VECMAtk: a scalable verification, validation and uncertainty quantification toolkit for scientific simulations

Groen, D.; Arabnejad, H.; Jancauskas, V.; Edeling, W.N.; Jansson, F. ; Richardson, R.A.; Lakhlili, J.; Veen, L. ; Bosak, B.; Kopta, P.; Wright, D.W.; Monnier, N.; Karlshoefer, P.; Suleimenova, D.; Sinclair, R.; Vassaux, M.; Nikishova, A.; Bieniek, M.; Luk, O.O.; Kulczewski, M.; Raffin, E.; Crommelin, D.; Hoenen, O.; Coster, D.P.; Coveney, P.V.

DOI
10.1098/rsta.2020.0221

Publication date
2021

Document Version
Final published version

Published in
Philosophical Transactions of the Royal Society A - Mathematical, Physical and Engineering Sciences

PHILOSOPHICAL TRANSACTIONS A

royalsocietypublishing.org/journal/rsta

Research

Cite this article: Groen D *et al*. 2021 VECMAtk: a scalable verification, validation and uncertainty quantification toolkit for scientific simulations. *Phil. Trans. R. Soc. A* **379**: 20200221.
https://doi.org/10.1098/rsta.2020.0221

Accepted: 10 November 2020

One contribution of 15 to a theme issue 'Reliability and reproducibility in computational science: implementing verification, validation and uncertainty quantification *in silico*'.

Subject Areas:
computer modelling and simulation, software, statistics

Keywords:
multiscale simulations, verification, validation, uncertainty quantification

Author for correspondence:
D. Groen
e-mail: derek.groen@brunel.ac.uk

VECMAtk: a scalable verification, validation and uncertainty quantification toolkit for scientific simulations

D. Groen[1,2], H. Arabnejad[1], V. Jancauskas[3], W. N. Edeling[4], F. Jansson[4,11], R. A. Richardson[2,5], J. Lakhlili[6], L. Veen[5], B. Bosak[7], P. Kopta[7], D. W. Wright[2], N. Monnier[8], P. Karlshoefer[8], D. Suleimenova[1], R. Sinclair[2], M. Vassaux[2], A. Nikishova[9], M. Bieniek[2], Onnie O. Luk[6], M. Kulczewski[7], E. Raffin[8], D. Crommelin[4,10], O. Hoenen[6], D. P. Coster[6], T. Piontek[7] and P. V. Coveney[2,9]

[1]Department of Computer Science, Brunel University London, London, UK
[2]Centre for Computational Science, University College London, London, UK
[3]Leibniz Supercomputing Centre, Garching, Germany
[4]Centrum Wiskunde and Informatica, Amsterdam, The Netherlands
[5]Netherlands eScience Center, Amsterdam, The Netherlands
[6]Max Planck Institute for Plasma Physics - Garching, Munich, Germany
[7]Poznań Supercomputing and Networking Center, Poznań, Poland
[8]CEPP - Center for Excellence in Performance Programming, Atos Bull, Rennes, France
[9]Computational Science Lab, Institute for Informatics, University of Amsterdam, Amsterdam, The Netherlands

THE ROYAL SOCIETY PUBLISHING

[10]Korteweg-de Vries Institute for Mathematics, Amsterdam, The Netherlands
[11]Department of Geoscience and Remote Sensing, Delft University of Technology, Delft, The Netherlands

DG, 0000-0001-7463-3765; HA, 0000-0002-0789-1825; VJ, 0000-0002-6051-210X; WE, 0000-0003-4734-7960; DS, 0000-0003-4474-0943; AN, 0000-0003-4813-9282; MKB, 0000-0002-3065-5417; OL, 0000-0003-0560-4797; ER, 0000-0003-0359-5372; PC, 0000-0002-8787-7256

We present the VECMA toolkit (VECMAtk), a flexible software environment for single and multiscale simulations that introduces directly applicable and reusable procedures for verification, validation (V&V), sensitivity analysis (SA) and uncertainty quantication (UQ). It enables users to verify key aspects of their applications, systematically compare and validate the simulation outputs against observational or benchmark data, and run simulations conveniently on any platform from the desktop to current multi-petascale computers. In this sequel to our paper on VECMAtk which we presented last year [1] we focus on a range of functional and performance improvements that we have introduced, cover newly introduced components, and applications examples from seven different domains such as conflict modelling and environmental sciences. We also present several implemented patterns for UQ/SA and V&V, and guide the reader through one example concerning COVID-19 modelling in detail.

This article is part of the theme issue 'Reliability and reproducibility in computational science: implementing verification, validation and uncertainty quantification *in silico*'.

1. Introduction

Computational models play an ever-growing role in predicting the behaviour of real-world systems or physical phenomena [2], using equations and/or heuristics to encode the natural laws and theories of the world we inhabit. Because models are a simplified representation of real-world systems, they can behave differently for a variety of reasons. Key model inputs, such as initial conditions, boundary conditions and important parameters controlling the model, are often not known with certainty or are inadequately described [3]. For example, in the case of human migration simulations [4], the model may require knowledge of a wide range of input parameters, such as details of transport modes, roads, settlement populations and conflict zones, before we can execute the simulation. However, often these parameters are not precisely known or cannot be obtained with high accuracy.

Another source of discrepancy between the model and reality are the assumptions and simplifications that are made as part of creating the conceptual model. Simplifications can reduce the computational cost of models, but also make them less accurate. And assumptions are by definition uncertain, which makes it necessary to test the model accuracy when these assumptions are adjusted to match realistic alternative scenarios.

The appropriate level of accuracy and reliability in the results can be obtained by ensuring *not only* that the computational model accurately represents the underlying mathematical model and its solution (*Verification*) [5,6], but the degree to which a model is an accurate representation of the real world based on comparisons between computational results and experimental data (i.e. the deviation of the model from reality) (*Validation*) [7], and how variations in the numerical and physical parameters affect simulation outcomes (*Uncertainty Quantification*). Collectively, the processes involved in evaluating our level of trust in the results obtained from models are known as VVUQ. VVUQ processes provide the basis for determining our level of trust in any given model and the results obtained using it [2,8,9].

Many scientific modelling challenges involve complex and possibly multiscale, multiphysics phenomena, the results of which are unavoidably approximate. VVUQ then becomes essential

to determine whether the results can be trusted, and whether decision makers can rely on them to guide subsequent actions, making our simulations *actionable*. Indeed, the impact of scientific computing relies directly on its trustworthiness [6], and VVUQ provides the basis for determining our level of trust in any given model and the results obtained using it [2].

Within this paper, we describe the latest release of the VECMA toolkit. VECMA (www.vecma.eu) is an EU-funded initiative that focuses on enabling VVUQ for large-scale and complex simulations. The toolkit we present here facilitate the use of VVUQ techniques in (multiscale) applications, as well as a range of Verification and Validation Patterns (VVPs) to enable a systematic comparison of simulation results against a range of validation targets, such as benchmarks or measurements. We review a range of related research and development efforts in §2, and then provide a brief description of the toolkit as a whole in §3. We describe the key aspects of each component in the toolkit in §4 and present a range of exemplar applications in §5, while we discuss on the main performance and scalability aspects of the toolkit in §6. Lastly, we share our main conclusions in §7.

2. Related work

Several other toolkits share a subset of the added values that VECMAtk provides. In the area of VVUQ, a well-known toolkit is Design Analysis for Optimization and Terascale Applications (DAKOTA)[1] [10], which provides a suite of algorithms for optimization, UQ, parameter studies, and model calibration. DAKOTA is a powerful tool, but has a relatively steep learning curve due to the large number of tools available [11] and offers no way to coordinate resources across concurrent runs [12,13]. Similarly, there are other toolkits that help with UQ directly, such as UQTK [14] and UQLab.[2] One particularly efficient method to handle UQ for a range of applications is the multi-level Monte Carlo Method [15].

In the area of VVUQ using HPC, there are several other relevant tools. OpenTURNS [16] focuses on probabilistic modelling and uncertainty management, connects to HPC facilities, and provides calibration/Bayesian methods and a full set of interface to optimization solvers. Uranie leverages the ROOT [3] framework to support a wide range of uncertainty quantification (UQ) and sensitivity analyses (SA) activities using local and HPC resources. A key requirement for performing many types of UQ and SA is the ability to effectively run large ensembles of simulations runs. In addition to QCG-PJ, presented in this paper, there are tools such as RADICAL-Cybertools [17] that can be used to initiate and manage large simulation ensembles on peta and exascale supercomputers. In the area of surrogate modelling, GPM/SA [18,19] helps to create surrogate models, calibrate them to observations of the system and give predictions of the expected system response. There is also a portfolio of available solutions for rapidly processing user-defined experiments consisting of large numbers of relatively small tasks. The examples are Swift/T [20] and Parsl [21], both of which support execution of data-driven workflows.

Another range of relevant related tools include more statistically oriented approaches. For instance, Uncertainpy [22] is a UQ and SA library that supports quasi-Monte Carlo (QMC) and polynomial chaos expansions (PCE) methods. PSUADE [23] is a toolbox for UQ, SA and model calibration in non-intrusive ways [24], while DUE [25] assesses uncertain environmental variables, and generates realizations of uncertain data for use in uncertainty propagation analyses. PyMC3 [26] is a Python package for Bayesian statistical modelling and probabilistic machine learning which focuses on Markov Chain Monte Carlo approaches and variational fitting. Similarly, SimLab[4] offers global UQ-SA based on non-intrusive Monte Carlo methods.

[1]See https://dakota.sandia.gov.

[2]See https://www.uqlab.com.

[3]See http://root.cern.ch.

