4 Performing Full Model Runs

There are two forms for storing training data. In the default form, where the parameter `SmoothEmulator_Train` is set to `training_format_smooth`, training data for a run generated from a specific point in model-parameter space is stored in a unique directory. To be more compatible with the *SURMISE* statistical package, there is a second option, where `SmoothEmulator_TrainingFormat` is set to `training_format_surmise`. For this option the model-parameter values for all the training runs are stored in a single file, defined by the parameter `SmoothEmulator_TrainingThetasFilename` and the model observables are stored in a single file `SmoothEmulator_TrainingThetasFilename`. These parameters are defined in `smooth_data/smooth_parameters/emulator_parameters.txt`. =

4.1 Default Format for Training Data

Once the training points are generated, the User must run the full model for each of the training points. At this point there is a directory, usually called `${MY_PROJECT}/modelruns/`, in which there are sub directories, `run0/, run1/, run2/`.... Within each sub-directory, `runI`, there should exist a text file `${MY_PROJECT}/modelruns/runI/mod_parameters.txt`, where $\boldsymbol{I = 0, 1, 2, \cdots}$. These files could have been generated by *Simplex Sampler*, but could have been generated by any other means, including by hand. The files should be of the form,

```
par1_name  par1_value
par2_name  par2_value
par3_name  par3_value
   ⋮
```

The parameter names must match those defined in `${MY_PROJECT}/Info/modelpar_info.txt`, the format of which is described in Sec. **??**.

The User must then perform full model runs using the model-parameter values as defined in each sub-directory. The full model runs should record results by writing a list of observable values in each run directory. Each file must be named `${MODEL_RUN_DIRNAME}/runI/obs.txt`, where $\boldsymbol{I = 0, 1, 2, \cdots}$. The directory `${MODEL_RUN_DIRNAME}/`. Typically this directory is `${MY_PROJECT/moderundirs}`, but that can be changed by the User. The format of those text files should be

```
observable1_name  observable1_value  observable1_random_uncertainty
observable2_name  observable2_value  observable2_random_uncertainty
observable3_name  observable3_value  observable3_random_uncertainty
.
.
```

The names must match those listed in `${MY_PROJECT}/Info/observable_info.txt`, which will be used by *Smooth Emulator*, as described in Sec. **??**. The values are the observable values as calculated by the full model for the model-parameter values listed in the corresponding `mod_parameters.txt` file in the same directory.

The random uncertainties refer only to those uncertainties due to noise in the full model. Random noise is that, which if the full model would be rerun at the same model-parameter values, would represent the variation in the observable values. In most cases this would be set to zero. But, if the full model has some aspect of sampling to it, for example generating observables from event generators with a finite number of events, that variation should be listed here. This variation is required for the emulator. If there is such a variation, the User might not wish to constrain the emulator to exactly reproduce the training point observables at the training points. The principal danger being, that if two training points are very close to one another, but with a finite fluctuation, exactly producing the training points might require very high slopes to exactly reproduce the training points. If the training points are far apart from one another, and if the random uncertainties are not large, it should be safe to ignore the random uncertainty and constrain the emulator to exactly reproduce the model values. Currently, if one wishes to account for the random uncertainty the User must set the following parameters in `parameters/emulator_parameters.txt`:

a) Set either `SmoothEmulator_TuneChooseMCMC` or `SmoothEmulator_TuneChooseMCMCPerfect` to `true`.
b) Set `SmoothEmulator_MCMCUseSigmaY` to `true`.

Once the observable files are produced for each of the full model runs, the User can then proceed to build and tune an emulator using *Smooth Emulator*.

4.2 SURMISE Compatible Format for Training Data

In this case the file defined by the parameter `SmoothEmulator_TrainingThetasFilename`. For example, this might be set to `TrainingThetas.txt`, which would then be located in the top level of the project directory. The User would construct the file in the form

```
X1   X2   X3   ....
X1   X2   X3   ....
  ⋮,
```

where each row listed the model parameters for a specific training run. Similary, the parameter `SmoothEmulator_TrainingThetasFilename` would describe the file where the observables from the training runs were listed. For example, the parameter might be set to `TrainingObs.txt`. The format might look like

```
Y1   Y2   Y3   ....
Y1   Y2   Y3   ....
   ⋮
```

Again, each row corresponds to a different training point. The two files should thus have the same number of rows.

With this option, there is no way to choose a subset of points from which to train.