + w\frac{(\gamma_2-\gamma_1)^2}{2({\sigma_1}^2+{\sigma_2}^2)+(\mu_2-\mu_1)^2+(|\gamma_1|+|\gamma_2|)^2}.$$

Compatibility distance (w_{tot}) has value ranges between 0 and 1, with 0 yields two identical distributions and 1 yields two completely different distributions. w can be any value between 0 and 1, and three weighting factor values were tested: 0.05, 0.50, and 0.95.

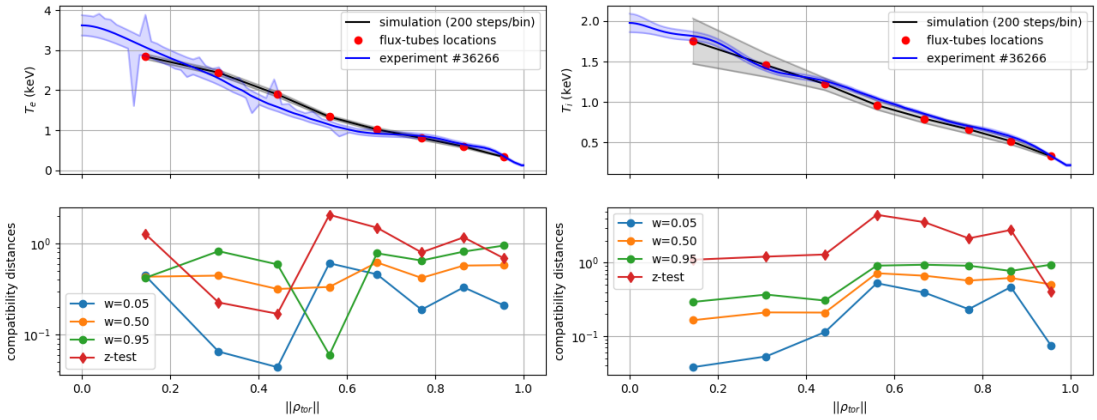


Figure 16: Comparison between experimental and simulated electron (left) and ion (right) temperature profiles in ASDEX Upgrade tokamak (top). The compatibility distance with various weighting factors *w* and z-score are plotted at the bottom panels. This figure is taken from [30].

The result from the compatibility measure with a weighting factor value of 0.05 is the closest to intuition (see top panels). The compatibility distance between the T_e distributions is below 0.5 at every flux-tube except for the one at $||\rho_{tor}|| = 0.561$, which has $w_{tot} = 0.61$. As for compatibility distance between T_i distributions from simulation and experiment, it is always below 0.53. The Z-test was also applied, which treats both distributions as Gaussians, and that also showed a similar outcome as the compatibility measure with a weighting factor of 0.05.

We are currently developing a rubric to interpret the measured w_{tot} values and determine whether the two distributions under comparison are (1) highly compatible, (2) compatible, (3) cautiously compatible, or (4) not compatible. We measure the compatibility between 2 simulation distributions. By comparing these two temperature distributions, we could determine whether the plasma has reached a quasi-steady state. That would mean w_{tot} has to show reasonable compatibility at every flux-tube. Similar practice is carried out between 2 distributions from experiments. Once we establish a rubric, this would help decide the compatibility of simulated temperature distribution with experiment measurement, hence, increase our confidence in simulation results.

4.6 In-Stent Restenosis 3D (ISR3D)

Metamodeling (UQP-B) example in ISR3D application

Coronary heart disease is mainly due to the accumulation and development of atheroscleroitc plaque which narrows the vessel lumen and reduces the flow of blood. It can further evolve into a heart attack or myocardial infarction. One of the common treatments of such disease is percutaneous coronary intervention including coronary angioplasty and stent deployment. However, balloon angioplasty or

stent placement not only removes the fatty substance blocking the blood flow, but also may denude the endothelium layer. It then triggers the inflammatory process involving platelets aggregation, smooth muscle cells activation, proliferation and migration, and extracellular matrix formation. Such a process leading to overgrowth of neointima is known as in-stent restenosis.

To study the mechanism of restenosis, a multiscale computational model, a comprehensive model, In-Stent Restenosis 3D (ISR3D) was developed and validated against in-vivo experimental data. In ISR3D, the blood flow, the neointima, the internal elastic lamina, the media, and the external elastic lamina are explicitly modelled, and the endothelium is modelled as an implicit property associated with the denuded smooth muscle cells.