[4]See https://ec.europa.eu/jrc/en/samo/simlab.

UQLab [27] and SAFE [28] are Matlab-based tools that provide support for UQ (using e.g. PCE) and SA (using e.g. Sobol's method), respectively.

3. The VECMA toolkit (VECMAtk)

The main objectives of VECMAtk are to facilitate the implementation of (a) uncertainty quantification and sensitivity analysis patterns (UQPs) and (b) verification and validation patterns (VVPs). The UQP implementations automate routines for uncertainty quantification and sensitivity analysis, while the VVP implementations enable verification and validation procedures for high performance (multiscale) computing applications. Ye *et al.* present the concepts of UQP on an algorithmic level [29], while an algorithmic paper on VVPs is still in preparation. Both implementations support remote execution on petascale and emerging exascale resources, as well as execution on local machines. In addition, we support a range of existing VVUQ mechanisms and provide tools for users to facilitate the adoption of VVUQ in their applications. As we want our toolkit to be general purpose, taken up by users and flexible, we have four main factors that shape our development approach:

(i) the need to fit into existing applications with minimal modification effort,
(ii) the need to support any application domain,
(iii) the flexible and recombinable nature of the toolkit itself, and
(iv) the geographically distributed nature of the users and particularly the developers.

As a result, we position VECMAtk as a collection of elements that can be reused and recombined in different workflows, interlinked with stable interfaces, data formats and application programming interface (APIs), to facilitate VVUQ in any application. VECMAtk specifically allows users with (multiscale) applications to combine generic, lightweight patterns to create a system-specific VVUQ approach that covers all relevant space and time scales, including the scale-bridges between them. The exemplar applications (which we present in §5) map to all three of the established multiscale computing patterns: Extreme Scaling (ES), Replica Computing (RC) and Heterogeneous Multiscale Computing (HMC) [30].

(a) Release schedule and changes compared to the initial release

The VECMA toolkit is an open-source and open development project, meaning that the latest source and development notes can be accessed at any time. In terms of releases, we have adopted a schedule with minor and major releases. Minor releases are made publicly every three months, are advertised within the project, and have limited amount of additional documentation and examples. The first major release was made in June 2019; the ensuing two releases will be made in June 2020 and June 2021, and will be public and fully advertised. They are accompanied with extensive documentation, tutorials, examples, training events and dedicated uptake activities. In addition to these two release types, we provide informal periodic updates to the master code branches, documentation and the integration of the components.

Since our first major release in June 2019, all VECMAtk components have improved scalability, a range of new features and capabilities, extended and revised application tutorials, improved technical documentation, and a wide range of user-requested bugfixes. We present the fundamental improvements and changes in the development of each VECMAtk component since the first major release as part of appendix A.

4. Overview of the VECMAtk components

We present a graphical overview of the various components in VECMAtk in figure 1. VECMAtk contains four main components: *EasyVVUQ* [31] simplifies the implementation and use of VVUQ workflows with focus on large scale scenarios, *FabSim3* [32] helps to automate computational

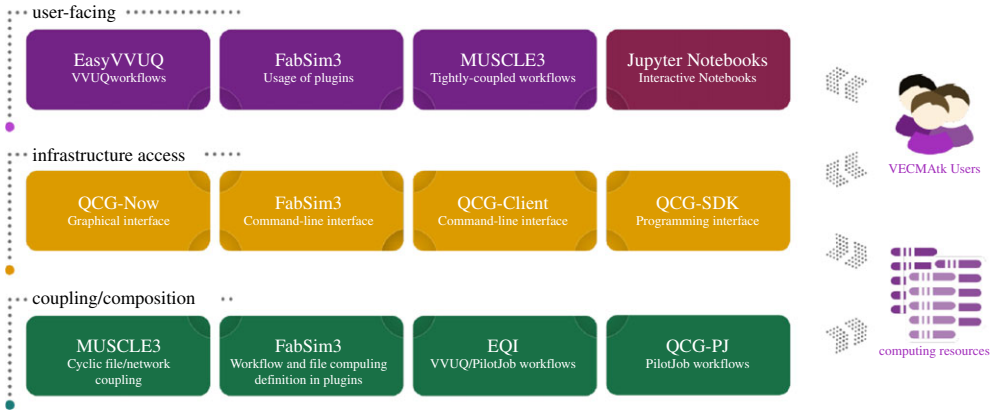

Figure 1. Overview of the main components, and their role within the VECMA toolkit. (Online version in colour.)

research activities, *MUSCLE 3* [33] supports the coupling multiscale applications, and the *QCG tools*[5] facilitate advanced workflows, and the process of design and execution of applications, using HPC infrastructures. All these components can be flexibly combined with other VECMAtk and third party components to create diverse application workflows. In addition, no single component is essential to all applications: for example, FabSim3 and QCG-Client are to a limited degree interchangeable, and at least one application fully relies on a third party tool (the Mogp emulator[6]) instead of EasyVVUQ.

In this section, we summarize each of the components that comprise VECMAtk, while we refer to the VECMAtk website https://www.vecma-toolkit.eu/ for more detailed technical information.

(a) EasyVVUQ

EasyVVUQ [31,34] is a Python library designed to facilitate implementation of advanced non-intrusive VVUQ techniques, with a particular focus on high performance computing, middleware agnosticism, along with single-scale and multiscale modelling. Generally, non-intrusive VVUQ workflows involve sampling the parameter space of the simulation. This usually means running the simulation multiple times. This process is tedious and error prone if done in an *ad hoc* manner, and further exacerbated if the parameter space is large with a corresponding large number of runs (e.g. of order thousands to millions of runs), or if each simulation is very computationally expensive. EasyVVUQ allows one to coordinate the whole process—from sample generation through execution to analysis—although the execution stage is external to EasyVVUQ, and can be done by tools such as QCG-PilotJob, FabSim3 or Dask, or even manually. EasyVVUQ seeks to be agnostic to the choice of execution middleware, due to the large range of possible execution patterns that may be required in an HPC workflow. Additionally, the library carefully tracks and logs applications of the sampling elements along the way, allowing a reasonable level of restartability and failure resistance.

EasyVVUQ breaks down the VVUQ process into five distinct stages.

(i) Application description, which further can be divided into the following items:
   (a) Encoder: responsible for producing input files for the simulation.
   (b) Decoder: responsible for parsing the output files of the simulation and extracting the needed values.
   (c) Execution action: describes how the simulation is run.

[5]See https://www.qoscosgrid.org.

[6]See https://github.com/alan-turing-institute/mogp_emulator.

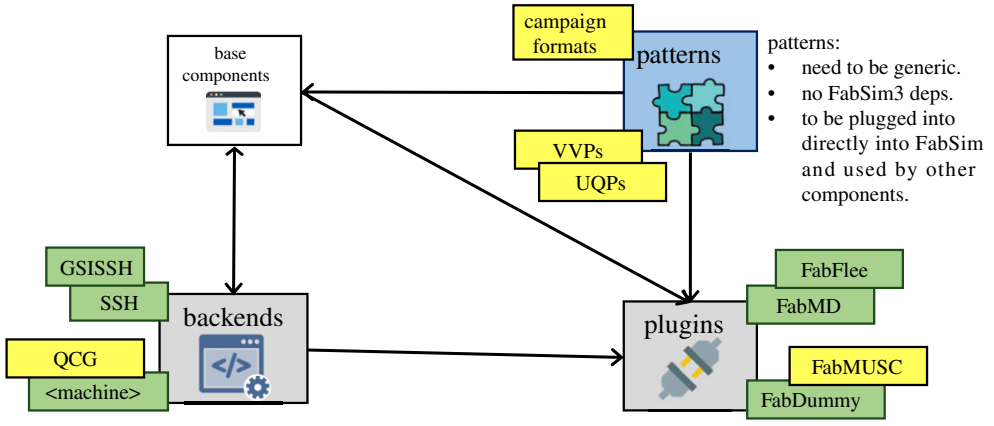

Figure 2. High-level overview of the FabSim3 architecture, showcasing a few of the key plugins, patterns and back-end functionalities available in the tool. Boxes in green (darker) are completed, while boxes in yellow (lighter) are working, but subject to further extensions and improvements. (Online version in colour.)

(ii) Sampling: this step is very dependent on the VVUQ technique in question. The main task of sampling components is to produce a list of parameter sets for the simulation.
(iii) Execution: this is handled entirely externally to EasyVVUQ.
(iv) Collation: collects the outputs of the simulations and then stores them in the EasyVVUQ database, to be retrieved later for analysis.
(v) Analysis: perform statistical analysis on the collected data. This analysis can then inform further actions such as collecting more samples.

EasyVVUQ currently has support for variance-driven sensitivity analysis, stochastic collocation and polynomial chaos expansion methods, as well as more basic statistical analysis methods such as bootstrapping. Additionally, all components of EasyVVUQ are easily extensible, allowing users to customize every stage of the VVUQ workflow to suit their needs. Nevertheless, for many applications the built-in components provide out-of-the-box functionality, requiring the user only to define input and output formats and perform analysis.

(b) FabSim3

FabSim3[7] [32] is a Python-based toolkit for automating complex computational research activities which has several aims: First it helps to automate and simplify the creation, management, execution and modification of complex application workflows. To do so, FabSim3 supports functionalities such as ensemble runs, remote executions, iterative processing and code couplings. Second, it aims to help researchers become quicker and more systematic in handling complex applications in particular. To support this, FabSim3 is designed to be easy to customize for use on new machines and with new simulation codes, and automatically curates a wide range of environment and state variables to aid the testing and debugging of application runs. As usage examples, researchers may want to run and rerun static configurations, run a range of slightly different workflows, define new types of complex applications altogether or construct a routine to automatically validate their code.

