Supporting material for "A Bayesian data assimilation framework for lake 3D hydrodynamic models with a physics-preserving particle filtering method"

Artur Safin[1], **Damien Bouffard**[1], **Firat Ozdemir**[2], **Cintia L Ramón**[3,1], **James Runnalls**[1], **Fotis Georgatos**[2], **Camille Minaudo**[4], **and Jonas Šukys**[1]

[1]Eawag: Swiss Federal Institute for Aquatic Science and Technology, Switzerland
[2]Swiss Data Science Center, Switzerland
[3]Water Research Institute and Department of Civil Engineering, University of Granada, Spain
[4]École Polytechnique Fédérale de Lausanne, Switzerland

Correspondence: Artur Safin (artur.safin@eawag.ch)

1 Introduction

This supplementary material presents:

1. A directly reproducible example of running a smaller version (in terms of parallelization requirements and dataset length) of the computational model using RENKU containers.

2. Inference results for a calibration model that include eddy viscosity and diffusivity parameters for the hydrodynamic model.

3. Additional details on the time-dependent parameters (Secchi disk depth and surface albedo) used for the hydrodynamic model.

4. A brief discussion on using a two month period (July and August) to calibrate model parameters.

2 Running the model using RENKU interactive environment

A RENKU (Swiss Data Science Center, 2021) image of the numerical software we use to obtain our numerical results can be found at renkulab.io/gitlab/artur.safin/DatalakesHydrodynamics. This image can be launched in a RENKU interactive environment using their `Environments` tab (we recommend requesting at least 4 GB of RAM for the computational resources). Once connected to the running instance, a small inference can be launched to infer model parameters based on a week-long dataset, using initial conditions extracted from the Datalakes website. This process takes approximately 1 hour using 6 parallel tasks to obtain 3 samples. The example is mostly for illustrative purposes, and will not provide useful parameter posteriors. Below we provide a short explanation of how to run the simulation, using shell terminal.

Table 1. Options for `config_settings.py`, where all the global settings are stored.

Parameter(s)	Type	Explanation
`globalStart`, `globalEnd`	`datetime`	The global timeframe for simulation, using UTC time coordinates. Note that each simulation must start and end at midnight.
`startTime`, `endTime`	`datetime`	Parameters to be used for sequential updating (more on this separately). For calibration purposes, as done here, we must set `startTime = globalStart` and `endTime = globalEnd`.
`dayIndex`, `dayDelta`	`int`	Also for sequential updating. Otherwise for calibration should have `dayIndex = 0` and `dayDelta` should ensure that `endTime = globalEnd` holds.
`meshSize`	`int`	Horizontal cell size for the hydrodynamic model. Currently only 1000 meter option is fully supported for inference.
`withRivers`	`True/False`	Whether to use BAFU river data.
`withLSTM`	`True/False`	Flag to use Bi-LSTM for assimilation of LSWT. For each particle (in the particle filter), the network requires 1.5 GB of RAM, which can be quite demanding. If `False`, then a simpler error model will be used to assimilate LSWT.
`initialCond`	`True/False`	Whether to use an initial condition. Should be set to `True` as long as a fully 3D initial condition can be supplied.
`sequential`	`True/False`	Option to turn on/off sequential updating.
`withSecchiD`	`True/False`	If `True`, will try to estimate the Secchi depth on its own using the Ornshtein-Uhlenbeck process. We do not recommend this option, and here use estimated values derived from in-situ photosynthetically active radiation (PAR) data.
`posterior`	`True/False`	Calibrate the hydrodynamic model from the base priors using EMCEE sampler (option `False`) or just draw values from the converged posterior parameter distributions we estimated from 2019 data (option `True`).
`nChains`	`int`	Number of sampler chains for the EMCEE sampler (must be even).
`nParticles`	`int`	Ensemble size for the particle filter.
`nSampler`	`int`	Number of chains to be evaluated in parallel for the EMCEE sampler. Note that `nChains/nSampler` must be an even number.
`nParticles`	`int`	Number of MPI tasks to assign per particle filter. Note that `nEnsemble` must be a divisor of `nParticles`
`nBatches`	`int`	Total number of EMCEE iterations to run.
`sandboxPath`	`str`	Path to the sandbox directory. Can also be `'/dev/shm/'` to only virtually store the intermediate statefiles, but does not work here due to the small size of the directory.

