D2.2: Report on advanced multiscale UQ algorithms, including intrusive approaches, and mapping thereof in UQPs and first results on V&V

Due Date	14 June 2020
Delivery	11 June 2020
Submission of updated version	N/A
Lead Partner	CWI
Dissemination Level	Public
Status	Final
Approved	Executive Board
Version	V1.5

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
06.04.2020 v1.1	D. Crommelin	Skeleton
20.04.2020 v1.2	W. Edeling, A. Nikishova, S. Wan, R. Sinclair, D. Ye, D. Suleimanova	Addition of main text
23.04.2020 v1.2	D. Crommelin, W. Edeling	Merging text additions, editing
28.04.2020 v1.3	F. Jansson, W. Edeling	Review and feedback
29.04.2020 v1.3	D. Crommelin	Incorporating comments, finalize draft
15.05.2020 v1.3	O. Luk, D. Coster	Review and feedback
18.05.2020 v1.4	J. Lakhlili	Text addition
20.05.2020 v1.4	D. Crommelin, W. Edeling	Editing
22.05.2020 v1.4	P. Coveney, X. Gu	Editing
09.06.2020 v1.5	V. Krzhizhanovskaya	Text addition, editing
10.06.2020 v1.5	D. Crommelin	Final editing

CONTRIBUTORS

1. Daan Crommelin (CWI) – Editor, author
2. Wouter Edeling (CWI) - Author
3. Shunzhou Wan (UCL) - Author
4. Robert Sinclair (UCL) - Author
5. Anna Nikishova (UvA) - Author
6. Dongwei Ye (UvA) – Author
7. Diana Suleimenova (UBRU) - Author
8. Fredrik Jansson (CWI) – Review
9. Onnie Luk (MPG) – Review
10. David Coster (MPG) – Review
11. Jalal Lakhlili (MPG) - Author
12. Valeria Krzhizhanovskaya (UvA) – Author, editor

Disclaimer

TABLE OF CONTENTS

1 Executive summary 4
2 Main body of the report 5
2.1 Introduction 5
2.2 Advanced multiscale UQ algorithms 5
2.2.1 Surrogate modelling 5
2.2.2 Sensitivity analysis based dimension reduction of multiscale models 10
2.3 Uncertainty Quantification Patterns 11
2.4 Verification & Validation procedures 13
2.5 Implementation of UQPs and VVPs in VECMAtk 14
2.6 Towards the exascale: scalable UQPs 15
3 Conclusions 16
4 Appendices 17
4.1 CNN surrogate model and its application to UQ analysis of the ISR2D model 17
4.1.1 Technical details of the CNN surrogate model 17
4.1.2 Application to UQ analysis of the ISR2D model 17
4.2 Two examples of SA-based dimension reduction 19
4.2.1 Example 1: Reaction equation 19
4.2.2 Example 2: Standard Ornstein-Uhlenbeck process 20
4.3 UQP2 for the fusion application 22
4.4 V&V patterns 23
4.4.1 Pattern: Quantity of Interest 23
4.4.2 Pattern: Stable Intermediate Forms 23
4.4.3 Pattern: Level of Refinement 23
4.4.4 Pattern: Ensemble Output Validation 25
5 References 26

1 Executive summary

In this deliverable we report on the progress and results obtained in Work Package 2 during the second year of the VECMA project. More specifically, we discuss current results from task 2.3 (*Advanced multiscale UQ algorithms, including intrusive approaches*) and task 2.4 (*Advanced UQP development based on task 2.3*). These tasks started in the second year of VECMA and are scheduled to continue until the end of the project (M36).

We have developed new, advanced algorithms to perform uncertainty quantification (UQ) for multiscale models, using novel methods for surrogate modelling and sensitivity analysis. In a multiscale context, surrogate models can be used to replace expensive single-scale model components, thereby speeding up computations and enabling UQ tasks that require many model evaluations. In many situations, the single-scale component possesses intrinsic uncertainty, for example because of internal chaotic behaviour. This uncertainty can be accounted for with a stochastic surrogate, constituting an intrusive approach at the level of the single-scale component. Furthermore, we developed an approach to employ sensitivity analysis for model reduction of multiscale systems, thereby facilitating more efficient uncertainty estimation.

We discuss how these algorithms connect to the family of UQ patterns that was discussed extensively in a previous deliverable (D2.1). They are being implemented in the “EasySurrogate” toolkit, thereby becoming part of the overall software toolkit VECMAtk. Furthermore, we report on our results setting up multiscale Verification & Validation (V&V) procedures and their implementation in VECMAtk. Finally, we discuss scaling of UQPs towards the exascale.

Altogether, WP2 and its tasks 2.3 and 2.4 are well underway. As both tasks continue until M36, we expect further results beyond what is reported here. For the third year of VECMA we aim to consolidate our newly developed algorithms, streamline their software implementation and integrate them further into the VECMA tookit.

2 Main body of the report

2.1 Introduction

VECMA project Work Package 2, entitled *Algorithms & Formalisms*, is focused on designing and formalizing algorithms for multiscale VVUQ (Verification, Validation and Uncertainty Quantification). Task 2.3 (*Advanced multiscale UQ algorithms, including intrusive approaches*) and task 2.4 (*Advanced UQP development based on task 2.3*) started as planned in M12 and M16, respectively. In this report we present the results obtained to date under these two tasks, while we note that both are scheduled to continue until M36 so that further results beyond those reported here are to be expected.

Previously, in VECMA deliverable D2.1 we reported on the conceptual basis and first results of WP2, including discussions of levels of intrusiveness of UQ methods, of the Multiscale Modelling and Simulation Framework (MMSF), and of a family of UQ patterns (UQPs) distinguished by their control structures and optimization dimensions (see Table 1 of D2.1). Here we build on the conceptual groundwork laid in D2.1. We present newly developed algorithms for surrogate modelling and sensitivity analysis, aimed at multiscale models. Next, we discuss translation and generalization of these algorithms into UQPs. Furthermore, we report on our results setting up multiscale V&V procedures. The implementation of UQPs and VVPs in the VECMAtk software toolkit is discussed, as is their scaling towards the exascale.

2.2 Advanced multiscale UQ algorithms

2.2.1 Surrogate modelling

2.2.1.1 Introduction

The construction and use of surrogate models (also referred to as metamodels or emulators) is a central computational strategy in UQ [1]. A surrogate model is trained or fitted to the output of a limited number of evaluations of an expensive computational model. Once trained, the surrogate can replace the expensive model and thereby enable tasks that require many model evaluations, e.g. detailed assessment of forward uncertainty propagation, or Bayesian model calibration. In the family of UQPs described in D2.1, surrogate modelling is a category of optimization UQPs, denoted UQP-B.

In the context of multiscale systems, surrogates are particularly interesting, because simulation with multiscale models is often very expensive. Besides the simple approach of replacing the entire multiscale model by a surrogate, a more advanced approach is to use surrogates for replacing

expensive single-scale model components. Doing this in an adequate manner can be complicated because of interdependencies between model components. In Figure 1 a schematic example is shown of a macromodel M and a single micromodel μ with a cyclic coupling (so M and μ are mutually dependent). In many practical cases, a single-scale model which represents the small spatial and/or temporal scales, i.e. μ in Figure 1, takes up most of the computational burden. It is therefore the most natural target for replacement by a surrogate, denoted $\tilde{\mu}$. When the response of μ that feeds back to 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 $\tilde{\mu}$. This can be viewed as an intrusive approach at the level of the single-scale component, see also D2.1.

The approximation errors made by the surrogate can amplify over time; however, this situation is alleviated for two reasons. First, the Quantities of Interest (QoIs) are often outputs of the macromodel, not of the micromodel. Secondly, in many applications we can tolerate an instantaneous error in the outputs of the macroscopic model. Let us denote the macroscopic outputs of the (M, μ) system by Q, and let $\tilde{Q}$ correspond to the output of the $(M, \tilde{\mu})$ system, see also Figure 1. Then our goal is to create a surrogate such that $\tilde{Q}$ preserves some overall characteristic of Q. An example is the climate application, where we wish to create a microscopic surrogate which preserves the time-averaged climate statistics of the macro model.