FabSim3 also supports a range of adaptable mechanisms for code coupling, and domain-agnostic code patterns that provide building blocks for enabling VVUQ in new applications. We provide an overview of the FabSim3 architecture in figure 2.

In the context of VECMAtk, FabSim3 plays a key role in introducing application-specific information in the Execution Layer, enabling users to combine different UQPs and VVPs, and

[7]See https://github.com/djgroen/FabSim3.

providing an approach to curate large sets of production runs. Also, FabSim3 can use QCG-Client or QCG-SDK to submit ensemble runs via a 'pilot job' mechanism [1,35] (see §4d(ii)), which boosts efficiency by starting and managing sub jobs without each of them needing to individually wait for resources to become available.

(c) MUSCLE 3

The third version of the Multiscale Coupling Library and Environment [33,36] (MUSCLE 3) is intended to simplify the coupling of temporally or spatially scale-separated submodels into a single (distributed) simulation. At run-time, the submodels are started in parallel, and exchange information via the network. Submodels and other simulation components are unaware of each other as MUSCLE 3 delivers the messages to receivers given in a configuration file. The Multiscale Modelling and Simulation Framework (MMSF, [37,38]) may be used to determine how the submodels should be coupled given their relative spatial and temporal scales.

MUSCLE 3 supports time-scale separated macro-micro couplings, including automatically re-running the micro model as often as needed to match the macro model's time steps. It also supports space scale separation, which entails coupling a single macro model to a set of micro models, via support of sets of multiple submodel instances and special vector ports that can be used to communicate with them. Model settings (parameters and technical configuration) are put in the single configuration file, and are transmitted to the individual instances automatically. Components may be inserted into the simulation that generate a parameter overlay, which combined with vector ports and sets of instances yields an ensemble, for example, UQ. MUSCLE 3 supports a range of UQPs, including those for semi-intrusive UQ [29,39]. With MUSCLE 3, this can be done by changing the configuration file to add a few more components and modifying the connections; the submodels can remain unaltered.

(d) QCG tools

The QCG suite of tools help to facilitate advanced workflows, and the process of design and execution of applications, using HPC infrastructures. Four specific QCG components form part of VECMAtk: QCG-Client, QCG-PJ, EQI and QCG-Now, which we now describe in turn.

(i) QCG-Client

QCG-Client is a command line tool to access remote computing infrastructure. The tool allows submission of different types of jobs, including complex workflows, parameter sweep tasks and array jobs, on single or multiple clusters. It uses the QCG-Broker service to manage the execution of workflows, e.g. through multi-criteria selection of resources. QCG-Client can be deployed as a container and is in this form integrated with FabSim3. This allows users to select from both command-line tools, QCG-Client and FabSim3, when they access QCG services.

(ii) QCG Pilot Job

To perform UQ procedures, we must be able to flexibly and efficiently execute a large number of simulations. This is because we need a large and dynamically created parameter-space, which in turn feeds into an ensemble of model executions to get statistically correct results. Within VECMAtk, we use QCG-PilotJob (QCG-PJ) by default to fulfil this requirement, primarily because it can be installed automatically by users without admin rights, and because of its scalability and limited dependency footprint. For instance, QCG-PJ can be set up by users without requiring other components from the QCG environment.

QCG-PJ is designed to schedule and execute many small jobs inside one scheduling system allocation on an HPC resource. Direct submission of a large group of jobs to a scheduling system can result in a long completion time as each single job is scheduled independently and waits in a queue. In addition, the submission of a group of jobs is often restricted (and sometimes even

prohibited) by system administrators. Some systems do support bespoke job array mechanisms, but these mechanisms normally only allow jobs with identical resource requirements. Jobs that are part of a multiscale simulation or application workflow by nature vary in requirements and therefore need more flexible solutions.

To use QCG-PJ, a user submits a job with a QCG-PJ Manager instance, which is executed like a regular job. However, unlike regular jobs this job internally manages a user-defined workflow consisting of many sub-jobs. The manager executes commands to submit jobs, cancel jobs and report resources usage; it may either be preset or listen dynamically to requests from the user. QCG-PJ then manages the resources and jobs in the system, taking into account resource availability and mutual dependencies between jobs. Users can interact with QCG-PJ either using a file-based or network-based interface. The file-based approach works well for static scenarios where the number of jobs is known in advance, while the network interface allows users to dynamically send new requests and track execution of previously submitted jobs at run-time. Although intended for HPC resources, QCG-PJ Manager also supports a local execution mode to allow users to test their scenarios on their own machines (with requiring a scheduling system allocation).

(iii) EQI

Many workflows involving EasyVVUQ require a large number of jobs to be executed, and EasyVVUQ delegates this responsibility to other tools (either existing, or user created). QCG-PJ is the main tool available in VECMAtk to fulfil that purpose, and it offers a flexible but generic API that is less appropriate for specific use cases. To provide an API that is specifically adjusted to EasyVVUQ, we have developed EQI, which is an acronym for the (E)asyVVUQ-(Q)CG-PJ (I)ntegration API. This API introduces domain-oriented concepts, such as predefined tasks for execution and encoding, that facilitate the easy integration of EasyVVUQ workflows with QCG-PJ. With the API, users can perform VVUQ computations with QCG-PJ, invoking HPC resources. The development life-cycle of EQI is synchronized with the releases of both EasyVVUQ and QCG-PJ, to ensure a persistently up-to-date API.

(iv) QCG-Now

Since the traditional command-line interfaces may be perceived by many users to be too complicated, within VECMAtk we decided to provision a easy-to-use graphical tool for users. QCG-Now is graphical desktop programme that enables the submission and management of jobs on HPC resources. It also supports automatic notifications, data sending and receiving. Since the first release of QCG-Now last year, several improvements have been made: for instance it is now integrated with the QCG-Monitoring system which allows users to track progress of execution of their tasks directly from its graphical interface. This capability is particularly important for VECMAtk users who want to keep track of the progress of long-lasting executions of EasyVVUQ and QCG-PJ workflows. Another new feature allows users to store frequently used execution schemes as quick templates for easy reuse. We present a graphical demonstration of the tool in figure 3.

(e) VECMAtk workflows

In figure 4, we show the main components and a few examples as to how they have been combined, leveraging added values where relevant while maintaining a limited deployment footprint. These components can be combined, but also integrated with third party components.

Up to now, the following combinations have been particularly common:

Figure 3. QCG-Now with the embedded QCG-Monitoring view. The integration with QCG-Monitoring allows users to track progress of execution of applications in a graphical way, for example presenting dynamically updated tables, images or, as in this screenshot, charts. (Online version in colour.)

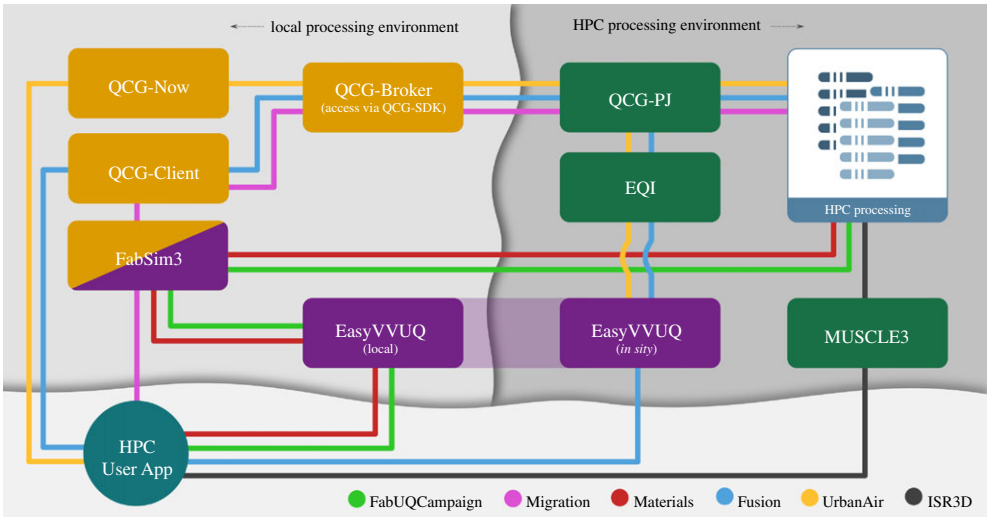

Figure 4. 'Tube map' showing which VECMAtk components are used in the various exemplar applications. VECMAtk components are given in boxes, and the application tutorials are indicated using coloured lines. Note that code using the EasyVVUQ [34] library may be located either on the local desktop for ease of use or on a remote HPC resource for improved performance. Source: https://www.vecma-toolkit.eu/toolkit/. (Online version in colour.)

— FabSim3 + QCG-Client: enables users to submit their job to QCG-Broker to schedule them across multiple remote (QCG-supporting) machines. Additionally, FabSim3 tool support other job scheduling system such as SLURM,[8] Portable Batch System (PBS).[9]

[8]See https://slurm.schedmd.com/overview.html.

— FabSim3 + EasyVVUQ: enables users to automate the VVUQ processes into their applications workflow. After creating their EasyVVUQ campaigns, users can convert them to FabSim3 ensembles using a one-liner (`campaign2ensemble`) command and submit the ensemble jobs through FabSim3. Then they convert results back to EasyVVUQ using the `ensemble2campaign` command, where it is decoded and further analysed.
— FabSim3 + QCG Pilot Job: enables users to create and manage pilot jobs using FabSim3 automation.
— EasyVVUQ + QCG Pilot Job + EQI: enables EasyVVUQ users to execute their tasks directly using pilot jobs.

