Appendices

A Emulator Parameters

The Emulator Parameter is the file user changes in order to use and optimize the code according to the model the user has and the output user is expecting. The parameters described below can be found in the parameter directory and in the file emulator_parameter.txt.

It is organized in the following way

The parameter map has a prefix for each of the parameters SmoothEmulator and in the appendix, they are mentioned directly with the values in front.

A.1 NTrainingPts

The number of training points determines how many data points are available for training the smooth emulator. It affects the number of training samples used in various calculations and tuning steps, and ultimately influences the accuracy and generalization ability of the emulator model.

A larger number will lead to a more accurate emulator, but it could also increase computational costs. The choice of the number depends on the specific application and the trade-off between accuracy and efficiency.

A.2 SigmaYMin

This value is used to set a lower limit for SigmaY, it might be used to prevent the emulator from overfitting or producing unrealistic predictions by penalizing or restricting very small uncertainties in the observed data.

A.3 NMC

This parameter specifies the number of iterations (steps) to be performed in the MCMC algorithm which is used for tuning the emulator parameters to find the optimal values that maximize the likelihood of the observed data. A larger value of NMC allows for a more thorough exploration of the parameter space, potentially leading to more accurate tuning results.

However, it also increases the computational cost of the tuning process since each iteration requires calculating the emulator output and evaluating the likelihood.

The choice of NMC should be made carefully to strike a balance between the accuracy of the tuning results and the computational efficiency of the algorithm. The optimal value of NMC may depend on the complexity of the problem, the size of the training data, and the available computational resources.

A.4 NASample

This parameter represents the number of samples generated from the smooth emulator to estimate uncertainty in the emulator predictions. It controls the number of times the emulator coefficients (A) are sampled to obtain multiple emulator outputs.

The value of the NASample should be chosen based on the desired level of confidence in the emulator predictions and the computational resources available. Larger values of the NASample generally provide more reliable uncertainty estimates but also require more computational time to generate the samples.

A.5 MCStepSizeis

This parameter controls the step size of the Metropolis-Hastings Markov Chain Monte Carlo (MCMC) algorithm during the tuning process.

A larger value of MCStepSize means larger steps, which can lead to faster exploration of the parameter space but might also result in less precise tuning. Smaller MCStepSize leads to smaller steps, which may improve accuracy but may require more iterations to fully explore the parameter space.

A.6 MCSigmaAStepSize

This parameter controls the step size for updating the standard deviation SigmaA during the Metropolis-Hastings Markov Chain Monte Carlo (MCMC) algorithm's tuning process.

This enables the algorithm to simultaneously find the optimal values of A and SigmaA that best fit the training data and provide accurate emulator predictions.

A.7 TuneChooseMCMC

This boolean parameter that controls the choice of tuning method used in the smooth emulator. The choice between MCMC and "perfect" tuning methods can significantly impact the accuracy and computational efficiency of the emulator.

The MCMC method is more robust and can handle complex parameter spaces, but it requires a larger number of iterations (NMC) and can be computationally expensive. On the other hand, the "perfect" tuning method may be faster, but it may not fully explore the parameter space and may not find the global optimum.

A.8 UseSigmaYRreal

This is also a boolean parameter that determines whether the true (real) standard deviation (SigmaY) of the observed data is used during the parameter tuning process of the smooth emulator.

By considering the true standard deviation SigmaY in the likelihood calculation, the tuning process accounts for the uncertainty in the observed data. This can lead to more accurate parameter

tuning and better emulator predictions, especially when the true standard deviation is known and informative about the variability in the observed data.

A.9 ConstrainA0

This boolean parameter controls whether the emulator coefficients $\boldsymbol{A_i}$ are constrained during the tuning process. Specifically, it determines whether the first coefficient $\boldsymbol{A_0}$ is fixed or allowed to vary during the tuning iterations.

For some problems, it may be known or desired that the first coefficient $\boldsymbol{A_0}$ should take a specific value based on physical constraints or prior knowledge. In such cases, setting the parameter to true ensures that the tuning process respects this constraint.

On the other hand, if there is no specific reason to constrain $\boldsymbol{A_0}$, setting the parameter to false allows the tuning algorithm to explore the full parameter space, including the possibility of $\boldsymbol{A_0}$ taking different values, which might lead to better-fitted emulator coefficients.

A.10 CutoffA

This parameter sets the extra width that keeps sigma A from drifting off to infinity.

A.11 LAMBDA

This parameter represents a regularization term used in the smooth emulator for tuning the model coefficients (A). The term "LAMBDA" is typically used to denote the smoothness parameter and it serves as a tuning knob to control the complexity of the smooth emulator model.

The Lamda helps control the trade-off between fitting the training data accurately and keeping the model complexity in check. A higher value of LAMBDA leads to a more regularized (smoother) model with simpler coefficients, while a lower value allows the model to adapt more to the training data, potentially leading to more complex and flexible coefficients. It is often determined through experimentation and fine-tuning to achieve the best balance between model accuracy and complexity.

A.12 ModelRunDirName

It is a string variable that stores the name of the directory where the smooth emulator stores the information related to a specific model run.

A.13 CoefficientsDirName

It is a string variable that stores the name of the directory where the smooth emulator stores the coefficients (parameters) obtained during the tuning process.

A.14 MAXRANK

It is a variable that represents the maximum rank allowed for the smooth emulator and it determines the number of basis functions used to represent the smooth functions during the modeling process.

A higher MAXRANK allows the emulator to capture more complex and flexible response surfaces, but it may require more data points and computational resources. Conversely, lower MAXRANK results in a simpler model with reduced flexibility but may be more computationally efficient and require fewer data points.

A.15 UseRFactor

It is a boolean parameter that determines whether the smooth emulator uses an "R-factor" regularization term during the tuning process. It is a measure of how well the smooth emulator reproduces the training data points.

By incorporating the R-factor into the tuning process, the emulator can optimize the coefficients (A) in a way that balances the fit to the training data with the overall smoothness of the model.