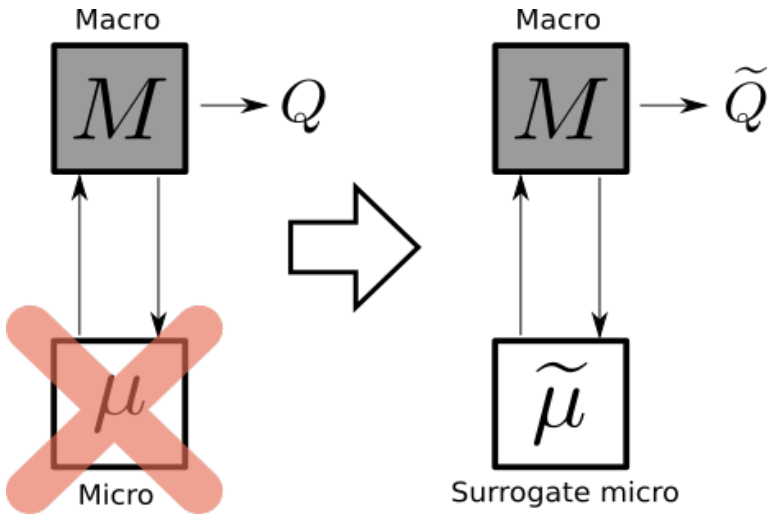


Figure 1: Schematic of a multiscale model consisting of a macromodel M coupled to a micromodel μ. The QoI (Quantity of Interest), denoted Q, is an output of the macromodel. The computationally most expensive component (often the micromodel) can be replaced by a surrogate.

Some techniques to construct surrogates that are well-established in the UQ domain are Non-Intrusive Spectral Projection (based on Polynomial Chaos Expansion), interpolating polynomials resulting from stochastic collocation, and Gaussian Process regression (aka Kriging). They are not specifically aimed at the multiscale setting however. A step forward was the semi-intrusive approach [2] (also reported in D2.1) where it was shown how these existing techniques can be successfully used as elements in a multiscale UQ framework.

Below we give an overview of newly developed, advanced techniques to obtain a surrogate model $\tilde{\mu}$. Specifically, we discuss:

1. Stochastic surrogates,
2. Reduced surrogates,
3. Surrogates based on convolutional neural networks

2.2.1.2 Stochastic surrogates

A given macroscopic state can correspond to multiple microscopic states. Therefore, we develop methods for stochastic surrogate modelling (or stochastic parameterization) of the microscopic model, which take the uncertainty in the microscopic state into account. Specifically, we are developing approaches that *resample* μ data coming from a reference simulation, when conditioned on macroscopic states. Let X be some collection of macroscopic variables. This could include the QoI Q, although not necessarily. In general, our surrogate $\tilde{\mu}$ takes the form of a conditional probability density function, i.e.

$$\tilde{\mu}_{j+1} \sim \mu_{j+1} \mid \tilde{X}_j, \tilde{X}_{j-1}, \tilde{X}_{j-2}, \dots, \tilde{\mu}_j, \tilde{\mu}_{j-1}, \tilde{\mu}_{j-2}, \dots \qquad (1)$$

Here, the index j corresponds to a given time t_j. Thus, in addition to a stochastic nature, we also have the option of embedding *memory* into the surrogate by conditioning on multiple time steps into the past. This is especially relevant when there is no clear time scale separation between the macromodel and the micromodel. In essence, by conditioning as $\mu_{j+1} \mid \tilde{X}_j, \tilde{X}_{j-1}, \tilde{X}_{j-2}, \dots, \tilde{\mu}_j, \tilde{\mu}_{j-1}, \tilde{\mu}_{j-2}, \dots$ we identify a *subset* of candidate μ_{j+1}reference samples, from which we randomly sample one value (i.e. $\tilde{\mu}_{j+1}$) to be used as the prediction for the next time step t_{j+1}.

Eq. (1) describes a *class* of different models. We implemented a model based on the so-called "binning" concept from [3], see [4]. Here, the space of conditioning variables is discretized into a set of non-overlapping bins, where each bin contains a given number of reference samples from μ. This is a direct

way to identify the required subset of reference samples, since the conditioning variables will lie inside a single bin at every time step. The results of the implementation were positive [4]. Notwithstanding this, a downside of the approach is that it is subject to the curse of dimensionality, since the number of bins grows *exponentially* with the number of time-lagged conditioning variables.

To circumvent this problem, we developed a conditional resampling model based on probabilistic classification via machine learning [7]. Now, instead of binning the conditioning variables, we bin the *output* (i.e. the reference μ samples), into K non-overlapping bins. The advantage is that this avoids the curse of dimensionality, since we do not include memory in the output, i.e. the number of bins remains equal to K. We now use a neural network to learn a discrete Probability Mass Function (PMF) over the Koutput bins, conditional on the time-lagged macroscopic input features. At any time step, we can sample a bin index from this PMF, and subsequently resample μreference data from the designated bin, see Figure 2.

We applied these stochastic surrogates to problems in the context of climate modelling. As mentioned earlier, the goal here is to obtain a surrogate such that the overall, time-averaged statistics of the macroscopic solver are accurately captured. The results so far are positive, when applied to a simplified atmospheric model [7] and to a more complex two-dimensional ocean circulation model [4]. Furthermore, we are extending the neural network approach to include a kernel-mixture network [14], enabling construction of a continuous Probability Density Function (PDF) instead of the discrete PMF used until now.

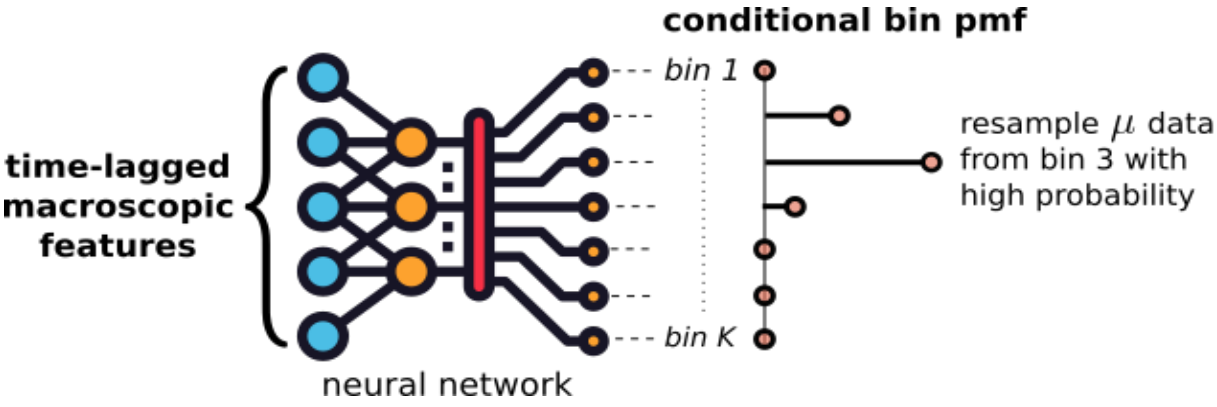


Figure 2: Schematic representation of the neural network used for resampling-based stochastic surrogate modeling, as proposed in [7].

2.2.1.3 Reduced surrogates

We reiterate that our goal is to create a surrogate such that the macroscopic output $\tilde{Q}$ preserves some overall characteristic of Q, see again Figure 1. It is not uncommon, e.g. in a climate context, that the QoI is some *global* (spatially integrated) quantity. We have developed alternative microscopic models $\tilde{\mu}$, which give the same statistics for a given set of global QoIs. The unclosed component of these new microscopic models, which is the only part for which we must learn a surrogate, is reduced in size by several orders of magnitude compared to the original microscopic model μ. Effectively, instead of creating a surrogate for a high-dimensional dynamic field, we only need to create a surrogate for a small number of scalar time series. We denote these as *reduced* surrogates, and the methodology is described in [5,6].

