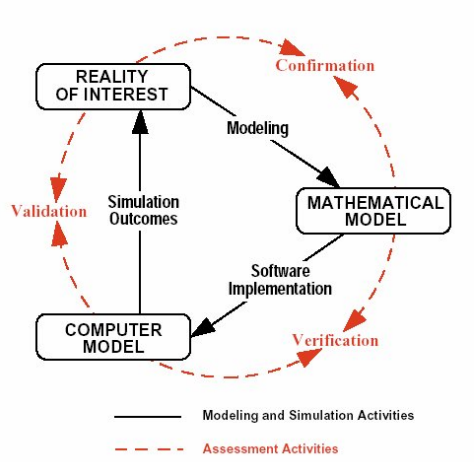
EasyVVUQ: Error Bars for Everyone

David W. Wright & Robin Richardson

12 December 2018

VVUQ

Verification - Does the computational model fit the mathematical description?
Validation - Is the model an accurate representation of the real world?



Uncertainty **Q**uantification - How do variations in the numerical and physical parameters affect simulation outcomes?

B.H.Thacker, *et al.*, “Concepts of Model Verification and Validation.” 2004. DOI: 10.2172/835920.

VECMA - Verified Exascale Computing for Multiscale Applications

Website: www.vecma.eu

Twitter: @VECMA4

- Goal: develop an open source VVUQ toolkit for multiscale/multiphysics applications
- Specifically targetting HPC applications
- Exemplar applications from fusion and advanced materials through climate and migration, to drug discovery and personalised medicine

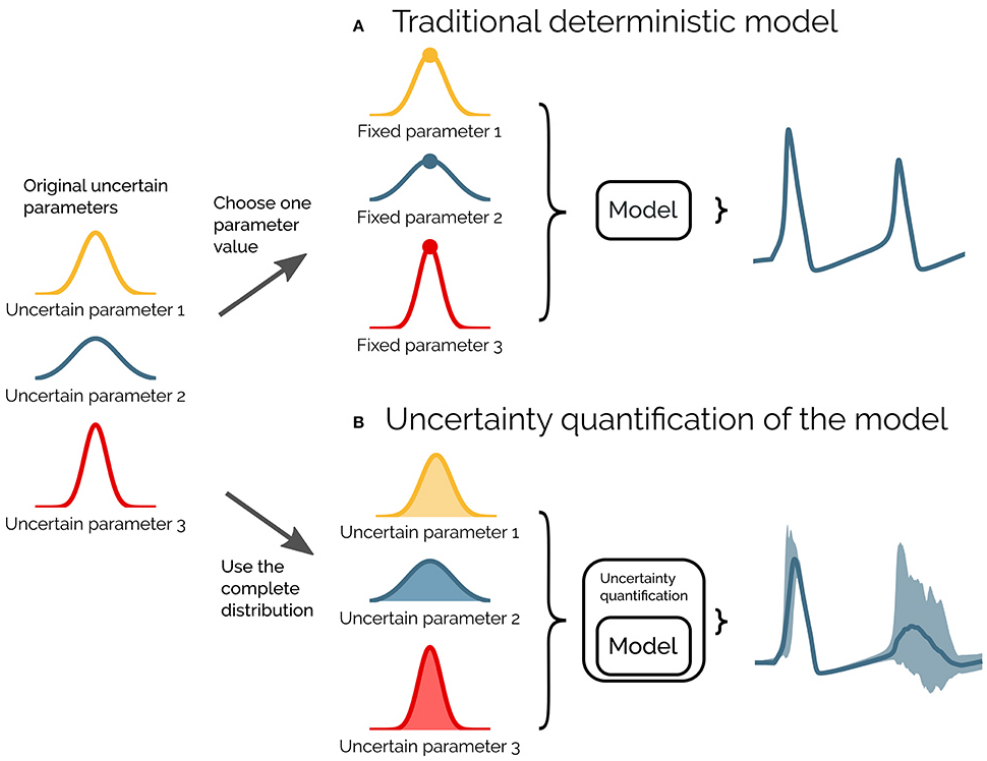
Existing VVUQ Libraries

- Many in Matlab (and engineering focussed)
- Most are Commercial
- Open Source/Python:
  - Uncertaintypy: https://github.com/simetenn/uncertainpy
  - OpenTurns: http://www.openturns.org/
  - UQpy: https://github.com/SURGroup/UQpy
  - UQSA: https://gitlab.com/cavs_group/Simulation_Study_Tools
  - PyMC3: https://github.com/pymc-devs/pymc3
  - UQ Toolkit: https://www.sandia.gov/UQToolkit/