2.1 Import and compile the computational model

The first step is to load the necessary scientific computing packages. The `SPUX` and `MITgcm` source code are installed using `easybuild` tools; in addition we need to load another module, `Datalakes-MITgcm` that adds additional interface routines between the two packages. We can load them using

```
module use ${HOME}/.local/easybuild/modules/all
module load MITgcm SPUX Datalakes-MITgcm
```

Next, we copy the main directory from `SPUX` and import the general settings file modified specifically for this repository, `config_settings.py`:

```
cp -r $SPUX_DIR/examples/MITgcm .
cp /work/$CI_PROJECT/configs/config_settings.py \
  MITgcm/config_settings.py
cd MITgcm/
```

Looking into this configuration file, we can choose the desired timeframe to be simulated, as well as a number of other options, documented in Table 1.

Finally, after selecting the appropriate choices in the file, we compile the model (please note that if you change the configuration file at any point, you may need to re-compile):

```
./configure.sh
```

In particular, you should see `Compilation successful` as part of the output. You will also get 2 segmentation fault error messages as part of `MITgcm` shared library compilation, but they should not indicate any issues.

2.2 Import and prepare an initial condition from the Datalakes website

From the Datalakes website, www.datalakes-eawag.ch/ we can extract 3D predictions relevant to our simulation period, and use them to prepare an initial condition. Using the API documentation for the Datalakes hydrodynamic predictions, we can fetch the relevant source data (234 MB):

```
year=2020
ISOweek=23
basePath='https://api.datalakes-eawag.ch/datalakesmodel/nc'
wget ${basePath}/geneva/${year}/${ISOweek} \
    -O Datalakes_2020_23.nc
```

Note that in the above script, we can select the year and week (ISO format) that we would like to download. Therefore, if you would like your simulation to begin on a different date, then you only need to adjust the parameters to download the ISO week data containing the target date. Next, assuming the download was successful, we now extract a single snapshot from the week-long dataset. This requires the use of python `netCDF4` libraries for the job.

```
mkdir -p datasets
cp /work/$CI_PROJECT/configs/generateInitialCondition.py .
cp /work/$CI_PROJECT/configs/pickup.0000000000.t001.nc datasets
python generateInitialCondition.py
```

The newly generated pickup file will be stored in the `datasets` folder and is now usable for the inference. In the `generateInitialCondition.py` script, you can specify the correct index of the dataset timestamp to be extracted using the `stampIndex` option. In case of a mismatch, the script will print out a warning message.

2.3 Import the observational data, and prepare input files

We simply clone the observational data repository, move it into a correct location (controlled by `config_settings.py`) and run a script to funnel the various sources of data into tables in the `datasets` folder (`dataset.dat` and `RSdata.dat`). Note that this downloads 1.20 GB of data (MeteoSwiss data is large, even in reduced form that re-packaged to).

```
repoRoot='https://renkulab.io/gitlab/artur.safin/'
repoName='datalakes-observational-data-snapshot.git'
git clone ${repoRoot}${repoName}
mv ${repoName} /work/${CI_PROJECT}
./auxiliary_scripts.py datasets
```

Finally, we need to prepare the MeteoSwiss input data, which can be interpolated using data from the same observational repository we just downloaded. The command is simply:

```
./prep_weather.sh
```

2.4 Running the inference

To determine the number of parallel ranks necessary to run the simulation, we can optionally invoke

```
./auxiliary_scripts.py update_job_script
```

which under the default options in `config_settings.py` informs that 6 tasks are required (parallel task partitioning: 1 global master + 1 chain master + 4 workers per chain). The inference is now launched as follows:

```
mpirun -np 6 python -m mpi4py execute.py --connector legacy
```

