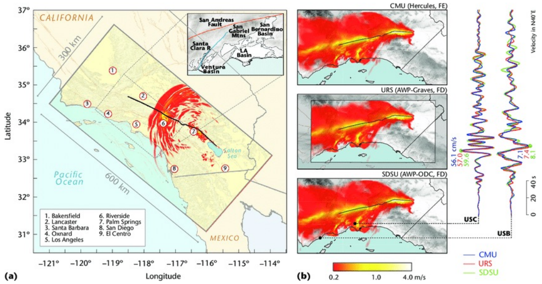
Uncertainty Quantification of Dynamic Rupture Modeling

The Alan Turing Institute

Eric G. Daub[1], Hamid Arabnejad[2], Derek Groen[2]

1. Research Engineering Group, Alan Turing Institute, London 2. Department of Computer Science, Brunel University, London

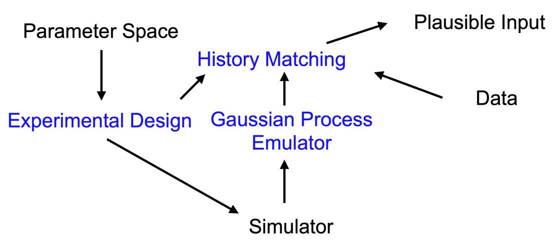
Brunel University London



SCEC activities rely heavily on simulations to study rare earthquake phenomena. Significant challenge to calibrate, validate, and propagate uncertainties through simulations. Complicated further by poor observational constraints, missing physics, and computational limitations.

In other words, how do we put error bars on our (known to be) imperfect simulations in a principled, robust way? This is a general problem, so broad goal is to develop methods that apply to all sorts of simulation phenomena.

Surrogate-based Calibration:



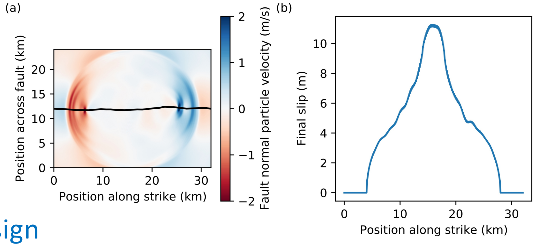
Approach aims to fit a cheap computational approximation, sometimes known as a "surrogate" or "emulator."

1. The simulation is run as many times as is feasible, with an effort to use an Experimental Design to choose points in a way that maximizes the accuracy of the surrogate.
2. These simulations are used to fit an approximation to the simulator based on a Gaussian Process Emulator that, once fit, can quicky estimate the simulator output for an arbitrary input.
3. The parameter space is then densely queried, and points are compared with the available observations while accounting for all uncertainties. Points can be systematically ruled out as unlikely using an approach known as History Matching.

The result is a set of points that are "Not Ruled Out Yet" that are plausible inputs to the simulator given all data and associated modeling uncertainties.

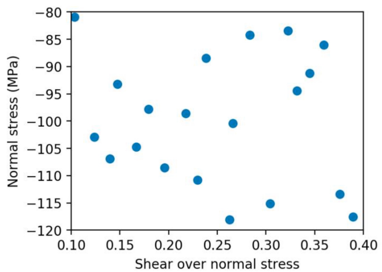
Example Application

Dynamic Rupture on a rough fault. Can we invert seismic moment to estimate the stress state on the fault and calibrate "plausible" values for the initial stress tensor?

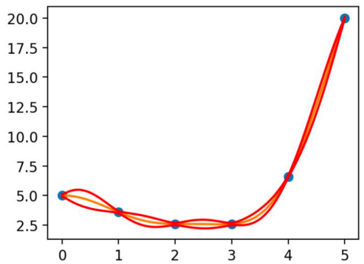


Experimental Design

Intentionally spread points out in parameter space. Use a Latin Hypercube design, which guarantees we sample from every quantile of each input parameter. Some randomness, but generally a good strategy given limited computation. Goal is not to choose best points for fitting the data, but best points for effectively approximating the simulation.



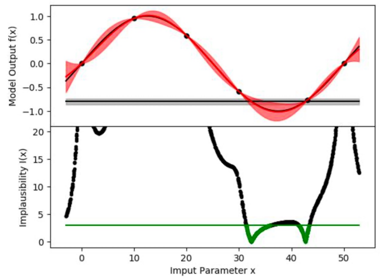
Gaussian Process Emulator



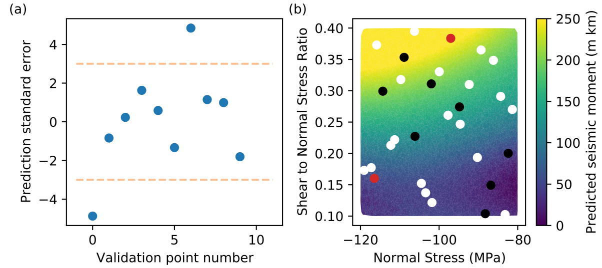
Fit an approximate model using Gaussian Process Regression, a non-parametric Bayesian regression method. GPs tend to be relatively robust to overfitting, and give an uncertainty for their predictions. Once fit, predictions are computationally cheap.

History Matching

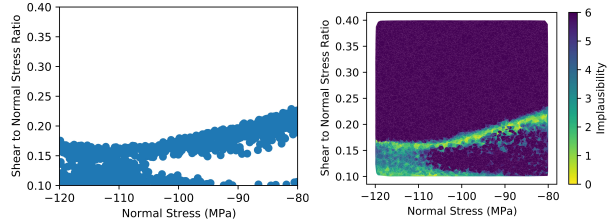
With fit emulator, can use Experimental Design to query densely from parameter space and compare to observations. With History Matching, don't necessarily invert a "best" solution but rather try to rule out points where the emulator prediction is implausibly different from observations (accounting for all uncertainties).



Results



Emulator cross-validation and output: (a) Standard error of fit Gaussian Process predictions on validation data set. The emulator produces reasonable predictions over much of the parameter space. (b) White dots are training points, black dots are valid hold-out test points, and red dots are the points where cross-validation failed. Failures tend to be in points where the simulation behavior changes rapidly, making the approximate emulator overconfident in the output value.



History Matching results for a synthetic test: (left) NROY points tend to cluster around a line with a particular shear/normal stress, with some dependence on the normal stress. This is because seismic moment is highly sensitive to stress, particularly on rough faults. (right) Implausibility metric (standard error between emulator predictions and "observed" value. Even with only 20 simulations, can learn some useful information about the parameter space.

More information:

- Paper: https://royalsocietypublishing.org/doi/full/10.1098/rsta.2020.0076
- Instructions for how to reproduce this work can be found at: https://github.com/alan-turing-institute/fabmogp_paper
- UQ software package implementing the methods described here: https://github.com/alan-turing-institute/mogp-emulator
- `mogp-emulator` documentation with demos: https://mogp-emulator.readthedocs.io