Why a new library

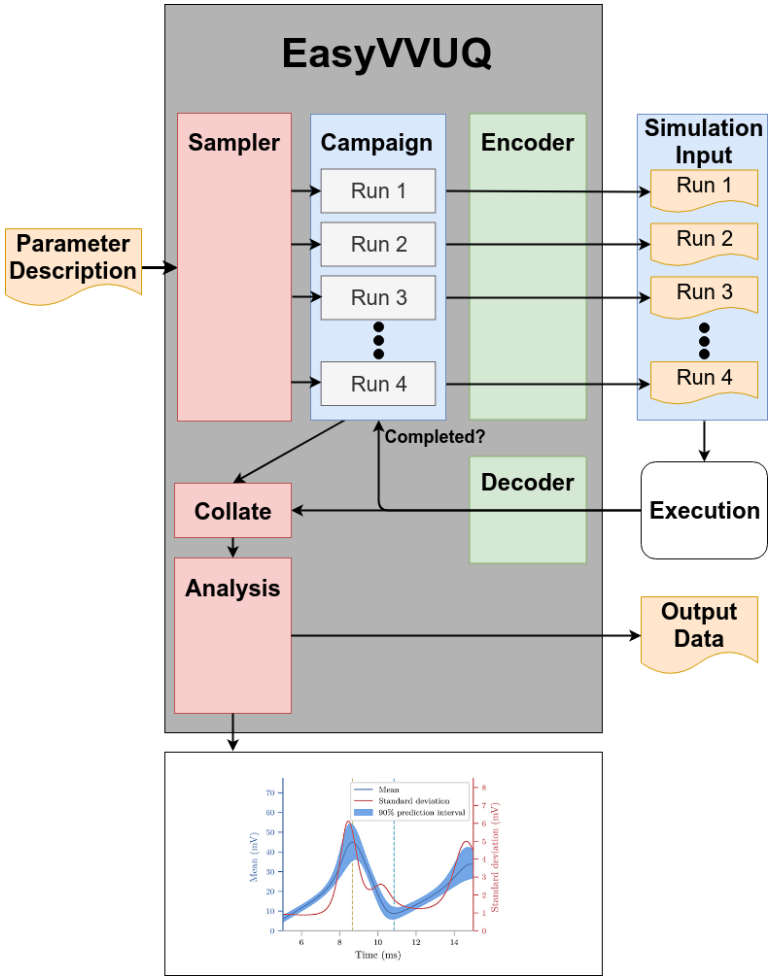
- Separation of simulation from:
  - Parameter sampling
  - Data collation and analysis
- Make it easy to add VVUQ to scientists' existing (HPC) workflows
- Testbed for new techniques developed within VECMA

Uncertainty Quantification



Simen Tennøe *et al.*, "Uncertainpy: A Python Toolbox for Uncertainty Quantification and Sensitivity Analysis in Computational Neuroscience", Front. Neuroinform., 2018 https://doi.org/10.3389/fninf.2018.00049