There are also several examples of integrations with third party components. For instance, there is an application example that uses the mogp emulator in place of EasyVVUQ,[10] and a working EasyVVUQ example that relies on Dask instead of QCG-PilotJob[11] for ensemble job submission. Other integration are indeed possible as well, and the inclusion of a different job management and execution engine may be beneficial for individual applications and/or resources. To name a few examples, RADICAL-Cybertools can offer exceptional performance on machines where it is set up [17], Swift/T [20] can facilitate complex data flows with minimal serialization while Parsl [21] is optimized for the fast and flexible execution of Python programs and Jupyter notebooks.

As indicated in figure 1 earlier, there are four components that provide a suitable entry point for new users to the toolkit. EasyVVUQ facilitates users that initially focus on incorporating VVUQ in their simulations. FabSim3 is aimed at users that initially focus on enabling complex and curated workflows, which could involve ensemble or dynamic execution. QCG-Now is intended for users that prefer a graphical interface for managing their simulations and VVUQ workflows. And MUSCLE3 is well suited for users whose primary concern is multiscale code coupling.

5. Exemplar applications

With VECMAtk, we aim to provide a toolkit that can be applicable to any domain/application where UQ and SA are required, and computational modelling and simulation can be applied. To achieve this aim, we rely on an increasing number of leading-edge scientific applications to support the testing and development of the toolkit. Within this paper, we present a selection of these applications which cover the following topics: (a) future energy sources, (b) human migration, (c) climate, (d) advanced materials, (e) urban air pollution and (f) biomedicine and (g) epidemiology. For each application we show how the toolkit provides added value in terms of enabling UQ. In addition, we describe how we use Verification and Validation Patterns (VVPs) to help facilitate V&V for the first two applications. In each of these concise application descriptions, we focus on how VECMAtk is being used by a range of different applications, which components are primarily employed/invoked, and what primary benefits the toolkit delivers for the application users.

(a) Fusion

Thermonuclear fusion has the potential of becoming a new source of energy that is carbon-free. The dynamics of fusion plasmas span a wide range of scales in both time and space, as, for example, micro-instabilities formed in plasma turbulence can interrupt the overall macroscopic transport and destroy confinement. To simulate such phenomena, we have built a multiscale fusion workflow that includes equilibrium, turbulence and transport physics to model the plasma dynamics [40].

[9]See https://www.pbspro.org/.

[10]See https://github.com/alan-turing-institute/mogp_emulator.

[11]See https://easyvvuq.readthedocs.io/en/dev/dask_tutorial.html.

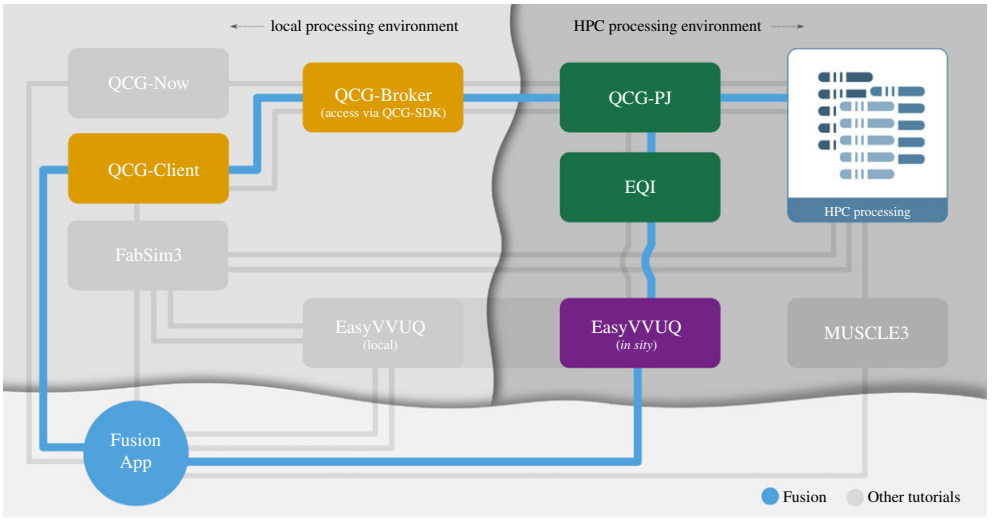

Figure 5. 'Tube map' showing which VECMAtk components are used by the Fusion application. (Online version in colour.)

Our central purpose is to ensure that the results are reproducible and can become a reliable representation of the experiment measurements. To achieve this, it is key to use VVUQ approaches.

We integrated UQ methods using the EasyVVUQ library and performed a parallel execution of samples in a single batch allocation with QCG-PJ through the EQI. Those tools, available as part of the VECMA toolkit (figure 5) and in conjunction with the coupled workflow nature of the fusion application, have proven to be simple to use and to allow a large variety of experiments with different methods and models (see [41] for further details). We also performed a validation using the `ValidationSimilarity` VVP, which is implemented in EasyVVUQ and help us find similarities between the probability distributions of the quantities of interest in our simulation results and in the experimental measurements [42]. For this purpose, we can choose between several relevant metrics, such as Hellinger distance [43], Jensen–Shannon distance which is a symmetrized and smoothed version of the Kullback–Leibler divergence [44] and Wasserstein metrics [45].

(b) Forced human migration

Forced migration reached record levels in 2019, and forecasting it is challenging as many forced population datasets are small and incomplete, and armed conflicts are often unpredictable in nature [46]. Nevertheless, forced migration predictions are essential to improve the allocation of humanitarian support by governments and NGOs, to investigate the effects of policy decisions and to help complete incomplete data collections on forced population movements. Through the use of the FLEE[12] agent-based simulation code, we model and forecast the distribution of incoming refugees across destination camps for African countries [47].

The VECMA toolkit provides us with the ability to automatically construct, execute and analyse ensemble simulations of refugee movements [48], to efficiently compute the sensitivities of key parameters in our simulation [4], to couple different types of models to form multiscale workflows [49] and to facilitate the rapid execution of large job ensembles on supercomputers. We also use a VVP named `EnsembleValidation` to systematically validate our simulations against data from a range of conflicts (see e.g. here [50]). This VVP performs a validation on the output directory of each run and uses an aggregation function to combine all output validation results into a combined metric.

[12]See https://github.com/djgroen/flee-release.

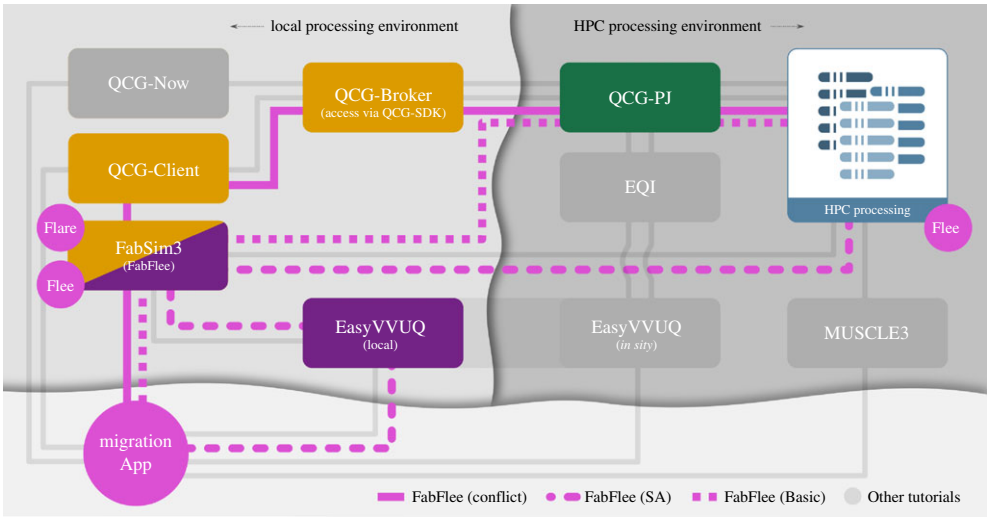

Figure 6. Tube Map showing which VECMAtk components are used in the FabFlee plugin. VECMAtk components are given in boxes which enable users to include their added values while retaining a limited deployment footprint. The black and grey lines define the FabFlee path using VECMAtk components for execution. (Online version in colour.)

We summarize our main application workflows in figure 6. We rely on the FabFlee plugin in FabSim3 to automate the simulation activities required to analyse different policy decisions, to provide the VVP functionality and to facilitate coupling with other models and data sources. Examples include an acyclic coupling to a conflict model [49], but we are also working on a cyclic macro–micro model for the South Sudan conflict. We also use EasyVVUQ with FabFlee to perform sensitivity analyses for varying agent awareness levels, speed limits of refugee movements, location move chances and other simulation parameters [4]. Lastly, we use QCG Pilot Job to facilitate the very large number of runs required to do this analysis, which is required for our more advanced workflows that so far involved up to 16 384 jobs.

(c) Climate

For climate modelling, the range of space and time scales present in (geophysical) turbulent flow problems poses challenges, as this range is too large to be fully resolved in a numerical simulation. As such, micro-scale effects on the resolved macroscopic scales must be taken into account by empirical parametrizations (e.g. [51]). These parametrizations often involve a number of coefficients, for which the value is only known in an approximate manner. In addition, they are subject to the so-called model error or model-form uncertainty, which is the uncertainty introduced due to the assumptions made in the mathematical form of the parametrization scheme.