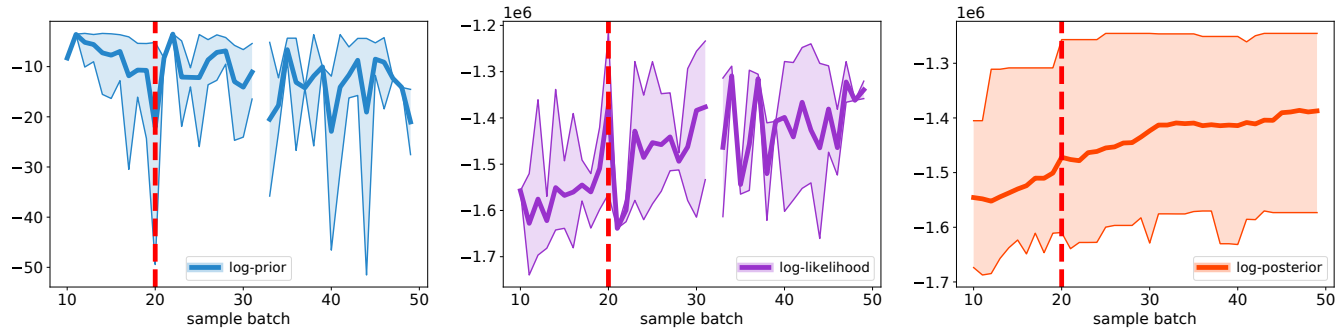

Figure S1. Log-priors, log-likelihoods and log-posteriors for the sampled model posterior parameters (accepted only). The solid lines indicate the mean and the semi-transparent spreads indicate the 5% - 95% percentiles across multiple concurrent chains of the sampler. The red dashed line indicates the posterior estimate at the best found parameters.

Once the simulation completes (in about two hours of real-time), its results are stored in the `output` and `report` folders. We can visualize these predictions using

```
python plot_results.py
```

which will take about 10 minutes. The predictions will be output to the `fig` folder, and a tutorial on how to interpret them is available from the [SPUX readme]. To generate a 3D result, we can run

```
./postprocessing.py Datalakes
./postprocessing.py Datalakes_Weekly
```

The first command generates a snapshot of mean predictions and spreads for every hour, for temperature and velocities, in `3D_states/statistical`. The 10-decimal integer in each filename is the UNIX Epoch time for the particular snapshot. The second command aggregates this data (but now using 3-hour intervals instead) and dumps into files by ISO week, `3D_states/Datalakes-*.nc`. The resulting file can be easily parsed by any `netCDF` readers. For example, below we visualize the evolution of surface temperature:

```
cp /work/$CI_PROJECT/configs/visualize_surface.py .
python visualize_surface.py
```

This will generate a movie called `surfaceT.gif` of the surface temperature.

3 On inferring eddy viscosity and diffusivity parameters for the hydrodynamic model

Originally, we sought to calibrate the horizontal eddy viscosity and diffusivity parameters together with the Smagorinsky viscosity for the `MITgcm` hydrodynamic model. However, this approach provided extremely poor convergence results, and therefore we chose a different set parameters to calibrate for the main result. In this section, we briefly show some of the convergence results for our initial approach. For the horizontal eddy parameters, we chose log-normal priors, with $\mu = -2, \sigma = 1.5$ for diffusivity, and $\mu = -7.5, \sigma = 1.5$ for the viscosity. We used the same uniform distribution prior $U(2 \leq x \leq 4)$ for the Smagorinsky parameter.

In Fig. S1, we show evolution of the prior, likelihood and posterior distributions as a function of EMCEE sampler iteration number. After about 30 iterations, the rate of convergence appears to slow down significantly, suggesting that significant improvements to the posterior distributions are unlikely. In Fig. S2, we show the evolution of the Markov chains, and can observe from the behavior of the 5%-95% percentile limits that the algorithm seems to diverge, indicating that the sampler find the parameters practically indistinguishable for predictive purposes. A possible explanation is that the stochasticity of the particle filtering algorithm influences the posterior much more strongly than the parameter choice. The posterior distributions are shown in Fig. S3, with the eddy parameters posteriors shown on the logarithmic scale. In contrast to their already wide priors, the two distributions provide an even larger spread of possible values, and are unlikely to provide posteriors of any practical use. It is possible however that a better designed error model for the particle filtering algorithm could assist the sampler in finding the best eddy parameters.