In short form, the microscopic model is given by the following expansion:

$$\tilde{\mu}(x, y, t) = \sum_{i=1}^{d} \tau_i(t) P_i(x, y, t)$$

Here, $\tau_i(t)$ are the generated time series for which we must learn a surrogate, and the $P_i(x, y, t)$ are dynamic fields, which are completely made up of macroscopic variables. Hence, the P_i do not need to be learned from data. In principle we could train any type of surrogate on the τ time series data. However, the plan for the next steps involves training the stochastic surrogates of Section 2.2.1.2 on this reduced training data, resulting in a reduced, stochastic surrogate model.

2.2.1.4 A surrogate based on convolutional neural networks

Building on the semi-intrusive approach from [2], we are developing a (deterministic) surrogate model based on a convolutional neural network (CNN). It is tested on the ISR2D model, a two-dimensional multiscale simulation of the post-stenting healing response of an artery [8,9]. In this model, the component for blood flow simulation is the computationally most expensive. It takes around 80% of the computational time, and the potential gain in performance obtained by replacing it with a surrogate model in the semi-intrusive UQ scenario is therefore highest.

The mapping between input and output of the blood flow simulation can be considered as a function f, which takes the geometry matrix ς and the inlet blood velocity v as input and produces a $2 \times k$ -dimensional vector of wall shear stress (WSS) magnitudes, τ_{wss} as the output:

$$\tau_{wss} = f(\varsigma, v)$$

where $k = 150$ is the grid size along x axis, $\varsigma = (\varsigma_{ij}) \in R^{k \times k}$. The geometry matrix ς was used for the blood flow simulation. The surrogate model $\hat{f}$ replaces the original blood flow model $f(\varsigma, v)$ and offers an approximate prediction of wall shear stress in a reduced amount of time.

The CNN model follows the network structure proposed in [10] and was optimized to fit our application. The model consists of three parts: shape encoding, nonlinear mapping and stress decoding, as shown in Figure 3. More information on these different parts, as well as details on the training of the CNN model and its application to the UQ analysis of the ISR2D model are given in the Appendix (section 4.1).

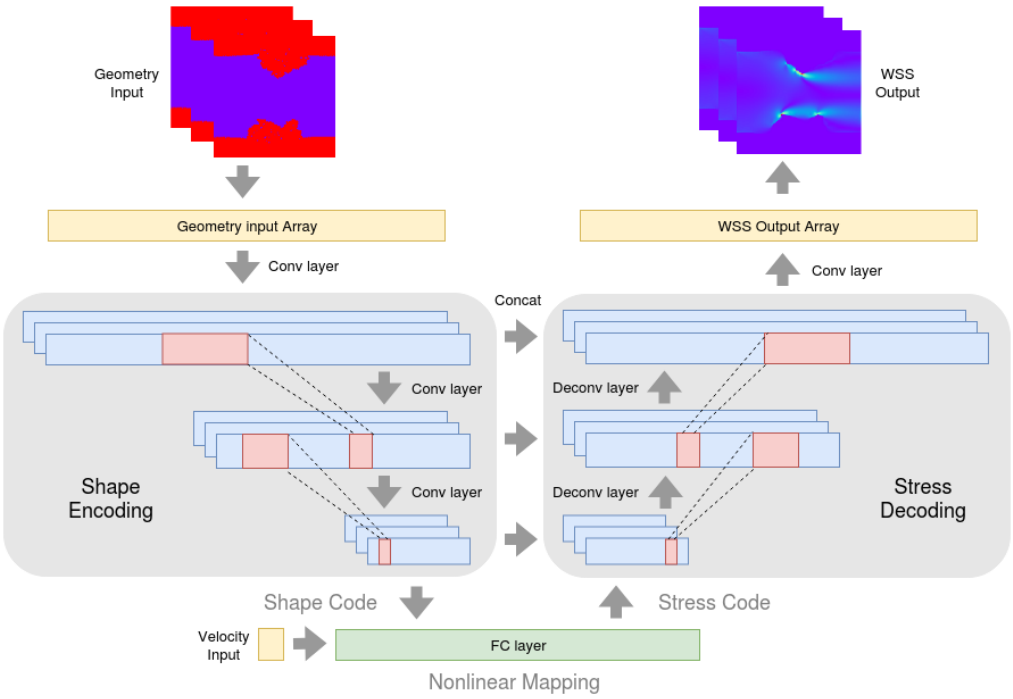


Figure 3: Diagram of the CNN-based surrogate model

2.2.2 Sensitivity analysis based dimension reduction of multiscale models

Here we discuss how sensitivity analysis (SA) of a single scale model is employed in order to reduce the input dimensionality of the related multiscale model, thereby improving the efficiency of its uncertainty estimation. The approach is illustrated with two examples based on a reaction model and the standard Ornstein-Uhlenbeck process. It is important to note that an analysis of the function defining the relation between single scale components is required to understand whether single scale sensitivity analysis can be used to reduce the dimensionality of the overall multiscale model input space. Such analysis for the presented examples was performed in [12].

We propose to evaluate the response sensitivity of the computationally cheap single scale model *f* to estimate an upper bound of the sensitivity of the multiscale model output *z* (Figure 4). This approach can be highly computationally efficient; however, the method is not applicable to all problems. In order to fix uncertain inputs according to the single scale model SA, it should be proved that the total sensitivity for an input x_i remains small also for the output of the model $g(x, x_i)$. This cannot be assumed in general, and it depends on the form of the model function *G*.

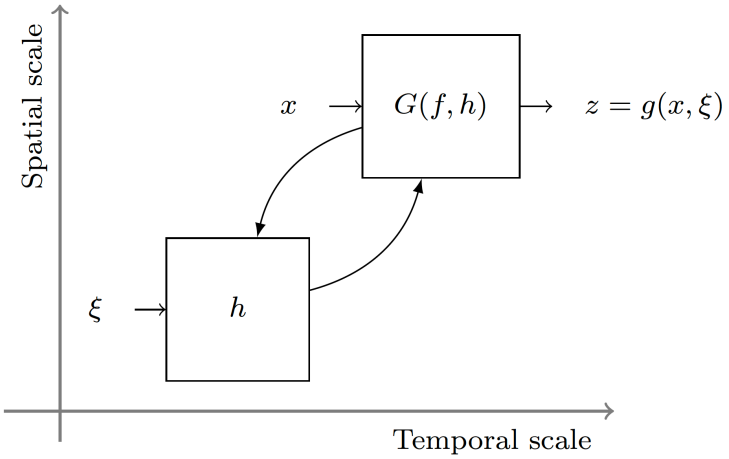


Figure 4: Scale separation map. The functions *G(f,h)* and *h* are the macro and micro models with inputs *x* and *ξ*, respectively. The function *G(f(x), h(ξ))* defines the relation between the response of the micro model and the macro model parameters denoted by *f*. The final multiscale model output *z = g(x, ξ) = G(f(x), h(ξ))* is produced by the macro model.

The first step of the proposed approach is to analyse the multiscale model function *G*. The next step is to estimate numerically the sensitivity indices of the output of the function *f* for all its inputs using a black box method, for instance using the Sobol global method. Then, if it is found that for some inputs the output sensitivity is low, it follows automatically that the sensitivity of the output of the overall multiscale model is low for these inputs. Hence, uncertainty can be estimated with these inputs being fixed without producing a large error in estimation of uncertainty. In the Appendix (section 4.2) we show several examples where the proposed method applies.

2.3 Uncertainty Quantification Patterns

Uncertainty Quantification Patterns (UQPs) are algorithmic specifications of UQ that are generic, extensible and widely applicable. The concept of UQPs was discussed extensively in deliverable D2.1. Our overall goal for the UQPs is to exploit the structure of a multiscale model in order to reduce the computational burden of performing UQ on these models. We identified a number of them in the proposal; see Figure 5. We can build a UQP from a combination of *control structure* UQPs and

optimization UQPs. Broadly speaking, a control structure UQP takes advantage of the way in which the different submodels are connected to each other, e.g. cyclic vs acyclic models. An optimization UQP is a generally applicable method aimed at reducing the cost of the final UQP, for instance through subsampling or surrogate modelling.