Within VECMA, we focus on the uncertainty in time-averaged climate statistics of geophysical flow models, due to various assumptions made in the parametrizations. EasyVVUQ allows us to assess the uncertainty in the output statistics due to the imperfectly known coefficients. In addition, we currently use EasyVVUQ for uncertainty quantification of an established local atmospheric model, DALES [52,53], which is used to simulate atmospheric processes including turbulence, convection, clouds and rain. Here, we quantify the uncertainty in model output from different sources, including uncertain physical input parameters, model choices and numerical settings, and the stochastic nature of turbulence modelling. To run the EasyVVUQ ensemble of simulations on HPC resources we use FabSim3, see figure 7. To address the more fundamental model-form uncertainty, we are currently investigating the use of data-driven stochastic surrogates to replace traditional deterministic parametrizations; see e.g. [55,56] for recent results.

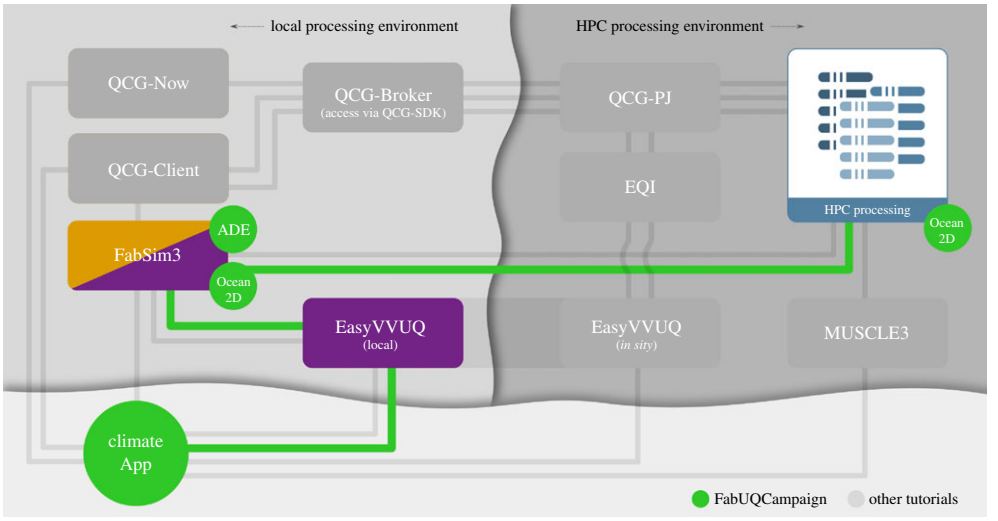

Figure 7. A Tube Map showing the VECMA components used in the climate application. Here 'FabUQCampaign' is a plugin used to run an ensemble of EasyVVUQ samples on HPC resources. Furthermore, 'ADE' and 'Ocean2D' are two example applications (advection diffusion equation and a two-dimensional ocean model) for which tutorials can be found in [54]. (Online version in colour.)

(d) Advanced materials

Developing novel advanced materials can be a very expensive and time consuming process, often taking over 20 years from initial discovery to application.[13] Chemical and material modelling techniques are well developed for single scales, but engineering applications require understanding across many scales [57–59]. Making actionable predictions with these modelling techniques, to improve on the laborious experimental development process, will require harnessing multiple simulation techniques and providing tight error bars on their predictions.

Using VECMAtk, one can generate, execute, collate and analyse ensemble simulations of mechanical tests in an automatic fashion. This allows us to effectively gather statistics on a system of interest and explore an input parameter space with minimal human oversight.

To do this, we use EasyVVUQ to explore an input parameter space by generating ensembles of simulations and to collate the results into an easily analysable object. To distribute the large number of simulations to be run in each ensemble, we use FabSim3 to automate the submission of jobs on remote HPC resources.

(e) Urban air pollution

Predicting air quality in complex urban areas is a challenging field where researchers need to balance accuracy against a practical turnaround time. There are numerous models for predicting contamination transport and dispersion, ranging from simple Gaussian models (that are fast and cheap but not necessarily accurate) to computational fluid dynamics simulations that are slower, more costly and potentially very accurate.

In all cases the most important, and often missing part, is an accurate emission database that contains pollutant emission rates for different types of sources: point (e.g. industrial chimneys), line (e.g. road transportation) and areas (e.g. house heat appliances). The pollutants we consider are NO_2/NO_x, SO_2 and two types of particulate matter (also known as floating dust): $PM_{2.5}$ for particles 2.5 µm or less in diameter and PM_{10} for particles 10 µm or less in diameter. We

[13]See https://www.mgi.gov/.

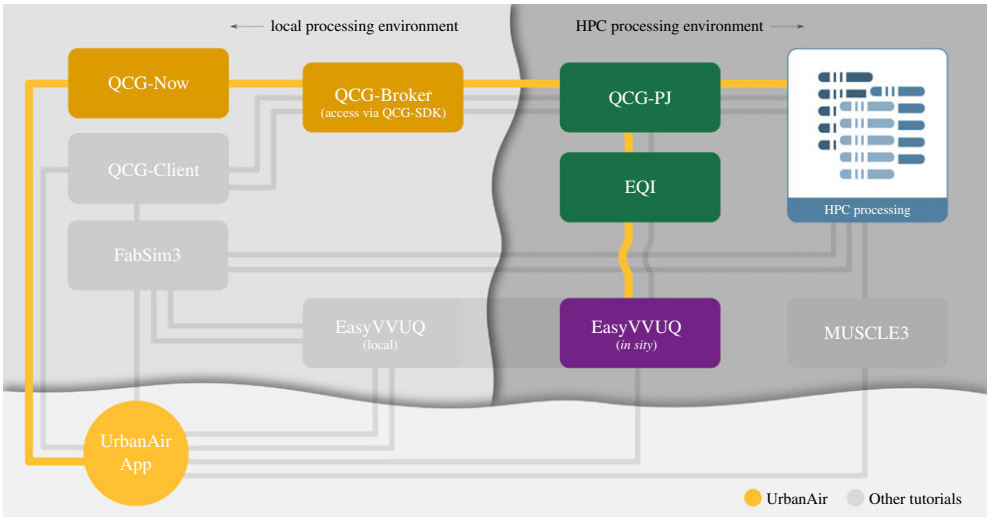

Figure 8. 'Tube map' showing which VECMAtk components are used by the UrbanAir application. (Online version in colour.)

use a mesoscale weather prediction model WRF [60] and an all-scale geophysical flow solver EULAG [61] in our multiscale simulation [62] to address uncertainties coming from incomplete emission database, and to provide a quantitative air quality prediction [63].

We use VECMAtk to automate sampling, ensemble generation, execution of simulation codes on HPC machines, collating results and post-execution analysis. This automation simplifies the provision of better prediction results, because we analyse many different initial conditions and can automatically provide mean results (or some weighted ones) from all simulations. We are also able to run sensitivity analysis of the input parameters to identify and select the most important ones, thus limiting the number of required ensembles for future runs.

In terms of tools, we rely on EasyVVUQ, coupled with QCG-PJ using EQI. Using Python we define the parameter space to be sampled, application to be run in HPC environment, hardware requirements (e.g. number of nodes and cores), and the parameters to be analysed after simulations ended. We then use QCG client and QCG-PJ to submit and control jobs on the HPC resources, EasyVVUQ to generate the samples, EQI for enabling the samples to be executed with QCG-PJ and EasyVVUQ to analyse the results. We summarize our workflow in figure 8.

(f) Biomedicine

In-stent restenosis (ISR) is the renewed occurrence of arterial stenosis (narrowing of an artery) after it was initially treated by installing a metal stent, as a result of excessive cell growth. The results of uncertainty and sensitivity analysis for a two-dimensional model of in-stent restenosis (ISR2D) are presented in [64]. Improved effectiveness of uncertainty propagation was achieved by applying a semi-intrusive metamodelling method and the results are demonstrated in [65] and in [66].

Current research on the design of the UQ and SA experiments for the three-dimensional model of the ISR model (ISR3D) is ongoing. The ISR3D model is implemented using MUSCLE 3 discussed in §4c, which allows performing more advanced semi-intrusive algorithms. In particular, we plan to train a metamodel on the fly and combine that with a cross-validation test of the effect of the approximation error on the results of the micro model.

(g) Coronavirus modelling

Since 2019, the world has been severely affected by the spread of the SARS-CoV-2 coronavirus, and the COVID-19 disease it causes. Although many models are able to approximate the viral

spread on the national level, few solutions are available to forecast the spread on a local level, e.g. in towns or city boroughs. Within the HiDALGO project[14] we are working on the Flu And Coronavirus Simulator (FACS)[15] to help address this challenge.

We use VECMAtk primarily to systematically validate the code, analyse parameter sensitivities, execute ensemble simulations to account for aleatory uncertainty in the code, and to easily produce ensemble forecasts which involve a wide range of scenarios, applied to a range of boroughs in London.

To achieve this, we primarily rely on FabCovid19, which is a FabSim3 plugin, as well as QCG-PJ. Both tools combined allow us to automate these tasks, and seamlessly create, run and post-process ensemble simulations. We also use EasyVVUQ to analyse parameter sensitivities. To demonstrate in what way the VECMA toolkit adds value to our application in a concrete way, we present an small example sensitivity analysis study in appendix B.

6. Toolkit performance and scalability

The VECMA toolkit has been developed for use with large supercomputers, and therefore needs to be both fast and scalable. In this section, we present a range of key performance aspects of the toolkit, and discuss the advantages and limitations of VECMAtk in terms of scalability and exascale readiness.

The main performance-critical aspects of the toolkit involve:

(a) The sampling of parameter values and creation of large numbers of simulation inputs.
(b) The submission, execution and retrieval of large ensembles of simulation jobs.
(c) The efficient movement of data between local and remote resources.
(d) The efficient movement of data between coupled models.