Forward uncertainty quantification is applied to investigate the influence of uncertain biological parameters on the neointima growth (UQP1). The four epistemic uncertainties considered in the forward propagation include endothelium regeneration time, balloon extension area, the threshold strain for smooth muscle cells bond breaking, and the percentage of fenestration in the internal elastic lamina.

However due to the expensive computational cost of ISR3D (around 800 core-hour per sample), we adapted UQP1-B and applied surrogate modeling to conduct uncertainty quantification. We applied the Gaussian process regression (GPR) method to develop a non-intrusive surrogate model for the ISR3D model. It maps our uncertain inputs to the neointima area (lumen volume). To further improve the computational efficiency, active learning method, Sequential Laplacian Regularized Gaussian Process (SLRGP) [15], is used to select the training points effectively and efficiently. The method is based on the Active Learning Cohn [10] but extends the point selection with laplacian regularisation. The details of the algorithm [15] is presented as following :

Algorithm 1: Sequential Laplacian Regularized Gaussian Process (SLRGP)

Input: Set of n pre-specified points (X)
Output: Design vector for the m evaluated points (Z), Estimated GP (f(x))
1: Determine Z by selecting m points from X using a space filling design
2: **Until satisfying some desired stopping criteria:**
2.1: Optimize the tuning parameter using Algorithm 2
2.2: For each $x_i \in U$ select x^* such that $\text{argmax}_{x_i \in U} \frac{\sum_{j=1}^{q-1}\left(\sigma_Z^2 f(x_j) - \sigma_{Z+x_i}^2 f(x_j)\right)}{q-1}$
2.3: $Z \leftarrow Z + x^*$
2.4: f(x)=$K_{xZ}\left(\sigma_n^2 I + K_{ZZ}\right)^{-1} y$
Calculate MSE
Go to Step 2.1

Algorithm 2: Optimization of the Tuning Parameter of the Laplacian Regularization Penalty (λ_*)

Input: Set of m evaluated points (Z)
Set of q unevaluated points $(X - Z)$
Set of candidate values for tuning parameter (λ_c)
Output: Optimized value of the tuning parameter (λ_*)
1: Initialise $P[i] = 0, i = 1, ..., size(\lambda + c)$
2: **For each** $\lambda \in \lambda_c$**, select** x^* **such that:**
2.1: $\mathrm{x}^* = \mathrm{argmax}_{x_1 \in U} \left[\max \left[K_{(U-x_i, U-x_i)}\right.\right.$
$- \mathrm{K}_{(U-x_i, Z+x_i)} \left[\sigma_m^2 I + K_{(Z+x_i, Z+x_i)} + \lambda K_{(Z+x_i, X)} L K_{(X, Z+x_i)}\right]^{-1} K_{(Z+x_i, U-x_i)}$

$$P[i] = \frac{\sum_{j=1}^{q-1} \left(\sigma_Z^2 f(x_j) - \sigma_{Z+x_i}^2 f(x_j)\right)}{q-1}$$

2.2: where $\sigma_Z^2 f(x_j) = K_{x_j, X_j} - K_{x_j, Z} \left[K_{ZZ} + \sigma_m^2 I\right]^{-1} K_{Z x_j}$
$f(x_j)\, \sigma_{Z+x_i}^2 = K_{x_j, X_j} - K_{x_j, Z+x^*} \left[K_{Z+x^*, Z+x^*} + \sigma_m^2 I\right]^{-1} K_{Z+x^*, x_j}$

$i \leftarrow i + 1$
3: $\lambda_* = \mathrm{argmax}_{\lambda \in \lambda_c}(P)$

To compare the SLRGP with other popular active learning strategies (including, Active learning MacKay (ALM), Active Learning Cohn (ALC) [10], Integrated Mean Squared Error (IMSE), Maximum mean squared error (MMSE)), a 5-fold validation based on ISR3D data was performed to access the surrogates with different AL approaches. We first used the quasi-Monte Carlo method to run ISR3D and generate 128 samples for surrogate training and validation and the experiment was setup as follows: (1) The data was randomly split into k = 5 segments. (2) One segment was set aside for validation while the others were for training. Each AL strategy was then used on the training data to fit a GP, where 20 points first were chosen and the AL strategy chose the following 20 points. Validation metrics were then measured via the validation set. This process was repeated K times such that each segment was in turn used as a validation set. (3) Repeat steps (1)-(2) 30 times.