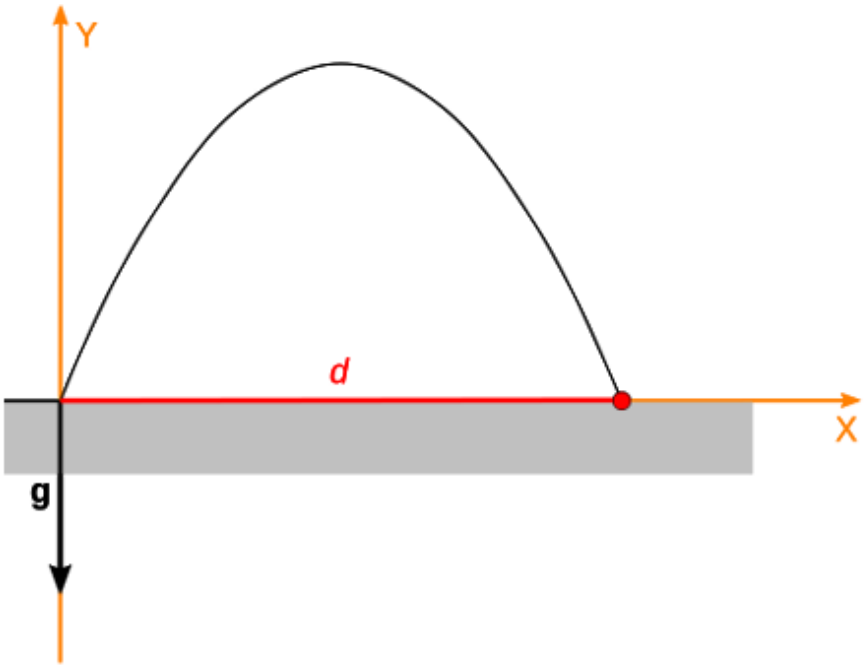
Design overview: High level design



Design overview: Campaigns

- Current implementation is as a Python dictionary
- Creates directories for each run
  - Input files contain parameter values for a specific sample
  - Additional input files copied or linked (e.g. geometries or forcefield files)
- In future scale may require use of other approaches
  - Potentially millions of runs
  - Investigating us of HDF5 with one exemplar project (fusion) in VECMA

User Perspective: Example problem



User Perspective: Parameter description

```
{
"app": {
  "input_encoder":     "generic_template",
  "encoder_delimiter": "#",
  "output_decoder":    "csv",
  "template":          "cannonsim.template",
  "input_filename":    "input.cannon"
  },
"params": {
  "angle":          {"type": "real", "min": "0.0",    "max": "6.28",   "default": "0.79"},
  "air_resistance": {"type": "real", "min": "0.0",    "max": "1.0",    "default": "0.2" },
  "height":         {"type": "real", "min": "0.0",    "max": "1000.0", "default": "1.0" },
  "time_step":      {"type": "real", "min": "0.0001", "max": "1.0",    "default": "0.01"},
  "gravity":        {"type": "real", "min": "0.0",    "max": "1000.0", "default": "9.8" },
  "mass":           {"type": "real", "min": "0.0001", "max": "1000.0", "default": "1.0" },
  "velocity":       {"type": "real", "min": "0.0",    "max": "1000.0", "default": "10.0"}
  }
}
```

User Perspective: Sampling

```
import easyvvuq as uq
input_json = "test_cannonsim.json"
output_json = "out_cannonsim.json"
number_of_samples = 15
my_campaign = uq.Campaign(state_filename=input_json)
# Set parameters to vary
my_campaign.vary_param("angle",
                       dist=uq.distributions.uniform(0.0, 1.0))
my_campaign.vary_param("height",
                       dist=uq.distributions.uniform_integer(0, 10))
my_campaign.vary_param("velocity",
                       dist=uq.distributions.normal(10.0, 1.0))
my_campaign.vary_param(
    "mass",
    dist=uq.distributions.custom_histogram("mass_distribution.csv")
)
# Use sampler to select values of parameters
random_sampler = uq.elements.sampling.RandomSampler(my_campaign)
my_campaign.add_runs(random_sampler, max_num=number_of_samples)

my_campaign.populate_runs_dir()
# Execute runs
```