We will now briefly discuss the performance-related characteristics of VECMAtk in relation to these four potential bottlenecks:

Sampling parameter values and creating large numbers of simulation inputs. There are multiple ways in which EasyVVUQ helps with the issues involved in sampling many and expensive simulation outputs. First of all, EasyVVUQ uses an SQL backend database that can handle 10000s of samples or more. It also tracks the status of job execution, allowing users to cope with hardware failures by restarting failed jobs without having to rerun successful ones. Furthermore, EasyVVUQ allows one to sample in stages—appending more samples if results are not yet sufficiently robust, without losing the results already in the database. Lastly, EasyVVUQ can delegate the ensemble job submission to other scalable components, such as Dask or QCG-PJ. These components then execute part or all of the simulations on HPC resources.

Submission and execution management of large ensembles of simulation jobs. 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.[16]

To improve the FabSim3 ensemble submission performance, we integrated it with QCG-PJ and enabled multi-threaded 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[17] application. Owing to limitation of maximum number of submitted job per user[18]

[14]See http://hidalgo-project.eu.

[15]See https://facs.readthedocs.io.

[16]See VECMA Deliverable 5.2: https://www.vecma.eu/wp-content/uploads/2019/12/VECMA_D5.2_First-Report-Infrastructure_PSNC_20191208.pdf.

[17]See https://github.com/djgroen/FabCovid19.

[18]The maximum number of jobs a user can have running and pending at a given time is 5000. If this limit is reached, new submission requests will be denied until existing jobs in this association complete.

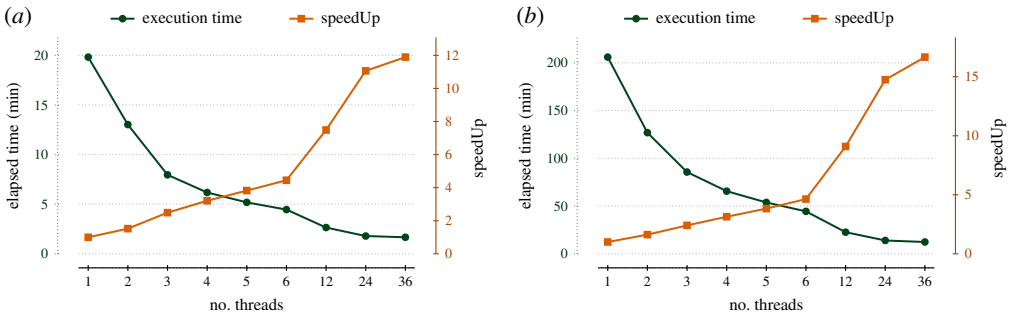

Figure 9. Time required to submit 15121/4865 jobs with FabSim3 (with/without QCG-PJ) relative to the number of job submission threads used. Graph is made using average of 10 repetition of each ensemble size. Please note that, here we only measure the job submission overhead, so, queuing time and job execution on computing nodes are not considered in our test. (*a*) ensemble size = 15121, QCG-PilotJob = True, (*b*) ensemble size = 4865, QCG-PilotJob = False. (Online version in colour.)

Table 1. Comparison of the three main ensemble job submission approaches in VECMA, in terms of general usability, whether files are staged to and from the remote resource as part of the approach, the need for remote deployment work and the overall performance.

				submission overhead	
approach	usability	remote file stage?	remote deployment?	per 100 jobs	per 1000 jobs
FabSim3 only	excellent	yes	not needed	40–90 s	not attempted
FabSim3 + QCG-PJ	excellent	yes	user-space only	33–36 s	250–300 s
QCG-PJ only	good	no	user-space only	<5 s	40 s

on the Eagle supercomputer, we use ensembles sizes of 4865 and 15 121 runs, and use up to 36 threads for the submission process. We enabled pilot jobs for the larger ensemble, but not for the smaller ensemble. We present the results of this scalability test in figure 9.

For FabSim3 ensembles without the use of QCG-PJ, we find that the job submission overhead is 2-3 seconds per job when using a single thread, but decreases to less than a second per job when using four or more threads. When using FabSim3 with QCG-PJ, the job submission overhead reduces by a factor 2 for ensembles with 100 jobs to about 0.35 s per job. The overhead further diminishes for larger ensembles, as we measure an overhead of about 0.275 s per job for ensembles with 1000 jobs. In table 1, we compare the three main ensemble submission approaches in VECMA in terms of performance, usability and in regards to the need of any remote deployment.

Efficient movement of data between local and remote resources. Large simulations, as well as large ensembles of smaller simulations, are often accompanied by the production of large amounts of complex data. Within the VECMA project, we find that the organizing and re-organizing of these data are a cognitively intensive task that is prone to human error if not automated. Both the FabSim3 and EasyVVUQ tools provide a range of data structure conventions that transparently automate low-level data management aspects, allowing users to focus on the complexities that occur at higher levels. For instance, FabSim3 automatically curates input and output and while EasyVVUQ automates the encoding and decoding of simulation files. In addition, FabSim3 and QCG-Client automatically perform file staging to and from HPC resources.

In terms of performance, we chose to focus less on optimizing file transfer rates (although FabSim3 and QCG-Client do support GridFTP), and more on limiting the number of file transfers altogether. The clearest example of this optimization is the tight integration of EasyVVUQ directly with QCG-PJ. Using EQI, users can use HPC resources to generate, run and analyse their

simulation ensembles for VVUQ purposes, and do this iteratively and dynamically within a single Python script. In this way, all the computations are brought to where the data are, and input files only need to be staged in once in advance, even if the workflow contains a large number of (dynamic) iterations.

The efficient movement of data between coupled models. MUSCLE 3 is primarily positioned to address this bottleneck, and provides added value for instance by eliminating the need for file I/O in the coupling.

7. Conclusion

In this paper, we have presented the VECMA toolkit for the verification, validation and uncertainty quantification (VVUQ) of single and multiscale applications. As showcased by the wide range of applications that use the toolkit already, VECMAtk is unique in its ability to combine a wide range of VVUQ procedures with a streamlined automation approach, and trivially accessible capabilities to execute large job ensembles on pre-exascale resources. In addition, VECMAtk components can be flexibly combined, allowing users to take advantage of parts of the toolkit while retaining a very limited deployment footprint.

As part of this paper, we have described a number of exemplar applications from a diverse range of scientific domains (fusion, forced migration, climate, advanced materials, urban air pollution, biomedical simulation and coronavirus modelling), all used by researchers today. All these examples are open source and accompanied with tutorials on http://www.vecma-toolkit.eu/, allowing new users to exploit them as building blocks for their own applications.

Through the VECMA toolkit, we aim to make VVUQ certification a standard practice and to make it simpler to quantify uncertainties, parameter sensitivities and errors in scientific simulations. VECMAtk is used to establish these procedures irrespective of the application domain, allowing key VVUQ algorithms to be reused across disciplines. Because VVUQ is essential to ensure that the key simulation results are indeed actionable, the VECMA toolkit is an important tool to help ensure that scientific simulations can be responsibly used to inform decision-making.

Data accessibility. This article has no additional data.

Competing interests. We declare we have no competing interests.

Funding. This work was supported by the VECMA project, which has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement no. 800925. The development of MUSCLE3 and its respective description was supported by The Netherlands eScience Center and NWO under grant no. 27015G01 (e-MUSC project). The development of the migration and coronavirus modelling applications was supported by the EU-funded HiDALGO project (grant agreement no. 824115). The calculations were performed in the Poznan Supercomputing and Networking Center.

Appendix A. Improvements since first VECMAtk release

In table 2, we present the fundamental improvements and changes in the development of each VECMAtk component since the first major release.

Appendix B. Demonstration of a single example VVUQ analysis: COVID-19 application

To demonstrate the added value offered by VECMAtk more concretely, we showcase one specific VVUQ procedure example, using the Flu And Coronavirus Simulator. We perform sensitivity analysis across six different input parameters of FACS[19] to identify their sensitivity relative to our quantity of interest (QoI). In table 3, we provide the default value for each parameter along

[19]See https://facs.readthedocs.io.

Table 2. The list of modifications and enhancements in the development of each VECMAtk component since the first annual release in June 2019.

VECMAtk Component	list of improvements and changes in development
FabSim3	— added support for multi-threaded job management
	— added automated installation and configuration of FabSim3 on different OS
	— fixed synchronizing/copying for a large number of files and folders
	— vastly improved the performance of `run_ensemble`
	— revamped the documentation
EasyVVUQ	— added support for vector-valued quantities of interest
	— developed an extensive unit testing suite
	— added support for Dask[a] as an execution back-end
	— improved sparse-grid Stochastic Collocation sampler
	— dimension-adaptive Stochastic Collocation sampler
QCG Pilot Job	— developed the node agent for more efficient launching of jobs across many nodes
	— added more optimal way of handling iterative jobs
	— increased the test base and test code coverage
	— simplified execution of bash scripts as Pilot Job's tasks
	— bugfixes for execution of multi-node Pilot Jobs and OMP jobs
MUSCLE3	— newly added; contributed by the e-MUSC project
	— submodel coupling for spatial and/or temporal scale separation
	— submodel instance sets and ensembles
	— messages may contain grids, lists, dictionaries and basic types
	— support for Python, C++, Fortran and MPI, with tutorials and API documentation
	— automatic building of its dependencies on a variety of platforms
EasyVVUQ-QCGPilotJob	— provided a new API that simplifies the execution of EasyVVUQ scenarios using the QCG Pilot Job system
	— introduced a new mechanism to support custom encoders usage
	— made available a detailed tutorial that can be helpful for new users
QCG-Now	— developed an internal view for monitoring of applications (integration with QCG-Monitoring)
	— provided quick-template functionality for frequently submitted tasks

[a]See https://dask.org.

with the range of likely values. We use the Chaospy library[20] in EasyVVUQ to generate samples from the input parameters. Specifically, in this example we used the stochastic collocation method with a sparse-grid sampling plan of 15 121 samples, which we then convert to simulation inputs using the EasyVVUQ encoder, and submit to the HPC resource using FabSim3. Once execution has concluded, we then decode and collate the results and perform a Sobol sensitivity analysis relative to our QoI (number of deaths over time).