UQP1 is the basic UQP, which makes no assumptions on the control structure, and employs no optimization to any of the submodels. In essence, it propagates uncertainty from the inputs to the outputs, in a non-intrusive, black-box fashion. UQP2 is a control structure UQP, applied to acyclic multiscale models. Here, the uncertain output of one submodel becomes the input of the next model, but no information flows back from the latter to the former. Essentially, this is a semi-intrusive method where we apply UQP1 to the *individual* submodels. The Fusion application is testing this approach; see the Appendix (section 4.3) for more details.

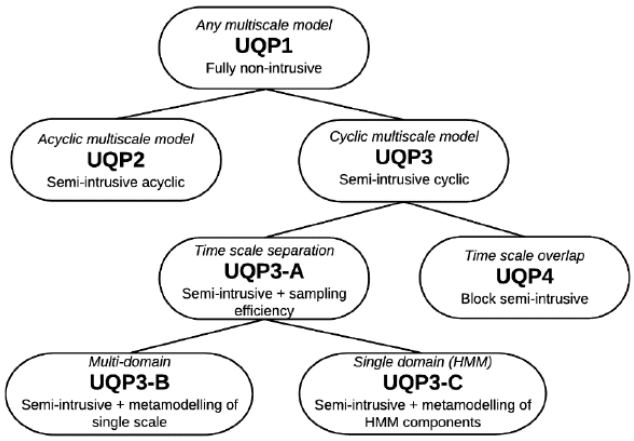


Figure 5: A family of Uncertainty Quantification Patterns (UQPs).

UQPs 3 and 4 deal with multiscale models with cyclic coupling; see also Table 1 in D2.1 (where UQP3 with a single domain was labeled as UQP5). In [2] a combination of control structure and optimization UQPs is used for a cyclic multiscale model. The expensive micromodel is either subsampled (i.e., a reduced number of samples is taken) or replaced by a Gaussian process surrogate. In addition, UQP1 is applied at the level of the macromodel to investigate the effect of the uncertain macroscopic input parameters. This approach is being tested by the In-Stent Restenosis application [13], as well as the Fusion application.

We are developing new optimization UQPs, primarily for models with cyclic coupling, in the form of (application-agnostic) surrogate modelling methodologies presented in Section 2.2.1. As discussed there, by including memory dependence we also target situations where the macro and micro models

have time scale overlap (see also [7] for a more technical discussion of this issue). This corresponds to the category UQP4-B in terms of Table 1 from D2.1.

Due to the modular nature of our framework we can readily combine optimization UQPs with other developed UQPs. For instance, we can apply UQP1 to the macromodel using EasyVVUQ, while the micromodel is replaced with a stochastic surrogate. By building a library of control structure UQPs and optimization UQPs, we aim to enable the use of such combinations for a range of multiscale problems.

2.4 Verification & Validation procedures

With VECMA, we address Verification and Validation (V&V) patterns where verification is the correctness of a solution relative to a given model (computational or mathematical), whereas validation is correctness of that model relative to the real-world system it is intended to describe [15]. Our prime focus is on a representative range of such applications – from fusion and advanced materials through climate and migration, to explore suitable V&V patterns that certify quality in terms of: (i) high fidelity output; (ii) verified at syntactic, semantic and dynamic levels; and (iii) validated models and their coupled integrations, such that they may be relied upon to take important decisions in the domains of concern.

Hoekstra et al. [16] posed an urgent question of V&V for multiscale models, as well as developing formal mathematical approaches appropriate to address the issue of error propagation in, and convergence of, multiscale models. However, there are a large variety of V&V methods due to the range of diversity in data types, subjects and applications. We identify the prominent V&V patterns which are most suitable for multiscale computing applications. We summarize them briefly below; a more extensive description can be found in the Appendix (section 4.4).

- The *Quantity of Interest pattern* focuses on extracting a distribution of QoIs from the simulations. We apply a similarity measure (e.g. Wasserstein distance) to quantify the similarity between the QoI distributions from the simulation, and from the validation data.
- *Code Comparison of QoIs with the Stable Intermediate Forms* is a pattern for monitoring iterative simulation development, evaluating each intermediate step.
- The *Level of Refinement pattern* is a verification pattern focused on asymptotic behaviour of QoIs upon changing certain model parameters, e.g. increasing grid resolution. Another

example is the convergence of Sobol indices when increasing the polynomial order of a (polynomial-based) surrogate model.

- *Ensemble Output Validation* employs a sample testing function to compare output from different model simulations, where each simulation has its own output directory.

2.5 Implementation of UQPs and VVPs in VECMAtk

We have chosen to implement UQPs and VVPs in the VECMA toolkit (VECMAtk, see also [18,19]) in two ways. The first method is that we take the concepts, and enhance our toolkit software (e.g., EasyVVUQ and MUSCLE3) to provide the functionalities required to incorporate specific UQPs/VVPs into application workflows. The second method is that we define an application-agnostic definition of the pattern in Python 3 code (which is used by most components in VECMAtk), and then allow tools and applications to use these generic definitions, allowing users and developers to embed the patterns into applications and specific supporting tools.

As the project progresses, and also in response to the feedback from the VECMA project midterm review, we are attempting to establish these application-agnostic implementations of the patterns within VECMAtk. For the stochastic surrogates, a prototype toolkit is being developed, with the working name “EasySurrogate”. To ensure continuity across the VECMA toolkit, we have opted for a design similar to EasyVVUQ. As in EasyVVUQ, the overall UQP object is called a “Campaign”, and it is broken down in to several generic UQ Elements (UQEs). Thus far, we have identified the following UQEs that make up a surrogate campaign UQP:

- Encoder: formally defined as a component that incorporates application-specific information in a generic definition of a UQ procedure. More specifically, this UQE contains procedures to e.g. load training data or create time-lagged training features required in Eq. (1).
- Method: the specific surrogate method that will be employed. This is the analogue of the “Sampler" object in EasyVVUQ.
- Analysis: several post-processing procedures meant to either verify the surrogate or validate the coupled surrogate – macroscopic model system. Examples include computing the error of the surrogate on the training data, or procedures which compute (statistical) properties of the coupled surrogate – macroscopic model system. The latter falls under VVP1 of Section 2.4.

This is an early-stage design of the toolkit, and it is likely that more UQEs will be identified later on. Finally, note that one major difference between EasySurrogate and EasyVVUQ is the lack of a black box assumption in the former, as the surrogate is meant to replace a part of the multiscale system.

The development of the application-agnostic VVPs mention in Section 2.4 is also underway. These are described in more detail in the Appendix (section 4.4).

2.6 Towards the exascale: scalable UQPs

The scaling of tools in VECMAtk towards future exascale supercomputers has several aspects. An important factor is the software implementation of the algorithms and UQPs and its suitability for e.g. easy generation of large job counts and creation of container jobs. To support this, the QCG Pilot Job Manager is integrated into the VECMA toolkit. As already mentioned in D2.1, a pilot job manager enables users to create a single container job on a remote supercomputer, and subsequently schedule O(1000) jobs or more within that single job container. This creates a pathway to UQ applications that scale towards future exascale machines.

The most straightforward problem involves UQP1 applied to a very expensive computational code, of which an example is given in the next subsection. Many of the implemented EasyVVUQ sampling techniques are subject to the curse of dimensionality, meaning that the required number of code evaluations increases exponentially with the number of uncertain parameters. Notwithstanding this, many of the more sophisticated “hybrid” UQPs contain a UQP1 component, and are as such candidates for a large ensemble run within a pilot job manager. For instance, UQP2 applies UQP1 to individual single-scale models, and by combining UQP4-B with UQP1 on the macro model we obtain a hybrid UQP with uncertainty on the macroscopic parameters. Another way in which a highly expensive UQP1 could tie into its more advanced counterparts is by generating a diverse database of high-resolution reference data, on which we can train or validate the various surrogate modelling techniques.