User Perspective: Campaign


```
"runs":
    "Run_0": {
        "angle": 0.7757645082270815,
        "height": 8,
        "velocity": 10.403787065219165,
        "mass": 0.25,
        "air_resistance": "0.2",
        "time_step": "0.01",
        "gravity": "9.8",
        "completed": true,
        "fixtures": {},
        "run_dir": "/tmp/EasyVVUQ_Campaign_xg82ky42/runs/Run_0"
    },
    "Run_1": {
        "angle": 0.05515909957568221,
        "height": 7,
        "velocity": 8.22498715745362,
        "mass": 0.75,
        "air_resistance": "0.2",
        "time_step": "0.01",
        "gravity": "9.8",
        "completed": true,
        "fixtures": {},
        "run_dir": "/tmp/EasyVVUQ_Campaign_xg82ky42/runs/Run_1"
    },
    .....
```

User Perspective: Analysis

```
output_filename = 'output.csv'
output_columns = ['Dist', 'lastvx', 'lastvy']
uq.elements.collate.aggregate_samples(my_campaign,
                                      output_filename=output_filename,
                                      output_columns=output_columns,
                                      header=0)
stats = uq.elements.analysis.BasicStats(my_campaign, value_cols=output_columns)
results, output_file = stats.apply()
my_campaign.save_state(output_json)
```

Encoders

- Convert parameters for a given run into simulation input
- We provide generic subsititution based encoders for common formats
- When this is not sufficient an ‘expert user’ develops a new encoder
  - Inherit from `BaseEncoder` class
  - Implements an `encode` function taking two inputs
    - `params` - dictionary containg run parameters
    - `target_dir` - directory for simulation inputs

Encoders: Generic

Template file (`cannonsim.template`):

```
CANONSIM_INPUT_FILE:
gravity = #gravity
mass = #mass
velocity = #velocity
angle = #angle
height = #height
air_resistance = #air_resistance
time_step = #time_step
```

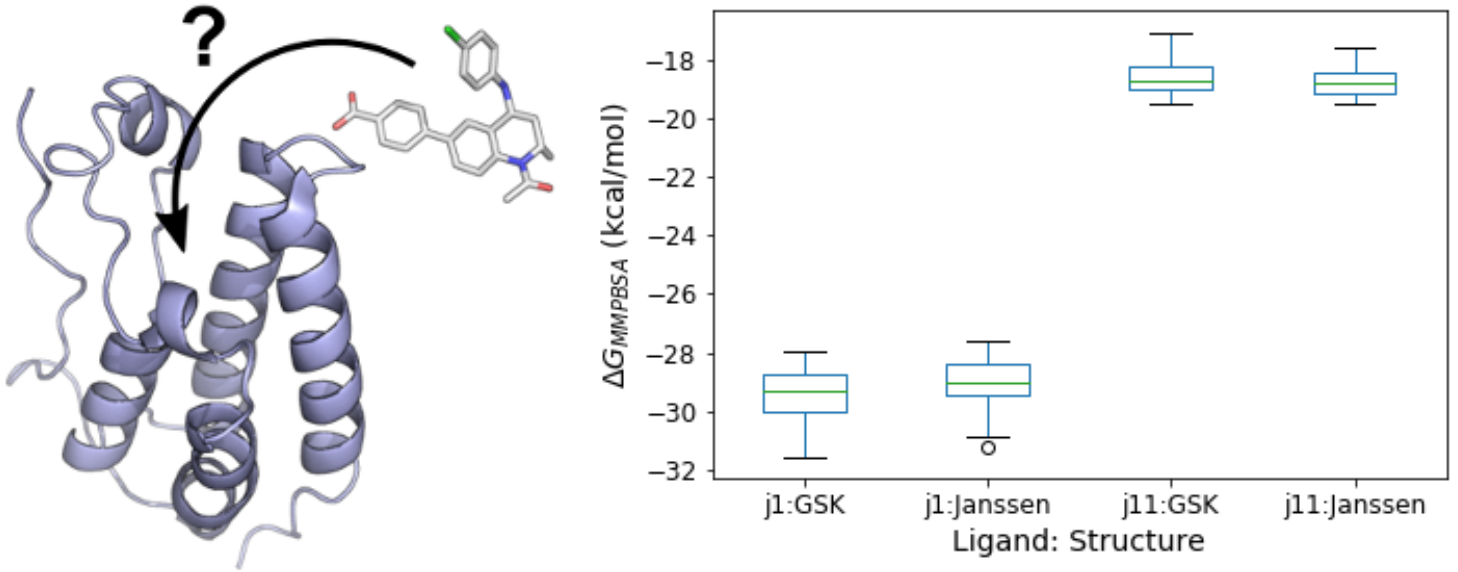
Decoders

- Interpret simulation output file
- `collate` elements loop through runs and use decoder to combine outputs into a pandas dataframe
- Again we implemet generic encoders (e.g. CSV)
- Once more, when this is not sufficient an ‘expert user’ develops a new encoder
  - Inherit from `BaseDecoder` class
  - Implements a `decode` function