The result of 5-fold cross validation RMSE scores and maximum error for different AL strategies when building GP surrogates are shown in the following figures.

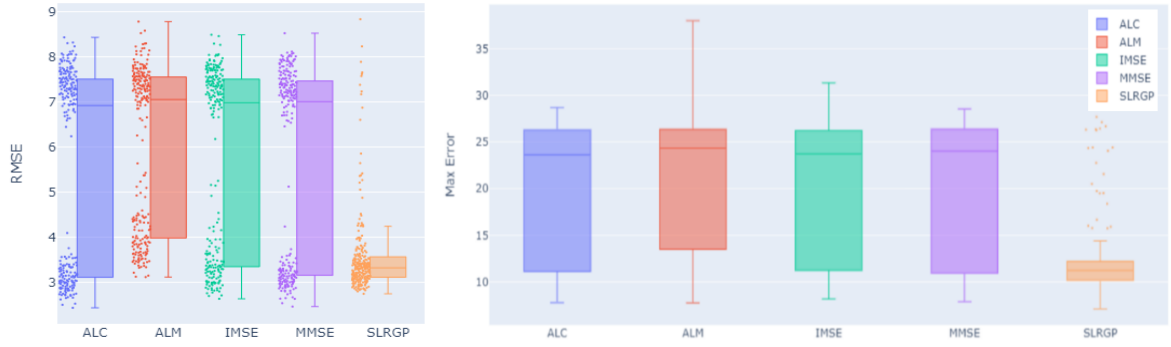


Figure 17: 5-fold cross validation metric scores for different AL strategies when building GP surrogates. Computed using a dataset of 128 ISR3D instances and repeated 50 times, where 20 instances are randomly chosen and the AL strategy chooses the following 20 points. Left) Root mean square error based on the validation dataset. Right) Maximum error based on the validation dataset.

With the surrogate model, we perform the uncertainty quantification with quasi Monte Carlo method. The uncertainty estimates for the QoI distribution with the surrogate modeling is demonstrated in following figures:

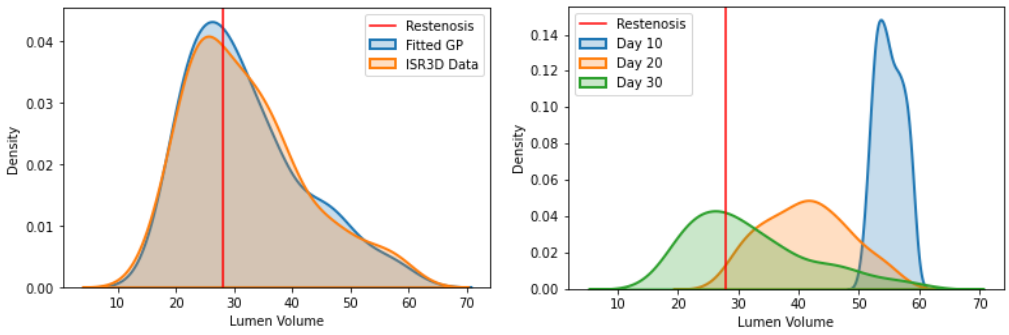


Figure 18: Uncertain distribution of the lumen volume. The red line denotes the threshold from which the result can be regarded as a restenosis. Left) The comparison of the distributions based on qMC between ISR3D and the surrogate model. Right) The uncertain distribution of the lumen area at different timestep (10, 20, 30 days) with the prediction of the surrogate model.

We also start to work on applying UQP3-B to ISR3D to get a more comprehensive understanding about restenosis, for example where the neointimal grows most. These detailed results are not able to be achieved by UQP1 as the only output is a scalar value. With semi-intrusive uncertainty quantification (SIUQ) method, one can preserve most of the mechanism behind the model and find out more useful information in the output. However the semi-intrusive surrogate modeling inevitably introduces error to the uncertain estimates and might accumulate, amplified during further iteration/computation. Therefore it is nontrivial to control as it's an intermediate part of the model. However, by taking the result from SIUQ as low fidelity data and combining it with Monte Carlo results of ISR3D (high fidelity), one could apply the multi-fidelity method [32] to improve the uncertain estimates. This part of research is still ongoing.

