

1 General description

This document describes the set of program codes that were used for the analysis in the study, “Fast Riemannian-manifold Hamiltonian Monte Carlo for hierarchical Gaussian-process models.” The program codes for the proposed method were written in C++ and OpenACC languages. Since a custom-made GPU-accelerated code in OpenACC cannot be made scalable and platform-independent, as pointed out in literature [3, 1, 2], users need to adjust internal parameters and use suitable compiler options so that the program is correctly compiled and run in their environments. In our environment with workstations equipped with recent Intel Xeon processors, 256GB main memory, and Tesla A100 80GB GPU cards, we compiled the program code, for example, as follows:

```
1 mpic++ -I /usr/local/include -DHAVE_INLINE -D GSL_RANGE_CHECK_OFF -mmodel=medium -acc -ta=
   tesla:cc80 -Mcuda -std=c++17 -Minfo=accel -c bayes_gp.cpp
2 mpic++ -L /usr/local/lib -mmodel=medium -acc -ta=tesla:cc80 -Mcuda -std=c++17 -Minfo=accel
   bayes_gp.o -o bayes_gp.out -lpthread -lcurl -lcublas -lcusolver -lcudart -ldl -lcublas
   -lgsl -lgslcblas -lm
```

. The other program codes were compiled in the same manner. As one can see in the above, the program codes use libraries from NVHPC (ver.23.1, NVIDIA) and Gnu Scientific Library (ver2.7.1).

The program codes for the implementation based on Hamiltorch and PyTorch were also written. The program can be run simply by calling from python.

2 Description of program codes

2.1 HMCs for simulated data

The program code, `bayes_gp.cpp`, generates simulated data and then performs the posterior sampling for Bayesian logistic regression on the data.

```
1 mpirun -n 1 ./bayes_gp.out n_kernel hmc_seed data_seed A C eps_lf max_jacobi_sweep tol_jacobi
   par_sabs log_fname label_fname data_fname
```

The above command generates data $\{X_i^{(\text{exp})}\}_{1 \leq i \leq 500}$ and label $\{X_i^{(\text{label})}\}_{1 \leq i \leq 500}$ with a seed specified as `data_seed`, and save them in `data_fname` and `label_fname`, respectively. RMHMC with `A` blocks of `C` leapfrogs of stepsize `eps_lf` is performed with a seed specified as `hmc_seed` for its random-number generator. In this implementation, the eigendecomposition of the metric is dynamically programmed by taking advantage of the eigenvectors of the metric for the previously sampled parameter values. The eigendecomposition is performed by sweeping the elements of the Hessian of the log-posterior density (maximally `max_jacobi_sweep` times) with the Jacobi method until the Frobenius norm of the non-diagonal elements of a diagonalised matrix becomes smaller than `tol_jacobi`. In `log_fname`, the wall time spent on RMHMC, the total number of performed

leapfrogs, the logarithm of the joint posterior density, the acceptance rate in the Metropolis-Hastings procedure and the average number of sweeps for a block of C leapfrogs are written out in each line.

The following commands run program codes for static eigendecomposition (bayes_gp_syevd.out : the divide-and-conquer algorithm ; bayes_gp_coldsyevj.out: the Jacobi method).

```
1 mpirun -n 1 ./bayes_gp_syevd.out n_kernel hmc_seed data_seed A C eps_lf max_jacobi_sweep
    tol_jacobi par_sabs log_fname label_fname data_fname
```

```
1 mpirun -n 1 ./bayes_gp_coldsyevj.out n_kernel hmc_seed data_seed A C eps_lf max_jacobi_sweep
    tol_jacobi par_sabs log_fname label_fname data_fname
```

The argument variables and the output files are provided in the same manner as for bayes_gp.cpp.

The program codes based on Hamiltorch perform RMHMC or NUT-HMC on the data generated by bayes_gp.cpp with the following commands:

```
1 python3 ./bayes_gp_rhmc.py N D hmc_seed data_seed A C eps_lf label_fname data_fname
    log_fname
```

```
1 python3 ./bayes_gp_nuts.py N D hmc_seed data_seed A C eps_lf label_fname data_fname log_fname
```

These codes take the input files label_fname and data_fname generated by bayes_gp.cpp.