We present the first-order Sobol sensitivity indices for each parameter in table 3 and in figure 10. These global sensitivity indices [67] measure the fraction of the output variance (over time here), that can be attributed to a single input parameter. During the first 20 days, all parameters except for the (non-mild) recovery period have a significant effect on the number of deaths. However, as the simulation progresses, the number of deaths is mainly sensitive to the

[20]See https://pypi.org/project/chaospy.

Table 3. Defining parameter space for the uncertain parameters of the Flu And Coronavirus Simulations (FACS).

parameters	type	default value	uniform range
infection rate	float	0.07	(0.0035, 0.14)
mortality period	float	8.0	(4.0, 16.0)
recovery period	float	8.0	(4.0, 16.0)
mild recovery period	float	8.05	(4.5, 12.5)
incubation period	float	3.0	(2.0, 6.0)
period to hospitalization	float	12.0	(8.0, 16.0)

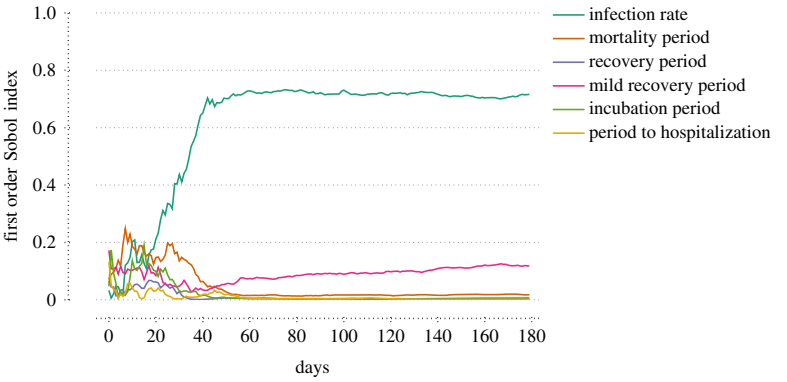

Figure 10. The first-order Sobol indices for each of the uncertain parameters of the Flu And Coronavirus Simulations (FACS) for the London Borough of Brent. (Online version in colour.)

infection rate (which describes how quickly the infection spreads) and the mild recovery period (which describes how long people with mild COVID remain ill and infectious).

References

1. Groen D *et al.* 2019 Introducing vecmatk-verification, validation and uncertainty quantification for multiscale and hpc simulations. In *Int. Conf. on Computational Science, Faro, Portugal*, pp. 479–492. Berlin, Germany: Springer. (doi:10.1007/978-3-030-22747-0_36)
2. Roy CJ, Oberkampf WL. 2011 A comprehensive framework for verification, validation, and uncertainty quantification in scientific computing. *Comput. Methods Appl. Mech. Eng.* **200**, 2131–2144. (doi:10.1016/j.cma.2011.03.016)
3. National Research Council of the National Academies. 2012 Assessing the reliability of complex models: Mathematical and statistical foundations of verification, validation, and uncertainty quantification. National Academies Press. (doi:10.17226/13395)
4. Suleimenova D, Arabnejad H, Edeling WN, Groen D. 2021 Sensitivity-driven simulation development: a case study in forced migration. *Phil. Trans. R. Soc. A* **379**, 20200077. (doi:10.1098/rsta.2020.0077)
5. Schwer LE. 2009 Guide for verification and validation in computational solid mechanics. In *the 20th Int. Conf. on Structural Mechanics in Reactor Technology*. New York, NY: American Society of Mechanical Engineers. See https://repository.lib.ncsu.edu/bitstream/handle/1840.20/23659/3_paper_2010.
6. Oberkampf WL, Roy CJ. 2010 *Verification and validation in scientific computing*. Cambridge, UK: Cambridge University Press.
7. Simmermacher T, Tipton G, Cap J, Mayes R. 2015 The role of model V&V in the defining of specifications. In *the Conf. Proc. of the Society for Experimental Mechanics Series*,

Orlando, FL (eds H Atamturktur, B Moaveni, C Papadimitriou, T Schoenherr). Model Validation and Uncertainty Quantification, vol. 3. pp. 257–263. Cham, Switzerland: Springer. (doi:10.1007/978-3-319-15224-0_27)

8. Binois M, Gramacy R, Ludkovski M. 2018 Practical heteroscedastic Gaussian process modeling for large simulation experiments. *J. Comput. Graph. Stat.* **27**, 808–821. (doi:10.1080/10618600.2018.1458625)
9. Baker E, Gramacy R, Huang J, Johnson L, Mondal A, Pires B, Sacks J, Sokolov V. 2020 Stochastic simulators: an overview with opportunities. (http://arxiv.org/abs/2002.01321)
10. Adams BM, Bohnhoff WJ, Dalbey KR, Eddy JP, Eldred MS, Gay DM, Haskell K, Hough PD, Swiler LP. 2009 DAKOTA, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: version 5.0 user's manual. Sandia National Laboratories, Technical Report. SAND2010-2183. (doi:10.2172/991841)
11. Lin G, Engel DW, Eslinger PW. 2012 Survey and evaluate uncertainty quantification methodologies. Pacific Northwest National Lab.(PNNL), Richland, WA (United States). (doi:10.2172/1035732)
12. Foley SS, Elwasif WR, Bernholdt DE, Shet AG, Bramley R. 2010 Many-task applications in the integrated plasma simulator. In *the 3rd Workshop on Many-Task Computing on Grids and Supercomputers*, pp. 1–10. IEEE. (doi:10.1109/MTAGS.2010.5699425)
13. Elwasif WR, Bernholdt DE, Pannala S, Allu S, Foley SS. 2012 Parameter sweep and optimization of loosely coupled simulations using the DAKOTA toolkit. In *the 15th Int. Conf. on Computational Science and Engineering, Nicosia, Cyprus*, pp. 102–110. Piscataway, NJ: IEEE. (doi:10.1109/ICCSE.2012.24)
14. Debusschere B, Sargsyan K, Safta C, Rai P, Chowdhary K. 2018 UQTk: a flexible Python/C++ Toolkit for Uncertainty Quantification. Albuquerque, NM: Sandia National Lab. (SNL-NM).
15. Giles M. 2015 Multilevel Monte Carlo methods. *Acta Numerica* **24**, 259–328. (doi:10.1017/S096249291500001X)
16. Baudin M, Dutfoy A, Iooss B, Popelin A. 2015 Open TURNS: an industrial software for uncertainty quantification in simulation. In *Handbook of uncertainty quantification* (eds R Ghanem, D Higdon and H Owhadi). Cham, Switzerland: Springer. See http://arxiv.org/abs/1501.05242.
17. Balasubramanian V, Jha S, Merzky A, Turilli M. 2019 Radical-cybertools: middleware building blocks for scalable science. See http://arxiv.org/abs/1904.03085.
18. Gattiker JR. 2008 Gaussian process models for simulation analysis (GPM/SA) command, function, and data structure reference. Los Alamos National Laboratory, Technical Report LA-UR-08-08057. See https://www.lanl.gov/org/docs/gpmsa-command-ref.pdf.
19. Gattiker J, Myers K, Williams B, Higdon D, Carzolio M, Hoegh A. 2017 Gaussian process-based sensitivity analysis and Bayesian model calibration with GPMSA. In *Handbook of uncertainty quantification*, pp. 1–41. Cham, Switzerland: Springer. (doi:10.1007/978-3-319-11259-6_58-1)
20. Wozniak JM, Armstrong TG, Wilde M, Katz DS, Lusk E, Foster IT. 2013 Swift/T: large-scale application composition via distributed-memory dataflow processing. In *2013 13th IEEE/ACM Int. Symp. on Cluster, Cloud, and Grid Computing*, pp. 95–102. (doi:10.1109/CCGrid.2013.99)
21. Babuji Y *et al.* 2019 Parsl: pervasive parallel programming in python. In *Proc. of the 28th Int. Symp. on High-Performance Parallel and Distributed Computing (HPDC '19)*. Association for Computing Machinery, New York, NY, USA, 25–36. (doi:10.1145/3307681.3325400)
22. Tennøe S, Halnes G, Einevoll GT. 2018 Uncertainpy: a python toolbox for uncertainty quantification and sensitivity analysis in computational neuroscience. *Front. Neuroinformatics* **12**, 1–29. (doi:10.3389/fninf.2018.00049)
23. Lawrence Livermore National Laboratory. 2016. *Non-intrusive uncertainty quantification: PSUADE*. Livermore, CA: Lawrence Livermore National Laboratory. See https://computing.llnl.gov/projects/psuade-uncertainty-quantification.
24. Hittinger JA, Cohen BI, Klein RI. 2010 *Uncertainty quantification in the fusion simulation project verification and validation activity*. Livermore, CA: Lawrence Livermore National Laboratory. (doi:10.2172/1119966)