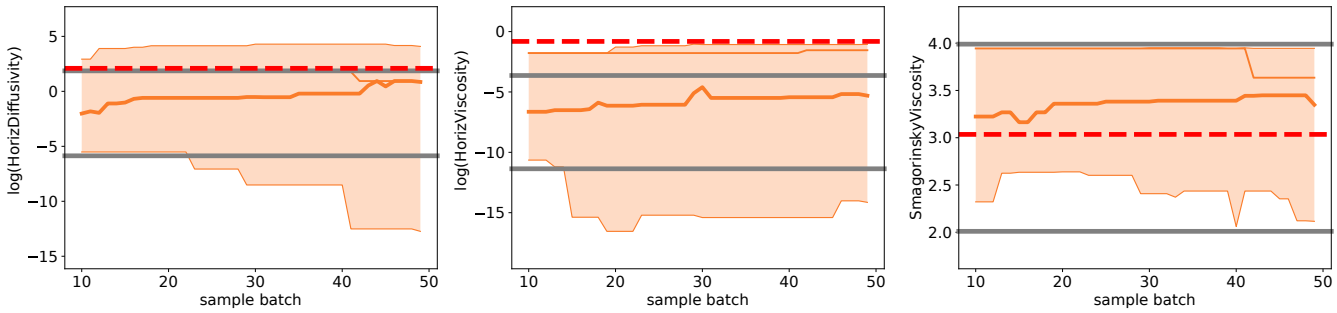

Figure S2. Markov chain parameters samples. The solid lines indicate the median and the semi-transparent spreads indicate the 5% - 95% percentiles across multiple concurrent chains of the sampler. The horizontal black lines indicate the bounds of the prior distribution. The red dashed line indicates the best found parameters values.

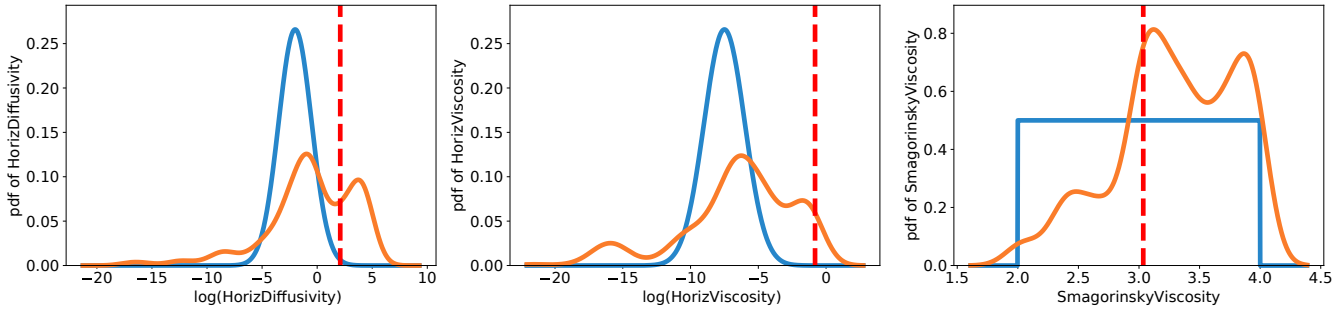

Figure S3. Marginal posterior (orange) and prior (blue) distributions of model parameters. The red dashed line indicates the best found parameters values.

4 Time-dependent parameters for the hydrodynamic model

To better account for the seasonality in the hydrodynamic model, two parameters were made time-dependent. First, the Secchi disk depth was based on the photosynthetically active radiation (PAR) measurements from the LéXPLORE platform, which were taken at multiple depths. Using only the measurements obtained between 9 AM and 6 PM, we estimated the attenuation parameter, μ, of the Beer-Lambert formula, $I(z) = I(0)\exp(-\mu z)$, where $I(z)$ is the intensity of light at depth z. To reduce the amount of noise in the data, only daily averages were used and physically unrealistic values were dropped. Finally we estimate the Secchi disk depth using $S = 1.7/\mu$, and supplied the value directly to the hydrodynamic model. Days with missing measurements were filled in by linearly interpolating nearest data points. In Fig. S4, we show the estimated values of the Secchi disk depth for 2019. In addition, we also use the albedo formula of Cogley Cogley (1979) to account for seasonal changes in surface reflectivity, shown in Fig. S5.