4.7 Materials

SCEMa

SCEMa (Simulation Coupling Environment for Materials) is a multiscale simulation tool to predict the properties of materials based on their underlying chemistry and nanoscale structure. SCEMa consists of an implementation of the Heterogeneous Multiscale Method coupling Deal.II (Finite Element Method) and LAMMPS (Molecular Dynamics). It enables simulations coupling semi-concurrently the

evolution of an atomistic and a continuum system. The evolution of the continuum system drives the mechanical evolution of the periodic homogeneous atomistic replicas.

We assumed +/-15% uncertainty in these three parameters, that is an input coefficient of variation of 8.7%. We found that the output coefficient of variation in the predicted force required to strain the material (elasticity) is 13.8%. The multiscale model amplifies the uncertainty by 59%. Further, UQP1 helped perform dimension reduction based on sensitivity analysis. The uncertainty in *temperature* was found to have a 6-fold and 8-fold smaller influence than *strain rate* and *length*, respectively. (see Figure 19)

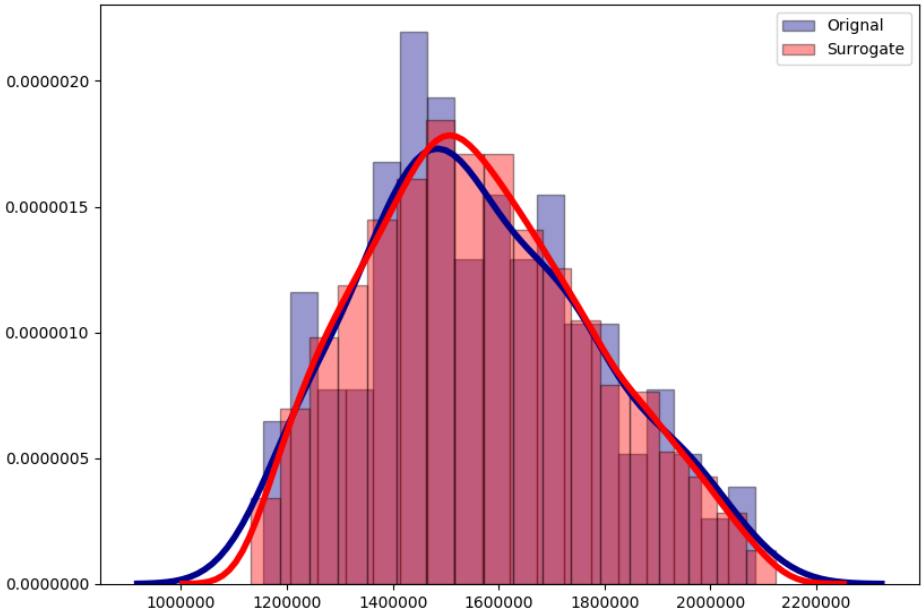


Figure 19: Histograms of applied force predicted by SCEMa (blue) and the non-intrusive GP surrogate model (red) for varying input parameters configuration samples. Continuous lines show the Kernel Density Estimator fit of the distributions.

Level of Refinement (VVP2) in clustering similar thermodynamic states of a piece of material

Let us consider a simple mechanical tensile test applied to a cubic sample of material. We solve the momentum balance PDEs over the volume of the sample using the finite element method. In turn, we are able to estimate the evolution of the global force on the boundary of the sample when a given strain is applied. The procedure to solve the balance PDEs requires us to define the local constitutive behaviour of the material, which relates the local stress with the thermodynamic state of the material (defined by the strain). We introduce a clustering algorithm that prevents from computing the stress for two locations (quadrature points) that have a similar thermodynamic state. In case of multiscale simulations, such an algorithm significantly reduces computational cost. The definition of similarity between two thermodynamic states introduces what we call a *clustering threshold* parameter. With

larger *clustering threshold* values, an increasing number of quadrature points stress estimates are clustered together, but a decrease of the solution accuracy follows. Our goal here is to apply LoR Verification observing the convergence of the error on the global force time series (with respect to the solution in absence of clustering) with decreasing values of the *clustering threshold* parameter.