Current Status

- Code: https://github.com/UCL-CCS/EasyVVUQ
- Aiming for a **V0.1** release this month
  - Most of the concepts are stable
  - Implementation still evolving
  - Only very basic features implemented
- Starting to write documentation

Exemplar 1: Ensemble drug binding simulations



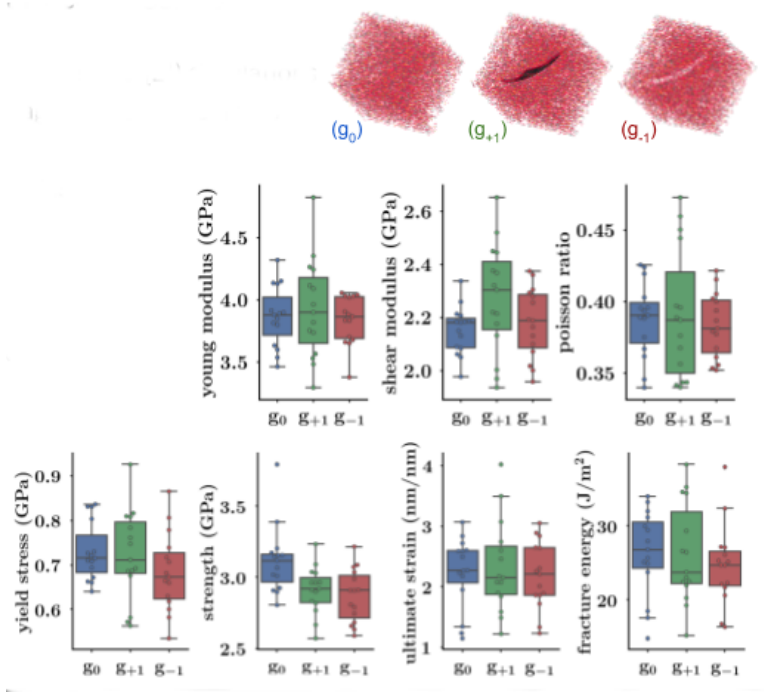
D. W. Wright *et al.* "Application of ESMACS binding free energy protocols to diverse datasets:Bromodomain-containing protein 4", Preprint

DOI: 10.26434/chemrxiv.7327019

Exemplar 2: Atomistic modelling of graphene-epoxy nanocomposites

- LAMMPS simulations with ReaxFF
  - g1 = graphene
  - g0 = no graphene
  - g-1 = graphene shaped hole
- Addition of graphene
  - increased shear modulus
  - reduced strength
  - had hardly any **active** influence

M. Vassaux, R. C. Sinclaur & P. V. Coveney, “The role of graphene in enhancing the properties of epoxy resins”, submitted Advanced Theory and Simulation.



Future Plans

- Implement more realistic sensitivity analysis workflow
  - Ishigami function as a test case
  - Polynomial chaos
- Added functionality
  - Read in existing datasets
  - Combine `Campaigns`
  - Support adding sampling to individual runs
  - Handle bigger larger numbers of runs
- Enable more users
  - Hackathon in early 2019
  - `Encoders`/`Decoders` for specific applications

Acknowledgements and contact

David Wright: dave.wright@ucl.ac.uk
@nothing_counter

Robin Richardson:
robin.richardson@ucl.ac.uk

VECMA

Derek Groen

David Coster

Jalal Lakhlili

CCS

Peter Coveney

Maxime Vassaux

Robbie Sinclar

James Suter