

MP-SPS (Massively Parallel Sequential Posterior Simulator)

This software package implements the SMC algorithm described in Durham and Geweke (2013). Sample models are provided in the `Models` subdirectory. Sample applications are provided in the `Examples` directory.

For the user who is not familiar with object-oriented programming in Matlab and is interested in delving into the inner working of this software, we recommend the following reading: <http://www.mathworks.com/help/matlab/object-oriented-programming.html>

To implement a new model, users should provide classes extending the `SMC_model`, `SMC_prior`, `SMC_data` and `SMC_design` base classes. These classes encapsulate information needed to run the simulator.

- `SMC_data` : Data container.
- `SMC_prior` : Description of prior.
- `SMC_model` : The model.
- `SMC_design` : Description of Metropolis steps to be undertaken in M phase.

Default implementations of `SMC_prior` and `SMC_design` are provided. These may be sufficient for many applications.

To run the simulator, the user must construct instances of these classes. The algorithm is then run by first instantiating an SMC object and then executing the `run` method. The algorithm is run twice in succession for the hybrid method described in Durham and Geweke (2013), e.g.,

```
smc = SMC( data, prior, model, design );
results = smc.run();
```

A `results` structure is returned. This structure contains various fields that may be of interest.

Class constructors often take arguments. For example, a data class may allow a file to be specified from which data is to be read. A prior class may allow for hyperparameters to be specified. The details will be model dependent, and the best source of documentation is the code itself.

Default settings for most classes can be overridden by supplying optional arguments to the constructor. For example, to change the defaults for J and N in the SMC algorithm, one could call the SMC constructor as follows:

```
smc = SMC( data, prior, model, design, 'J', 64, 'N', 1024 );
```

The global variable `SMC_GPU` controls whether the algorithm is run on the GPU: 0 to run on CPU; 1 to run on GPU using Matlab code for anything model-specific (but some C/CUDA for internal calculations); and 2 to run using custom C/CUDA kernels for model evaluation. Using `SMC_GPU=2` is generally much faster than the alternatives but requires some additional effort to code model-specific CUDA kernels.

The simplest way to understand how to use and extend this software is by studying the sample models and applications provided in the Models and Examples directories. The sample applications can be executed by running the script `run_smc` in the folder where the application files are located, e.g., `Examples/logit`.

Before using this package, some mex files must be compiled. For this, a compliant C compiler is needed (see Matlab documentation for details). To compile, run `make mex` in the SMC root directory. To run on the GPU, the Nvidia CUDA development kit and driver must be installed, as well as the Matlab parallel toolkit. To compile the cuda kernels, run `make cuda` in the SMC root directory.