Example: Bio-medicine

Both speed and reliability are essential in order to accelerate the drug discovery process, as well as to offer real-time clinical decision support in personalised drug selection. To accelerate the calculations for a large number of drug compounds and/or protein variants, we have been using task farming for concurrent execution of multiple tasks on high performance computers. UQ, which in this application has been available for a considerable period of time in production mode, is handled by a stripped down version of EasyVVUQ; indeed, the architecture of EasyVVUQ is built on this particular application. The

job farming function on SuperMUC-NG, for example, enables running a number of independent parallel jobs with a single batch job. It is well aligned with the need to perform ensemble-based molecular dynamics studies. The job farming method is based on the light-weight data base server redis, which offers several desirable features like flexibility and extensibility. It is one of the most efficient ways to run multiple simulations in parallel. The job farming submits multiple jobs to the batch system like a single large job and is thus easier to schedule. This technique is particularly useful in our studies, as we need to perform ensembles of independent MD simulations simultaneously. In addition, we also use RADICAL-Cybertools [17] for flexible task-level parallelism; QCG-PJM is a now emerging alternative. This helps reduce the burden of coordination of the many tasks by providing well defined execution patterns, and is utilized by user-facing domain specific workflows such as the high-throughput binding affinity calculator. In the elapsed time needed to perform one single simulation, we can run all the replicas within an ensemble. On SuperMUC-NG, the maximum number of nodes for a single submission is 6044 (the full production partition comprising 311,000 cores), which enables us to produce binding affinity predictions for of the order of 1200 compounds within a day or so. The ability to perform such calculations fast and reliably is now having an important impact on the use of such methods in areas such as drug discovery and clinical decision making.

3 Conclusions

The development of advanced multiscale VVUQ algorithms in VECMA is progressing well. For the advanced UQPs (UQP2-5) we have developed multiple new methods for surrogate modelling and sensitivity analysis, as reported in section 2.2. They are being implemented in the “EasySurrogate” toolkit and as such become part of the overall toolkit VECMAtk. The pilot-job manager integrated into VECMAtk contributes to scaling of the toolkit towards the exascale. Besides new advanced UQ functionality, formal procedures for V&V are developed and added to VECMAtk. In the final, third year of VECMA we will consolidate these newly developed algorithms, streamline their software implementation and integrate them further into the VECMA tookit.

4 Appendices

4.1 CNN surrogate model and its application to UQ analysis of the ISR2D model

4.1.1 Technical details of the CNN surrogate model

As discussed in section 2.2.1.4, the CNN model consists of three parts, for shape encoding, nonlinear mapping and stress decoding. The shape encoding layers extract the features of the geometry to the shape code. A fully connected (FC) layer then maps the shape code together with the blood flow velocity to the stress code. The stress decoding part is responsible for a mapping from the stress code to wall shear stress. In this surrogate model, the geometry input was transformed from a binary map to a $2 \times k$ array which indicates the locations of upper and lower fluid-solid boundaries. The convolution layers then take the information from both boundaries into account and predict the shear stress on these boundaries. There are three convolution layers, a fully connected layer and three deconvolution layers deployed between the input layer and output layer. Each of them is followed by a rectifier linear unit (ReLU) as the activation function. Besides, the output of each convolution layer is concatenated to the corresponding deconvolution layer to help with the decoding process.

The training data for the surrogate model comes from the runs of the ISR2D model. One run of ISR2D calls the Lattice Boltzmann solver of the blood flow model component 1440 times (once per hour of simulated time). This means that with only a few runs of the simulation, a considerable amount of flow data for training is already available. We trained the surrogate model with the data from four runs of ISR2D simulation, hence 5760 blood vessel geometries and wall shear stress distributions were used for training. The training optimization was based on the mean squared error loss function:

$$L(\tau_w ss) = 1/n \sum_{i=1}^{n} \left|\left| \tau_{wss}^{(i)} - \hat{\tau}_{wss}^{(i)} \right|\right|^2$$

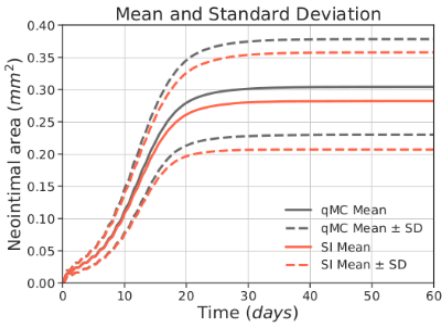
where n denotes the number of samples of the training set. The *Adam* optimizer was used to optimize the hyper-parameters in the model. A validation against a test dataset was done during the training process to prevent the model from overfitting. The epoch was set to 80 as the loss does not decrease significantly after that. The surrogate model was implemented in Keras [11].

4.1.2 Application to UQ analysis of the ISR2D model

We applied the CNN surrogate model to a two-dimensional model of in-stent restenosis. This model simulates a repeated growth of cells inside a coronary artery due to the healing process from installing a stent to treat the initial stenosis. For a more detailed description of the application, we refer the reader to [8,9]. In Figure 6, we show results of the UQ analysis obtained by applying the CNN surrogate

model, as well as its corresponding speedup compared to the quasi Monte Carlo (qMC) method and to previous semi-intrusive UQ methods. The UQ estimates with the CNN surrogate model are very close to the qMC results and outperformed the results of most previous surrogate models except Data Driven (DD) I. The results with DD I are still slightly better which may be due to the large training dataset it used [9]. However the speedup obtained with DD I is much lower than with the CNN model, since the CNN model learns the latent pattern of the data, while DD I simply looks for similar cases among all the training data. Because of this, the prediction cost of CNN is significantly lower.

(a)



(b)

UQ Method	Micro Model	Mean Estimation x 10^{-1} (mm^2)		Standard Deviation x 10^{-2} (mm^2)		Restenosis Ratio (%)	
		Value	Error	Value	Error	Value	Error
qMC	LBM[1]	3.04	0	7.41	0	12.2	0
SI	DD I[2]	2.88	0.16	7.43	0.02	8.3	3.9
SI	DD II[2]	2.79	0.25	7.51	0.10	6.3	5.9
SI	Phys[2]	2.26	0.78	7.98	0.57	1.1	11.1
SI	CNN	2.82	0.22	7.54	0.13	7.7	4.5

(c)

UQ Method	Micro Model	$\mathcal{T}_{ISR}$ (min)	$\mathcal{T}_{ISR^*}$ (min)	$\mathcal{T}_{micro}$ (min)	$\mathcal{T}_{train}$ (min)	$\mathcal{T}_{sample}$ (min)	N	Speedup of UQ
qMC	LBM[1]	89.4	/	74.9	/	/	1024	1
SI	DD I[2]	/	50.9	37.8	/	894	1024	1.72
SI	DD II[2]	/	14.6	2.05	/	447	1024	5.94
SI	Phys[2]	/	11.9	0.08	/	/	1024	7.51
SI	CNN	/	12.8	0.26	9.9	357.6	1024	6.79

Figure 6: (a) Mean and standard deviation of the ISR2D model output on the neointimal area with quasi-Monte Carlo (qMC) and with the semi-intrusive (SI) method. (b) Comparison of the estimates of means and standard deviations of neointimal growth and restenosis ratio with qMC and SI. (c): Speedup of SI compared to qMC. The details of the DD I, DD II and Phys surrogate models for SI can be found in [9].

4.2 Two examples of SA-based dimension reduction

4.2.1 Example 1: Reaction equation

The first example is a reaction equation presented by an acyclic model with initial conditions provided by some function f(x) (see [12] for further details):

$$\frac{\partial z(t,x,\xi)}{\partial t} = -\psi(\xi) z(t,x,\xi),$$
$$z(0,x,\xi) = f(x),$$

where x and x_i are uncertain model inputs. The sensitivity analysis of f yields:

$$S^f_{T_{x_1}} \approx 2.9 \cdot 10^{-1},$$
$$S^f_{T_{x_2}} \approx 7.2 \cdot 10^{-2},$$
$$S^f_{T_{x_3}} \approx 6.5 \cdot 10^{-1},$$

suggesting that the parameter x_2 does not significantly affect the output of the function f. Therefore, the value of this parameter can be equated to its mean when estimating uncertainty of the overall model response z.