25. Brown JD, Heuvelink GBM. 2007 The data uncertainty engine (DUE): a software tool for assessing and simulating uncertain environmental variables. *Comput. Geosci.* **33**, 172–190. (doi:10.1016/j.cageo.2006.06.015)
26. Salvatier J, Wiecki TV, Fonnesbeck C. 2016 Probabilistic programming in Python using PyMC3. *PeerJ Comput. Sci.* **2**, e55. (doi:10.7717/peerj-cs.55)
27. Marelli S, Sudret B. 2014 UQLab: a framework for uncertainty quantification in Matlab. In *the 2nd Int. Conf. on Vulnerability, Risk Analysis and Management, Liverpool, UK*, pp. 2554–2563. (doi:10.1061/9780784413609.257)
28. Pianosi F, Sarrazin F, Wagener T. 2015 A Matlab toolbox for global sensitivity analysis. *Environ. Model. Softw.* **70**, 80–85. (doi:10.1016/j.envsoft.2015.04.009)
29. Ye D, Veen L, Nikishova A, Lakhlili J, Edeling W, Luk OO, Krzhizhanovskaya VV, Hoekstra AG. 2021 Uncertainty quantification patterns for multiscale models. *Phil. Trans. R. Soc. A* **379**, 20200072. (doi:10.1098/rsta.2020.0072)
30. Alowayyed S, Groen D, Coveney PV, Hoekstra AG. 2017 Multiscale computing in the exascale era. *J. Comput. Sci.* **22**, 15–25. (doi:10.1016/j.jocs.2017.07.004)
31. Jancauskas V, Lakhlili J, Richardson RA, Wright DW. 2020 EasyVVUQ. See https://github.com/UCL-CCS/EasyVVUQ.
32. Groen D, Bhati AP, Suter J, Hetherington J, Zasada SJ, Coveney PV. 2016 FabSim: facilitating computational research through automation on large-scale and distributed e-infrastructures. *Comput. Phys. Commun.* **270**, 375–385. (doi:10.1016/j.cpc.2016.05.020)
33. Lourens V. 2020 MUSCLE 3: the multiscale coupling library and environment. See https://github.com/multiscale/muscle3.
34. 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**, 1–8. (doi:10.5334/jors.303)
35. Luckow A, Santcroos M, Weidner O, Merzky A, Maddineni S, Jha S. 2012 Towards a common model for pilot-jobs. In *Proc. of the 21st Int. Symp. on High-Performance Parallel and Distributed Computing*, pp. 123–124.
36. Veen LE, Hoekstra AG. In press. *Easing multiscale model design and coupling with muscle 3. Computational Science – ICCS 2020*. Berlin, Germany: Springer.
37. Borgdorff J, Falcone J, Lorenz E, Bona-Casas C, Chopard B, Hoekstra AG. 2013 Foundations of distributed multiscale computing: formalization, specification, and analysis. *J. Parallel Distrib. Comput.* **73**, 465–483. (doi:10.1016/j.jpdc.2012.12.011)
38. Chopard B, Borgdorff J, Hoekstra AG. 2014 A framework for multi-scale modelling. *Phil. Trans. R. Soc. A* **372**, 20130378. (doi:10.1098/rsta.2013.0378)
39. Nikishova A, Hoekstra AG. 2019 Semi-intrusive uncertainty propagation for multiscale models. *J. Comput. Sci.* **35**, 80–90. (doi:10.1016/j.jocs.2019.06.007)
40. Luk OO, Hoenen O, Bottino A, Scott BD, Coster DP. 2019 ComPat framework for multiscale simulations applied to fusion plasmas. *Comput. Phys. Commun.* **239**, 126–133. Elsevier. (doi:10.1016/j.cpc.2018.12.021)
41. Lakhlili J, Hoenen O, Luk OO, Coster DP. 2020 Uncertainty quantification for multiscale fusion plasma simulations with VECMA toolkit. In *Computational Science - ICCS 2020* (eds V Krzhizhanovskaya *et al.*). Lecture Notes in Computer Science, vol. 12143. Springer, Cham. (doi:10.1007/978-3-030-50436-6_53)
42. Luk OO, Lakhlili J, Hoenen O, von Toussaint U, Scott BD, Coster DP. 2021 Towards validated multiscale simulations for fusion. *Phil. Trans. R. Soc. A* **379**, 20200074. (doi:10.1098/rsta.2020.0074)
43. Nikulin, Mikhail S. 2001 *Hellinger distance*, vol. 78. Springer, NY: Encyclopedia of mathematics.
44. Kullback S, Leibler RA. 1951 On information and sufficiency. *Ann. Math. Stat.* **22**, 79–86. (doi:10.1214/aoms/1177729694)
45. Villani C. 2016 *Optimal transport: old and new*. Berlin, Germany: Grundlehren der mathematischen.
46. Groen D. 2016 Simulating refugee movements: where would you go? *Procedia Comput. Sci.* **80**, 2251–2255. (doi:10.1016/j.procs.2016.05.400)
47. Suleimenova D, Bell D, Groen D. 2017 A generalized simulation development approach for predicting refugee destinations. *Sci. Rep.* **7**, 13377. (doi:10.1038/s41598-017-13828-9)
48. Suleimenova D, Bell D, Groen D. 2017 Towards an automated framework for agent-based simulation of refugee movements. In *The Proc. of the 2017 Winter Simulation Conf., Las Vegas,*

NV (eds WKV Chan, A DAmbrogio, G Zacharewicz, N Mustafee, G Wainer, E Page), pp. 1240–1251. Piscataway, NJ: IEEE. (doi:10.1109/WSC.2017.8247870)
49. Groen D, Bell D, Arabnejad H, Suleimenova D, Taylor SJE, Anagnostou A. 2019 Towards modelling the effect of evolving violence on forced migration. In *the 2019 Winter Simulation Conf. (WSC)*, pp. 297–307. (doi:10.1109/WSC40007.2019.9004683)
50. Suleimenova D, Groen D. 2020 How policy decisions affect refugee journeys in South Sudan: a study using automated ensemble simulations. *J. Artif. Soc. Soc. Simul.* **23**, 14. (doi:10.18564/jasss.4193)
51. Gent PR, Mcwilliams JC. 1990 Isopycnal mixing in ocean circulation models. *J. Phys. Oceanogr.* **20**, 150–155. (doi:10.1175/1520-0485(1990)020<0150:IMIOCM>2.0.CO;2)
52. Heus T *et al.* 2010 Formulation of the Dutch Atmospheric Large-Eddy Simulation (DALES) and overview of its applications. *Geosci. Model Dev.* **3**, 415–444. (doi:10.5194/gmd-3-415-2010)
53. 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)
54. Edeling W, Groen D. 2019 FabUQCampaign. See https://github.com/wedeling/FabUQCampaign.
55. Edeling W, Crommelin D. 2020 Reducing data-driven dynamical subgrid scale models by physical constraints. *Comput. Fluids* **201**, 1–11. (doi:10.1016/j.compfluid.2020.104470)
56. Crommelin D, Edeling W. 2020 Resampling with neural networks for stochastic parameterization in multiscale systems. (http://arxiv.org/abs/2004.01457)
57. Vassaux M, Sinclair RC, Richardson RA, Suter JL, Coveney PV. 2020 Toward high fidelity materials property prediction from multiscale modeling and simulation. *Adv. Theory Simul.* **3**, 1–18. (doi:10.1002/adts.201900122)
58. Vassaux M, Richardson RA, Coveney PV. 2019 The heterogeneous multiscale method applied to inelastic polymer mechanics. *Phil. Trans. R. Soc. A* **377**, 20180150. (doi:10.1098/rsta.2018.0150)
59. Suter J, Sinclair R, Coveney P. 2020 Principles governing control of aggregation and dispersion of graphene and graphene oxide in polymer melts. *Adv. Mater.* **32**, 2003213. (doi:10.1002/adma.202003213)
60. Chen F *et al.* 2011 The integrated WRF/urban modeling system and its applications to urban environmental problems. *Int. J. Climatol.* **31**, 273–288. (doi:10.1002/joc.215810.1002/joc.2158)
61. Prusa JM, Smolarkiewicz PK, Wyszogrodzki A. 2008 EULAG a computational model for multiscale flows. *Comput. Fluids* **37**, 1193–1207. (doi:10.1016/j.compfluid.2007.12.001)
62. Wyszogrodzki A, Miao S, Chen F. 2012 Evaluation of the coupling between mesoscale-WRF and LES-EULAG models for simulating fine-scale urban dispersion. *Atmos. Res.* **118**, 324–345. (doi:10.1016/j.atmosres.2012.07.023)
63. Wright DW *et al.* 2020 Building confidence in simulation: applications of EasyVVUQ. *Adv. Theory Simul.* **3**, 1900246. (doi:10.1002/adts.201900246)
64. Nikishova A, Veen L, Zun P, Hoekstra AG. 2018 Uncertainty quantification of a multiscale model for in-stent restenosis. *Cardiovasc. Eng. Technol.* **9**, 761–774. (doi:10.1007/s13239-018-00372-4)
65. Nikishova A, Veen L, Zun P, Hoekstra AG. 2019 Semi-intrusive multiscale metamodelling uncertainty quantification with application to a model of in-stent restenosis. *Phil. Trans R. Soc. A* **377**, 20180154. (doi:10.1098/rsta.2018.0154)
66. Ye D, Nikishova A, Veen L, Zun P, Hoekstra AG. 2020 Non-intrusive and semi-intrusive uncertainty quantification of a multiscale in-stent restenosis model. (http://arxiv.org/abs/2009.00354)
67. Saltelli A, Ratto M, Andres T, Saisana M, Tarantola S. 2008 *Global sensitivity analysis: the primer*. New York, NY: John Wiley & Sons.