5 On using short timeframe for model calibrations

The main result of the paper relies on a calibration using an 11 month period of data from 2019. This requires a large amount of computational time, and therefore originally we sought a faster approach. To speed up the process of calibration for the Dalton number and Smagorinsky viscosity, we attempted to use a 2 month subset, July 1 - August 31, 2019. Using this approach, we computed 125 batches for the EMCEE sampler, with the same parallel configuration as in the long-term study.

In Fig. S6, we show the means and 5%-95% percentiles for the likelihood and posteriors of the Markov chain parameters, with relative stationarity obtained after about 70 iterations. The evolution of parameters, shown in Fig. S7, is also relatively unchanging after 70 iterations. Thus, eliminating the first 70 iterations as the burn-in period, the parameter posteriors are shown in Fig. S8. We can observe that the maximum aposteriori parameter (MAP) for the Dalton number is 0.0599, and exactly at the upper bound of the prior, indicating that a wider prior should have been used. At the same time, a comparison to the posteriors from the main study clearly demonstrate that a study using only summer months tend to provide an over-estimation of the

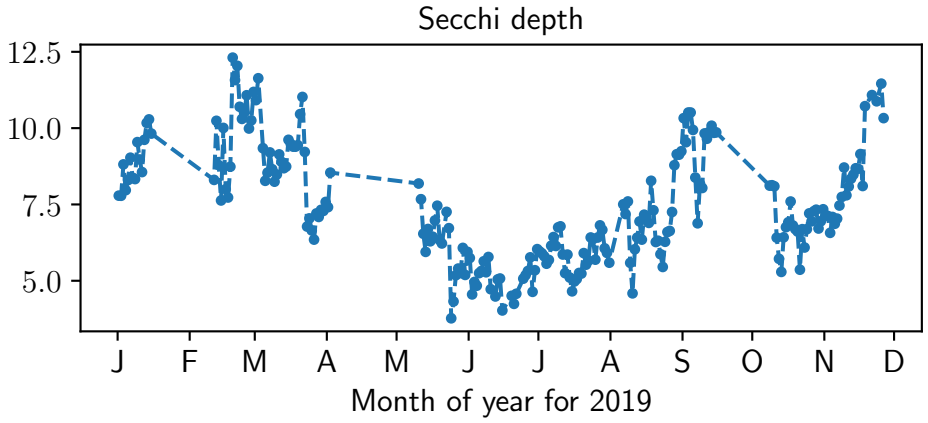

Figure S4. Secchi disk depth values for 2019 estimated from PAR data. The dashed lines indicate filled-in values using linear interpolation.

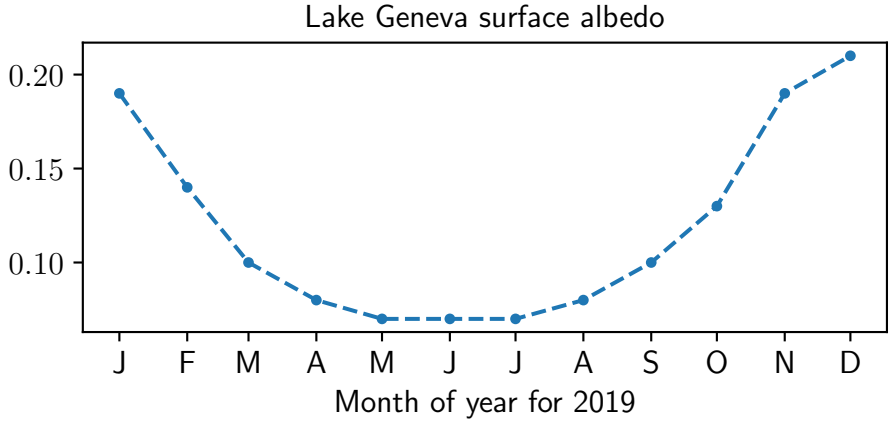

Figure S5. Surface albedo for Lake Geneva.