Figure 7 (a) illustrates a satisfactory match between the mean values and standard deviations obtained by sampling the results varying all the uncertain inputs and keeping the input x_2 equal to its mean

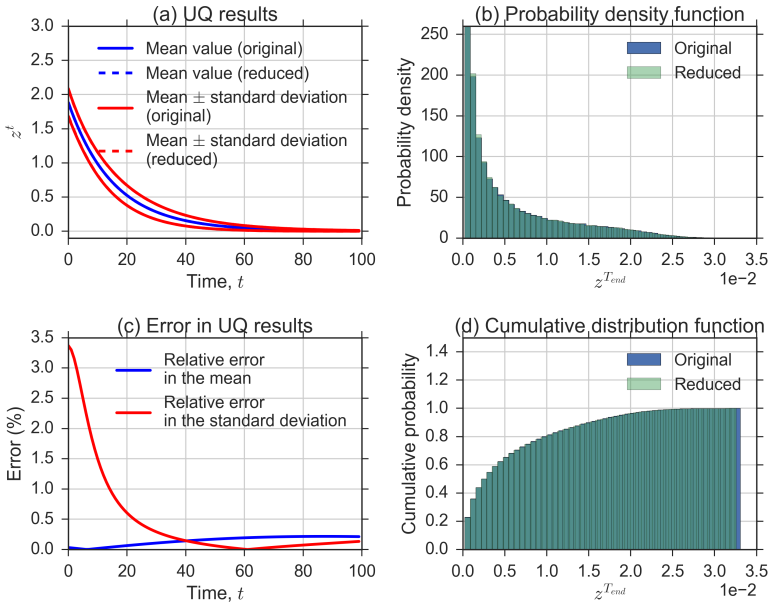


Figure 7: (a) Comparison of the estimated mean and standard deviation of the model response *z(t)* using the original sample and the sample with the unimportant parameter x_2 equal to its mean value (reduced); (b) and (d) Comparison of the probability density functions and the cumulative distribution functions at the final simulation time T_{end} =100; (c) Relative error in the estimated mean and standard deviation using the samples with the reduced number of uncertain input.

value. Figure 7 (c) shows that the relative error in the standard deviation does not exceed 3.5% at any simulation time. Moreover, the resulting p-value of Levene's test is about 0.84. Therefore, the null hypothesis that the samples are obtained from distributions with equal variances cannot be rejected. Figure 7 (b) and (d) show the probability density functions (PDFs) and the cumulative distribution functions (CDFs) of the uncertain model output *z* at the final simulation time obtained using these two samples. There is a good match in the PDFs and CDFs with Kolmogorov–Smirnov (K-S) two-sample test shows the K-S distance of $3.6 \cdot 10^{-4}$ and p-value larger than 0.5, therefore, the hypothesis that the two samples are drawn from the same distributions cannot be rejected.

4.2.2 Example 2: Standard Ornstein-Uhlenbeck process

The second example, also from [12], is an a-cyclic multiscale model whose micro scale dynamics do not depend on the macro scale response. Let us consider the following system (Figure 8 (a)):

$$\begin{aligned}\frac{\partial z}{\partial t} &= v + f(x),\\ \frac{\partial v}{\partial t} &= -\frac{1}{\epsilon}v + \frac{1}{\sqrt{\epsilon}}\dot{W}_t,\\ f(x) &= -x_1 + (x_2^2\, x_3 + x_4),\end{aligned}$$

where *z* simulates the slow processes with *z(t=0) = 1*, *v* is the fast process with *v(t=0)=1*, ϵ=10^{-2}, $\dot{W}_t$ is a white noise with unit variance. The fast dynamics is the standard Ornstein-Uhlenbeck process.

Sensitivity analysis of the function *f(x)* yields

$$\begin{aligned}S^f_{T_{x_1}} &\approx 7.7 \cdot 10^{-1},\\ S^f_{T_{x_2}} &\approx 2.6 \cdot 10^{-4},\\ S^f_{T_{x_3}} &\approx 3.9 \cdot 10^{-4},\\ S^f_{T_{x_4}} &\approx 2.0 \cdot 10^{-1}.\end{aligned}$$

At any simulation time, the inputs x_2 and x_3 do not influence significantly the output of the function *f*. Therefore, they can be equated to their mean values without a substantial loss of accuracy of the uncertainty estimate.

The uncertainty estimation results of *z* are presented in Figure 8(b). As it is proven analytically, the estimates obtained by sampling the model results with uncertain parameters x_2 and x_3 equal to their mean values are close to those resulting from samples where all the uncertain inputs vary. At any simulation time, the relative error between these estimates of the standard deviation does not exceed 1.1% (Figure 8 (d)). Additionally, Levene's test shows a p-value of 0.66, therefore, we cannot reject the hypothesis that the two samples are drawn from distributions with the same variance.

The PDFs and CDFs for the model result at the final point in time (T_{end}) obtained from these two samples are shown in Figure 8 (c) and (e). There is a good match of the PDFs and CDFs obtained from these two samples, and K-S test produces the distance of 0.01 and p-value of 0.47, therefore, the hypothesis that the two samples are drawn from the same distributions cannot be rejected.

