February 2022

1 Emulator effectiveness, effective uncertainty

The goal is to isolate the likely region of parameter space using original set of full-model runs and then augment those runs.

- Obtain simplex points
- Evaluate them thus obtaining training points in parameter space
- Compare to data (chi-squared analysis)
- Determine the likelihood or probability the parameters describe the model given experimental result
- Determine badness of fit after adding a new training point
- Repeat steps 4-5 until representative set of coefficients and parameters is formed
- Weigh that region of parameter space with likelihood when training
- Average over best set of coefficients to finalize emulator

The goal is as follows: we have a model that we want to emulate, however this model is computationally expensive and thus we can't use MCMC techniques as that would take too long. The task therefore becomes to build an emulator which is a simpler version of the model, i.e. it's less accurate but faster. The emulator needs to be trained with training points prior to emulating the true model as a whole. However the problem is that one can't know where the best training points are, one can only choose randomly which poses the problem of not choosing the optimal training set and having an emulator which isn't "good". So the problem arises of how does one defines the "goodness" of an emulator. To do that one might decide to isolate the likelier regions of parameter space to sample and then weigh those regions with the probability likelihood. In this problem there's the experimental data that we want to emulate, Y_{exp}, the mature emulator $E_s(\theta)$ which is what we're trying to develop and the model function which simply goes through all the training points Y_{model}. The emulator being trained is the smooth emulator described previously as:

$$E(\vec{\theta}) = \sum_{\vec{n},\text{s.t. } K(\vec{n})\leq K_{\max}} d_{\vec{n}} f_{K(\vec{n})}(|\vec{\theta}|) A_{\vec{n}} \left(\frac{\theta_1}{\Lambda}\right)^{n_1} \left(\frac{\theta_2}{\Lambda}\right)^{n_2} \cdots \left(\frac{\theta_N}{\Lambda}\right)^{n_N}. \quad (1)$$

We want to isolate the most likely region of parameter space. We have the experimental data Y_{exp}, the model which is the emulator evaluated at the training points given by the simplex and the actual emulator which is given over all the points, training and testing. Here P_s denotes the emulator likelihood i.e. how well determined the training points are. We want to pick the most probable parameters θ to maximize the likelihood which means minimizing χ^2. However we also want to test the reliability of the chosen parameters which goes as:

$$P_s(\theta) \propto e^{[-\chi^2(\theta)/2]} \quad (2)$$

where P_s is the maximum likelihood and where

$$\chi^2_s(\theta) = \frac{(y_i - y_{i,exp})^2}{\sigma^2} \quad (3)$$

where in this case y_i is the emulator $E_s(\theta)$ and $y_{i,exp}$ is Y_{exp}. and

$$U = var = \sum_{s=1}^{N} \int d\theta (P_s(\theta) - \bar{P}(\theta))^2 \frac{1}{N_s} \quad (4)$$

where the variance describes the range of parameter values that don't adversely affect the model fit. $\bar{P}(\theta)$ here represents the maximum likelihood i.e. the most probable parameter set after averaging over the most representative coefficients. It is what we want to find.

The idea behind this is that when you first start the emulation process, you pick a number of training points (which depends on the number of dimensions in parameter space) which are governed by your simplex and there's a number of different parameter combinations that will make it through all the training points given that the number of coefficients A is much larger than the number of training points so we need to narrow it down. Each set or combination of coefficients that goes through the training points is denoted by s as in P_s. The question is then to determine where to place the next training point to optimize improving the emulator. Certain regions of parameter space will be more likely and others will be less likely so it doesn't make sense to prioritize both in the same way. It's best to focus on the likely region of parameter space as that will produce the "best" emulator, i.e. one that is most in agreement with experimental data. It is also computationally advantageous. Considering the computational intensity of the model to begin with the goal is to proceed with as little training data points as possible and sampling the entire parameter space is less efficient because more training points will be necessary to tune it properly.

The approach then in choosing a new point is to place it anywhere close to Y_{exp} and see whether the variance, σ^2 which I will also call the "badness", of

the fit goes up or down. That way you know how much badness is lowered or increased and how likely the set of parameters is to represent the true model. This process will progressively move the working likelihood to the true likelihood of the model. It will allow to generate a representative set of coefficients which can then be averaged out for final emulator values. It's important to note that the emulator is built for one experimental observable which depends on the parameters θ. It is possible that for other observables other emulators will need to be built or at least retrained.