The global force on the boundary of the finite element mesh is computed as the integral of the force on the nodes located at the loaded boundary (where the imposed displacements are applied).

To execute the LoR VVP we run the following two commands (full code will be updated at https://github.com/wedeling/FabUQCampaign/blob/master/examples/VVP_test/test_clustering.py):

```
from vvp import ensemble_vvp
ensemble_vvp(results_dir, load_forces, check_convergence)
```

Here, `load_forces' just loads the time-series of the global force evolution from a .CSV file in each directory (one directory for each threshold), and `check_convergence` receives a list with all the force time-series and outputs the following (distance/error to the reference force time-series):

```
Threshold value = 1000.00000000
Forces norm with current threshold = 3509168.727 , exact = 3384732.508662 , error = 279922.682395 , relative error = 0.410578
============================================================
Threshold value = 100.00000000
Forces norm with current threshold = 3509168.727 , exact = 3384732.508662 , error = 279922.682395 , relative error = 0.410578
============================================================
Threshold value = 10.00000000
Forces norm with current threshold = 5369131.168 , exact = 3384732.508662 , error = 2835013.531172 , relative error = 3.598374
============================================================
Threshold value = 1.00000000
Forces norm with current threshold = 3156693.912 , exact = 3384732.508662 , error = 370540.329383 , relative error = 0.463393
============================================================
Threshold value = 0.10000000
Forces norm with current threshold = 3374043.545 , exact = 3384732.508662 , error = 20701.854289 , relative error = 0.026235
============================================================
Threshold value = 0.01000000
Forces norm with current threshold = 3382294.906 , exact = 3384732.508662 , error = 4479.336446 , relative error = 0.006436
============================================================
Threshold value = 0.00100000
Forces norm with current threshold = 3384563.179 , exact = 3384732.508662 , error = 322.741383 , relative error = 0.000453
============================================================
Threshold value = 0.00010000
Forces norm with current threshold = 3384730.845 , exact = 3384732.508662 , error = 3.464102 , relative error = 0.000005
============================================================
Threshold value = 0.00001000
Forces norm with current threshold = 3384732.509 , exact = 3384732.508662 , error = 0.000000 , relative error = 0.000000
```

```
=========================================================
Threshold value = 0.00000100
Forces norm with current threshold = 3384732.509 , exact = 3384732.508662 , error = 0.000000 , relative error = 0.000000
=========================================================
Threshold value = 0.00000010
Forces norm with current threshold = 3384732.509 , exact = 3384732.508662 , error = 0.000000 , relative error = 0.000000
=========================================================
Threshold value = 0.00000001
Forces norm with current threshold = 3384732.509 , exact = 3384732.508662 , error = 0.000000 , relative error = 0.000000
=========================================================
```

For *clustering threshold* values below 0.00001 the global force on the boundary appears to have converged to the fully resolved value.

5 References

[1] Groen, Derek, et al. "Introducing VECMAtk-verification, validation and uncertainty quantification for multiscale and HPC simulations." *International Conference on Computational Science*. Springer, Cham, 2019.

[2] Groen, D., et al. "VECMAtk: a scalable verification, validation and uncertainty quantification toolkit for scientific simulations." *Philosophical Transactions of the Royal Society A* 379.2197 (2021): 20200221.

[3] https://www.vecma-toolkit.eu/

[4] Ye D., Veen L., Nikishova A., Lakhlili J., Edeling W., Luk O. O., Krzhizhanovskaya V. V. and Hoekstra A. G. 2021 Uncertainty quantification patterns for multiscale models. *Phil. Trans. R. Soc. A.* 379:20200072. http://doi.org/10.1098/rsta.2020.0072

[5] https://nature.altmetric.com/details/100643772

[6] https://github.com/djgroen/facs

[7] D.Groen, "Simulating Refugee Movements: Where would You Go?," Procedia Computer Science, vol. 80, pp. 2251-2255, 2016

[8] http://www.github.com/djgroen/flee

[9] Suleimenova, D., Bell, D., Groen, D.: A generalized simulation development approach for predicting refugee destinations. Scientific reports 7(1), 1–13 (2017)

[10] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, ‘‘Active learning with statistical models,’’ J. Artif. Intell. Res., vol. 4, pp. 129–145, Mar. 1996

[11] https://github.com/UCL-CCS/SCEMa

[12] https://www.compbiomed.eu/services/software-hub/compbiomed-software-bac/

[13] Suleimenova, D. et. al. Tutorial applications for Verification, Validation and Uncertainty Quantification using VECMA toolkit. Submitted to Journal of Computational Science.