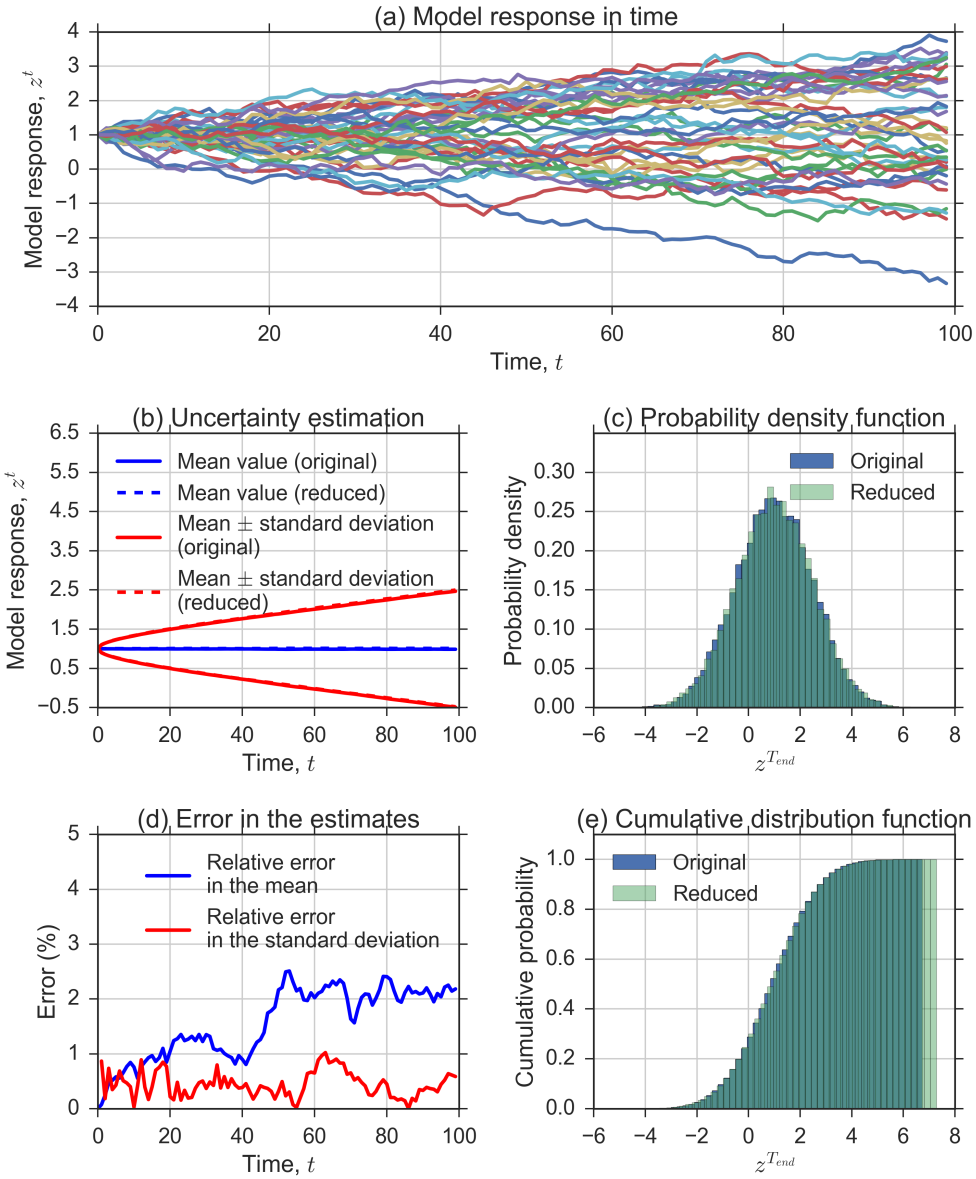


Figure 8: (a) Standard Ornstein-Uhlenbeck process; (b) Comparison of UQ result using the original sample and the sample obtained with values of the unimportant parameters x_2 and x_3 equal to their mean (reduced); (c) and (e) Comparison of the PDF and CDF at the final time step; (d) Relative error in the estimation of the mean and standard deviation

4.3 UQP2 for the fusion application

For the fusion application, the pattern UQP2 is being tested (as mentioned in section 2.3). Currently, the workflow for this application combines three main models, as shown in **Error! Reference source not found.**: an equilibrium code that describes the plasma geometry, a turbulence code that models the heat fluxes (which are then converted into transport coefficient), and a transport solver that evolves temperature profiles at the macro time scale.

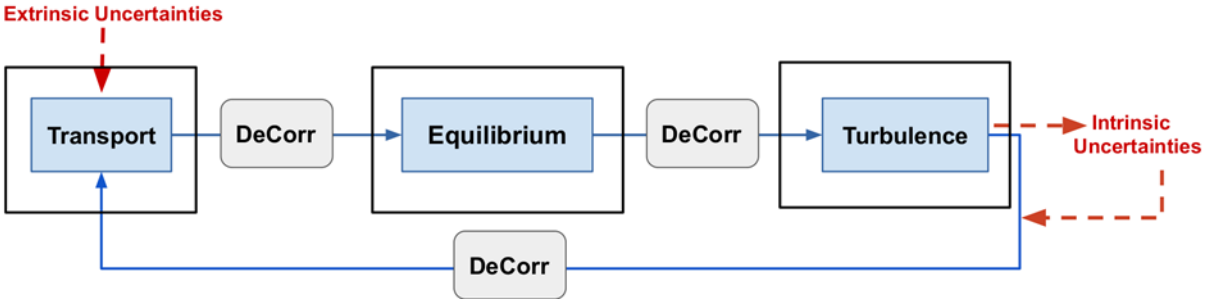


Figure 9: The UQP2 pattern applied to the fusion model workflow.

The *DeCorr* boxes act as a bridge between two UQP1 boxes. At this stage we can parametrize the UQP1 outputs using for example splines, and then we transform correlated random variables of non-normal distribution using the Rosenblatt transformation. For example, let us treat the boundary conditions defining the electron temperature Te at the plasma edge as the uncertain parameters. As Figure 10 (A) shows, the *Transport* code output Te contains uncertainties at the edge. Before using it for the next box (the *Equilibrium* code), we approximate the Te mean by a cubic spline and we only use the two last control points (CP1 and CP2) of the approximated curve instead of all Te points. Figure 10 (B) shows the output distribution at those control points. Finally, we run the second code using less parameters. Figure 10 (C) shows the uncertainties in the pressure quantity, an output of the *Equilibrium* code.

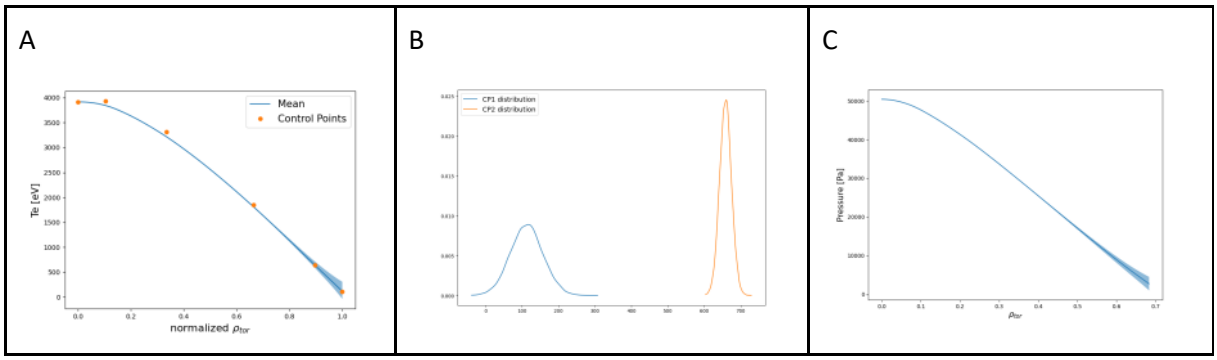


Figure 10: The Electron temperature profile Te, splines control points (A) and pressure profile (C) are shown with respect to the normalized toroidal flux coordinate ρ_{tor}. The probability densities in the two last control points are plotted in (B).

4.4 V&V patterns

4.4.1 Pattern: Quantity of Interest

Quantity of Interest (QoI) focuses on extracting a distribution of QoIs from the simulations runs (using the Decoder and Analysis). We apply the similarity measure to quantify the similarity between the QoI distributions from the simulation, and from the validation data. The main input parameters are:

- Decoder, to translate raw simulation output to computable data (see EasyVVUQ);
- Analysis, to extract the QoI distributions from the computable data (see EasyVVUQ);
- Mathematical definition of the similarity measure (e.g., Wasserstein metric, Hellinger distance, Kullback-Leibler divergence or distance between means).

The pseudocode below describes the QoI pattern as follows:

```
def extract_and_compare_qoi_dist(decode, decode_validation_data, analyse, similarity_measure,
validation_qoi_source, output_dir):
    qoi_distribution_validation = decode_validation_data(validation_qoi_source)
    output_data = decode(output_dir)
    qoi_distribution_simulation = analyse(output_data)
    return similarity_measure(qoi_distribution_simulation, qoi_distribution_validation)
```

In turn, we obtain a notion of the distance between the distribution, for instance a single number per time point, or a combination of distance between means and entropy.

4.4.2 Pattern: Stable Intermediate Forms

Stable Intermediate Forms (SIF) is a general, gradual, observable and risk-averse pattern for making progress in simulation development iteratively and limiting the effects of failure. When undertaking a series of changes to move from one step to another, each intermediate step is evaluated and once it is considered stable it will be used as the basis for further changes. This helps in gaining confidence in the iterative and incremental model development rather than carrying out sudden changes with a single large stride, making it too complex and possibly deviating from the development pipeline.

4.4.3 Pattern: Level of Refinement

Level of Refinement (LoR) is a general verification pattern that seeks asymptotic behaviour in QoI upon increasing the resolution of certain model parameters. It is important to note that the *same* quantity of interest is computed at every given resolution. One might for instance perform a grid refinement study in Computational Fluids Dynamics, where the same output quantity is computed on grids of

increasing spatial resolution. The point from which the simulation outcomes no longer change significantly marks the largest spatial resolution for which the results are independent of the computational grid. Another example concerns the convergence of Sobol sensitivity indices (demonstrated below), computed by replacing the expensive simulation code with a cheap polynomial-based surrogate model. Here, the polynomial order of the surrogate should be high enough to ensure converged sensitivity estimates.

We require the following inputs for the LoR pattern:

- results_dirs: list of result dirs, one directory for each resolution and each one containing the same QoIs stored to disk.
- sample_load_function: a function which loads the QoIs from each subdirectory of the results_dirs.
- aggregation_function: function to combine all results.
- **kwargs: custom parameters. The 'items' parameter must be used to give explicit ordering of the various subdirectories, indicating the order of the refinement.