2.2 RMHMC for the calculation of BME

The following command runs a program code for writing out the $\ln P(X|a)$ value for a sample from P_τ conditioned on simulated data for each value of τ in lnP_fname file. As we described in Appendix B of the main text, running this code multiple times and integrating them with respect to τ yield the BME value. The values of parameters, d_tau (= 0.01) and Sigma ($\Sigma = 1.0$) must also be provided.

```
1 mpirun -n 1 ./bayes_gp_syevj_bme.out n_kernel hmc_seed data_seed A C eps_lf max_jacobi_sweep
    tol_jacobi par_sabs d_tau Sigma log_fname label_fname data_fname lnP_fname
```

The following command runs a program code for calculating the BME value based on the Laplace approximation conditioned on hyperparameter values (namely, INLA).

```
1 python3 ./bayes_gp_laplace.py N D Sigma tol_BFGS label_fname data_fname output_fname log_fname
```

This code takes the input files label_fname and data_fname generated by bayes_gp.cpp. In the output file output_fname, the values of $\hat{P}(\theta)$ and $|\hat{H}_{aa}(\theta)|$ for different values of θ are written out (see Appendix C of the main text). In the log file log_fname, the intermediate results in the optimisation are written out for debugging. The values of parameters, N (number of samples N), D (data dimensionality D), Sigma (prior variance of bias parameter Σ) and tol_BFGS (tolerance parameter for LBFGS) must also be provided.

The following commands run program codes for writing out the $\ln P(X|a)$ value for a sample from P_τ conditioned on NMES data (provided in `data_fname` (`nmes.df.tsv`)) for each value of τ in `lnP_fname` file. As we described in Appendix B of the main text, running this code multiple times and integrating them with respect to τ yield the BME value.

(Linear model for the conditional mean)

```
1 mpirun -n 1 ./bayes_gp_bmeLm.out n_kernel data_fname hmc_seed data_seed A C eps_lf
    max_jacobi_sweep tol_jacobi par_sabs sigma_0_sqr d_tau Sigma init_fname log_fname
    label_fname data_fname lnP_fname
```

(Linear model for the conditional mean and variance)

```
1 mpirun -n 1 ./bayes_gp_bmeLmv.out n_kernel data_fname hmc_seed data_seed A C eps_lf
    max_jacobi_sweep tol_jacobi par_sabs sigma_0_sqr d_tau Sigma init_fname log_fname
    label_fname data_fname lnP_fname
```

(Nonlinear model for the conditional mean)

```
1 mpirun -n 1 ./bayes_gp_bmeNLm.out n_kernel data_fname hmc_seed data_seed A C eps_lf
    max_jacobi_sweep tol_jacobi par_sabs sigma_0_sqr d_tau Sigma init_fname log_fname
    label_fname data_fname lnP_fname
```

(Nonlinear model for the conditional mean and variance)

```
1 mpirun -n 1 ./bayes_gp_bmeNLmv.out n_kernel data_fname hmc_seed data_seed A C eps_lf
    max_jacobi_sweep tol_jacobi par_sabs sigma_0_sqr d_tau Sigma init_fname log_fname
    label_fname data_fname lnP_fname
```

In addition to the argument variables for `bayes_gp_syevj_bme.cpp`, the parameter σ_0^2 in Eq.(2) of the main text is specified as `sigma_0_sqr`. The initial condition for the above MC integration must be calculated and provided as `init_fname`.

The following commands run program codes for calculating the BME value based on the Laplace approximation conditioned on hyperparameter values (namely, INLA).

(Linear model for the conditional mean)

```
1 mpirun -n 1 ./bayes_gp_bmeLm.out N D data_fname output_fname log_fname
```

(Linear model for the conditional mean and variance)

```
1 mpirun -n 1 ./bayes_gp_bmeLmv.out N D data_fname output_fname log_fname
```

(Nonlinear model for the conditional mean)

```
1 mpirun -n 1 ./bayes_gp_bmeNLm.out N D data_fname output_fname log_fname
```

(Nonlinear model for the conditional mean and variance)

```
1 mpirun -n 1 ./bayes_gp_bmeNLmv.out N D data_fname output_fname log_fname
```

These codes take the input file `data_fname` (NMES dataset `nmes_df.tsv`). In the output file `output_fname`, the values of $\hat{P}(\theta)$ and $|\hat{H}_{aa}(\theta)|$ for different values of θ are written out (see Appendix C of the main text). In the log file `log_fname`, the intermediate results in the optimisation are written out for debugging. The parameters, `N` (number of samples $N = 9708$), `D` (data dimensionality $D = 21$), `Sigma` (prior variance of bias parameter $\Sigma = 1.0$) and `tol_BFGS` (tolerance parameter for LBFGS) must also be provided.

References

- [1] CHANDRASEKARAN, S., AND JUCKELAND, G. *OpenACC for Programmers: Concepts and Strategies*. Addison-Wesley Professional, 2017.
- [2] FARBER, R. *Parallel programming with OpenACC*. Newnes, 2016.
- [3] REINDERS, J., ASHBAUGH, B., BRODMAN, J., KINSNER, M., PENNYCOOK, J., AND TIAN, X. *Data parallel C++: mastering DPC++ for programming of heterogeneous systems using C++ and SYCL*. Springer Nature, 2021.