Dalton number. Therefore, we concluded that this 2-month calibration approach ultimately was not beneficial for long-term data assimilation models.

References

Cogley, J. G.: The Albedo of Water as a Function of Latitude, Monthly Weather Review, 107, 775–781, https://doi.org/10.1175/1520-0493(1979)107<0775:TAOWAA>2.0.CO;2, 1979.

Swiss Data Science Center: RENKU, https://renkulab.io/, 2021.

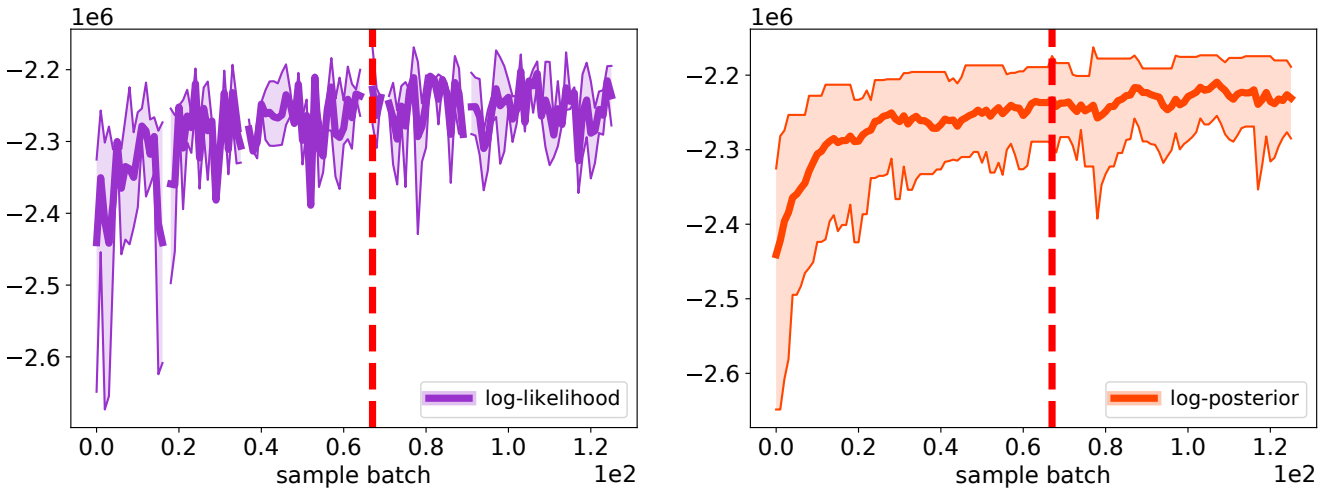

Figure S6. Log-likelihoods and log-posteriors for the sampled model posterior parameters (accepted only). The solid lines indicate the mean and the semi-transparent spreads indicate the 5% - 95% percentiles across multiple concurrent chains of the sampler. The red dashed line indicates the posterior estimate at the best found parameters.