In turn, the pattern returns the aggregated validation outcome, which is inspected for convergence with the level of refinement. We demonstrate the LoR VVP by computing the first-order Sobol indices of the so-called Sobol g-function in two dimensions, which is given by:

$$g(x_1, x_2) = \prod_{i=1,2} \frac{|4x_i - 2| + a_i}{1 + a_i}$$

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

```
from vvp import ensemble_vvp
ensemble_vvp('/tmp/sobols', load_sobols, check_convergence, items=items,
        poly_orders=poly_orders)
```

Here, `load sobols' just loads the Sobol indices from a .CSV file in each directory, and `check_convergence` receives a list with all Sobol indices and outputs the following:

```
Polynomial order = 2
Sobol x1 = 0.722 , exact = 0.750 , error = 0.028
Sobol x2 = 0.154 , exact = 0.188 , error = 0.033
==========================================================
```

```
Polynomial order = 3
Sobol x1 = 0.758 , exact = 0.750 , error = 0.008
Sobol x2 = 0.197 , exact = 0.188 , error = 0.010
========================================================
Polynomial order = 4
Sobol x1 = 0.739 , exact = 0.750 , error = 0.011
Sobol x2 = 0.175 , exact = 0.188 , error = 0.013
========================================================
Polynomial order = 5
Sobol x1 = 0.754 , exact = 0.750 , error = 0.004
Sobol x2 = 0.192 , exact = 0.188 , error = 0.005
========================================================
Polynomial order = 6
Sobol x1 = 0.744 , exact = 0.750 , error = 0.006
Sobol x2 = 0.181 , exact = 0.188 , error = 0.007
========================================================
Polynomial order = 7
Sobol x1 = 0.752 , exact = 0.750 , error = 0.002
Sobol x2 = 0.190 , exact = 0.188 , error = 0.003
========================================================
Polynomial order = 8
Sobol x1 = 0.747 , exact = 0.750 , error = 0.003
Sobol x2 = 0.183 , exact = 0.188 , error = 0.004
========================================================
Polynomial order = 9
Sobol x1 = 0.751 , exact = 0.750 , error = 0.001
Sobol x2 = 0.189 , exact = 0.188 , error = 0.002
========================================================
```

In this particular example there is also a validation element, since we can analytically compute the exact first-order Sobol indices from the g-function (0.75 and 0.188). However, even without this information it would be clear that from a polynomial order of 5 and onward the results do not change much

4.4.4 Pattern: Ensemble Output Validation

Ensemble Output Validation (EOV) will operate a sample testing function on each directory, and print the outputs to screen, and use an aggregation function to combine all outputs into a compound metric. There is no explicit argument to indicate where the validation data resides (currently assumed to be packaged with the simulation output and known by the function). The validation function should return a set of metrics using the following code:

```
def ensemble_vvp(results_dirs, sample_testing_function, aggregation_function, **kwargs):
    #if a single result_dir is specified, still add it to a list
    if type(results_dirs) == str:
        tmp = []; tmp.append(results_dirs); results_dirs = tmp

    for results_dir in results_dirs:


        scores = []

        #use user-specified sample directories if specified,
        #otherwise look for uq results in all directories in results_dir
        if 'items' in kwargs:
            items = kwargs['items']
        else:
            items = os.listdir("{}".format(results_dir))

        for item in items:
            if os.path.isdir(os.path.join(results_dir, item)):
                print(os.path.join(results_dir, item))
                scores.append(sample_testing_function(os.path.join(results_dir,item), **kwargs))

        aggregation_function(scores, **kwargs)
```

The code is available at https://github.com/djgroen/FabSim3/blob/master/VVP/vvp.py

5 References

[1] R.C. Smith, *Uncertainty quantification: theory, implementation, and applications*, SIAM, 2013.

[2] A. Nikishova, A. G. Hoekstra, *Semi-intrusive Uncertainty Propagation for multiscale models*, Journal of Computational Science, 2019. DOI:10.1016/j.jocs.2019.06.007

[3] N. Verheul, D. Crommelin, *Data-driven stochastic representations of unresolved features in multiscale models*, Communications in Mathematical Sciences, Vol. 14 (2016), pp 1213 – 1236. https://dx.doi.org/10.4310/CMS.2016.v14.n5.a2

[4] W. Edeling, D. Crommelin, *Towards data-driven dynamics surrogate models for ocean flow*. In: Proceedings of the Platform for Advanced Scientific Computing Conference 2019 (PASC '19), Zurich, Switzerland. ACM, New York, NY, USA. https://doi.org/10.1145/3324989.3325713

[5] W. Edeling, D. Crommelin, *Reduced model-error source terms for fluid flow*, Proceedings of UNCECOMP 2019 conference. DOI:10.13140/RG.2.2.36773.35048

[6] W. Edeling, D. Crommelin, *Reducing data-driven dynamical subgrid scale models by physical constraints*, Computers and Fluids, 2020. https://doi.org/10.1016/j.compfluid.2020.104470

[7] D. Crommelin, W. Edeling, *Resampling with neural networks for stochastic parameterization in multiscale systems*, 2020. arXiv preprint arXiv:2004.01457

[8] H. Tahir, A. G. Hoekstra, E. Lorenz, P. V. Lawford, D. R. Hose, J. Gunn, 410 D. J.W. Evans, *Multi-scale simulations of the dynamics of in-stent restenosis: impact of stent deployment and design*, Interface Focus 1 (3) (2011). https://doi.org/10.1098/rsfs.2010.0024

[9] H. Tahir, C. Bona-Casas, A. G. Hoekstra, *Modelling the effect of a functional endothelium on the development of in-stent restenosis*, PLoS One 8(6) (2013) e66138. DOI: 10.1371/journal.pone.0066138

[10] X. Guo, W. Li, F. Iorio, *Convolutional neural networks for steady flow approximation*, in: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, New York, NY, USA, 2016, pp. 481-490. https://doi.org/10.1145/2939672.2939738

[11] F. Chollet et al., *KERAS*, 2015, https://keras.io

[12] A. Nikishova, G.E. Comi, A.G. Hoekstra, *Sensitivity analysis based dimension reduction of multiscale models*. Mathematics and Computers in Simulation 170 (2020), 205-220. https://doi.org/10.1016/j.matcom.2019.10.013

[13] A. Nikishova, L. Veen, P. Zun, A.G. Hoekstra, *Uncertainty quantification of a multiscale model for in-stent restenosis*, Cardiovascular engineering and technology, 9, 4 (2018), 761—774. doi.org/10.1007/s13239-018-00372-4

[14] Ambrogioni, L., Güçlü, U., van Gerven, M. A., & Maris, E. *The kernel mixture network: A nonparametric method for conditional density estimation of continuous random variables*, 2017. arXiv preprint arXiv:1705.07111.

[15] J. Banks, *Handbook of simulation: principles, methodology, advances, applications, and practice*. John Wiley & Sons. (1998)

[16] A. Hoekstra, B. Chopard and P. Coveney, *Multiscale modelling and simulation: A position paper*. Phil. Trans. R. Soc., A: Mathematical, Physical and Engineering Sciences, 372: 20130377. (2014) https://doi.org/10.1098/rsta.2013.0377

[17] V. Balasubramanian, S. Jha, A. Merzky, M. Turilli, *RADICAL-Cybertools: Middleware Building Blocks for Scalable Science*, 2019. arXiv:1904.03085

[18] D. Groen, R. A. Richardson, D. W. Wright, V. Jancauskas, R. Sinclair, P. Karlshoefer, M. Vassaux, H. Arabnejad, T. Piontek, P. Kopta, B. Bosak, J. Lakhlili, O. Hoenen, D. Suleimenova, W. Edeling, D. Crommelin, A. Nikishova, P. V. Coveney, *Introducing VECMAtk – verification, validation and uncertainty quantification for multiscale and HPC simulations*, Lecture Notes in Computer Science, 1153, 479-492 (2019), DOI:10.1007/978-3-030-22747-0_36

[19] VECMA toolkit website, https://www.vecma-toolkit.eu/