[14] FiPy: A Finite Volume PDE Solver Using Python. J. E. Guyer, D. Wheeler & J. A. Warren, "FiPy: Partial Differential Equations with Python," Computing in Science & Engineering 11(3) pp. 6—15 (2009), doi:10.1109/MCSE.2009.52, http://www.ctcms.nist.gov/fipy

[15] Meka, Rajitha, et al. "An active learning methodology for efficient estimation of expensive noisy black-box functions using Gaussian process regression." *IEEE Access* 8 (2020): 111460-111474.

[16] https://github.com/djgroen/FabCovid19

[17] Gugole, F., Coffeng, L. E., Edeling, W., Sanderse, B., de Vlas, S. J., Crommelin, D. (2021). Uncertainty quantification and sensitivity analysis of COVID-19 exit strategies in an individual-based transmission model. *medRxiv*, https://doi.org/10.1101/2021.03.24.21254218

[18] https://github.com/djgroen/FabFlee

[19] https://github.com/UCL-CCS/EasyVVUQ

[20] https://qcg-pilotjob.readthedocs.io

[21] https://github.com/djgroen/FabSim3

[22] Jansson F, Edeling W, Attema J, Crommelin D. 2021 Assessing uncertainties from physical parameters and modelling choices in an atmospheric large eddy simulation model. Phil. Trans. R. Soc. A. *379*:20200073. (doi:10.1098/rsta.2020.0073)

[23] Crommelin D, and Edeling W. 2021 Resampling with neural networks for stochastic parameterization in multiscale systems. *Physica D: Nonlinear Phenomena*, *422*, 132894. (doi:10.1016/j.physd.2021.132894)
https://www.sciencedirect.com/science/article/pii/S016727892100052X

[24] Edeling W, and Crommelin D. 2020 Reducing data-driven dynamical subgrid scale models by physical constraints. *Computers & Fluids*, *201*, 104470. (doi:10.1016/j.compfluid.2020.104470)

[25] Luk OO, Hoenen O, Bottino A, Scott B, Coster D. 2019 Compat framework for multiscale simulations applied to fusion plasmas. Comput. Phys. Commun. 239, 126–133. (doi:10.1016/j.cpc.2018.12.021)

[26] Lakhlili J, Hoenen O, Luk OO, Coster DP. 2020 Uncertainty quantification for multiscale fusion plasma simulations with VECMA toolkit. In Computational Science - ICCS 2020 (ed. V Krzhizhanovskaya et al.). Lecture Notes in Computer Science, vol. 12143. Cham: Springer.

[27] Richardson RA, Wright DW, Edeling W, Jancauskas V, Lakhlili J, Coveney PV. 2020 EasyVVUQ: a library for verification, validation and uncertainty quantification in high performance computing. J. Open Res. Softw. 8, 11. (doi:10.5334/jors.303)

[28] L Piegl WT. 1995 The NURBS Book, 2nd edn. Berlin, Heidelberg: Springer.

[29] Wright DW et al. 2020 Building confidence in simulation: applications of EasyVVUQ. Adv. Theory Simul. 3, 1900246. (doi:10.1002/adts.201900246)

[30] Luk O. O., Lakhlili J., Hoenen O., von Toussaint U., Scott B. D. and Coster D. P. 2021 Towards validated multiscale simulations for fusion. Phil. Trans. R. Soc. A. 379:20200074. http://doi.org/10.1098/rsta.2020.0074

[31] https://www.ipp.mpg.de/16195/asdex

[32] Peherstorfer B, Willcox K, Gunzburger M. Optimal model management for multifidelity Monte Carlo estimation. SIAM J Sci Comput. 2016;38(5):A3163–94.

[33] Coffeng LE. virsim; 2020. https://gitlab.com/luccoffeng/virsim/-/tree/v1.0.5.