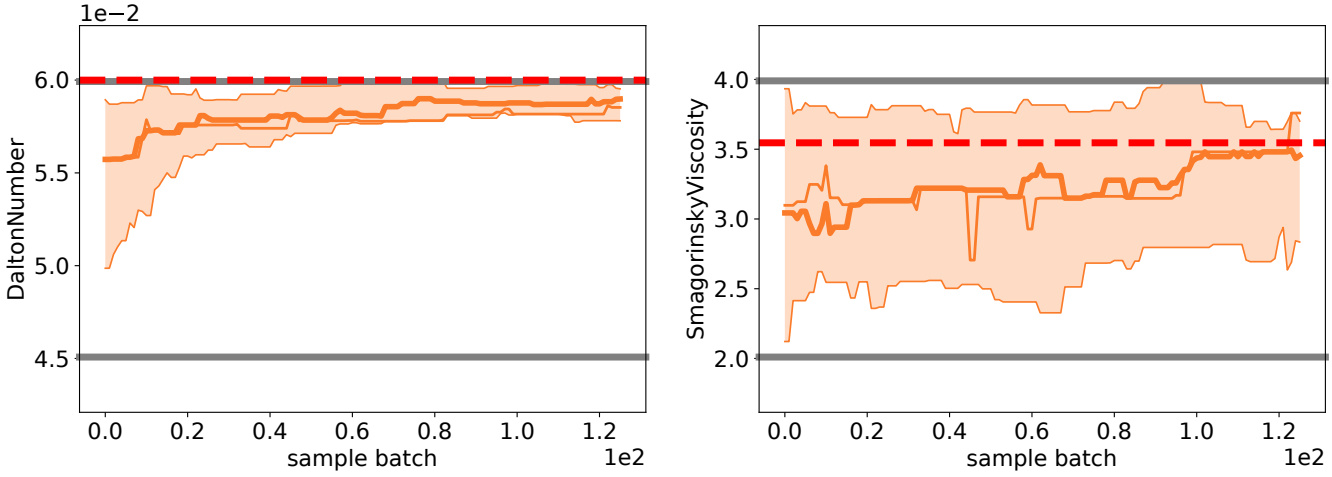

Figure S7. Markov chain parameters samples. The solid lines indicate the median and the semi-transparent spreads indicate the 5% - 95% percentiles across multiple concurrent chains of the sampler. The horizontal black lines indicate the bounds of the prior distribution. The red dashed line indicates the best found parameters values.

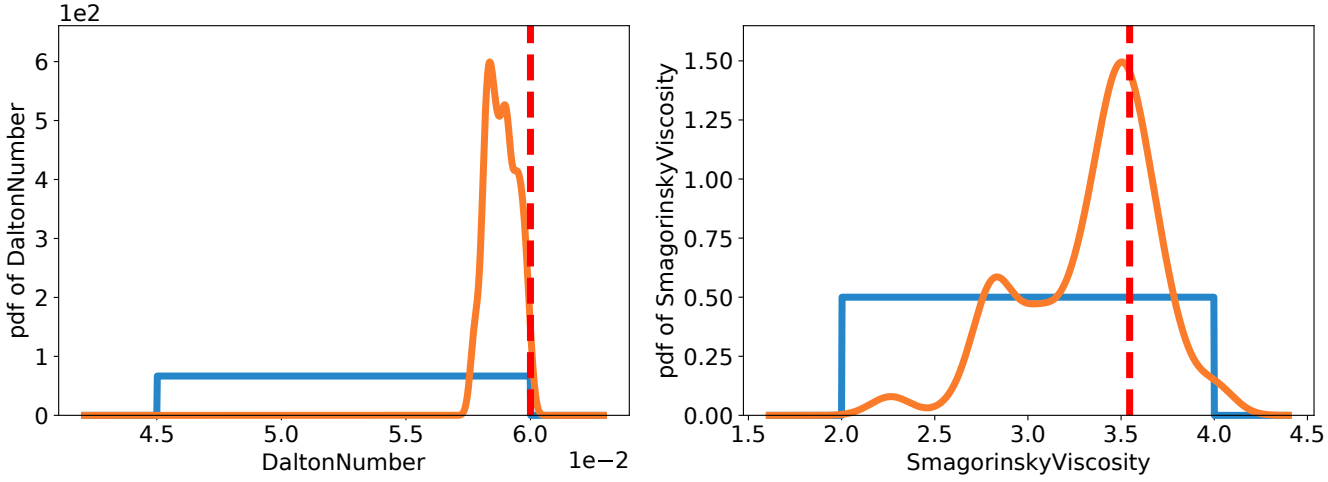

Figure S8. Markov chain parameters samples. The solid lines indicate the median and the semi-transparent spreads indicate the 5% - 95% percentiles across multiple concurrent chains of the sampler. The horizontal black lines indicate the bounds of the prior distribution. The red dashed line indicates the best found parameters values.