UNNECESSARILY COMPLICATED RESEARCH TITLE

{ JOHN SMITH, JAMES SMITH AND JANE SMITH } UNIVERSITY AND DEPARTMENT NAME

OBJECTIVES

Probabilistic programming language and Bayesian inference engine Stan supports general model specification and uses dynamic HMC for fully Bayesian analysis [1]. Torsten is a library of Stan functions that simplifies pharmacometric modeling and extends the range of models that may be implemented [3]. To improve the performance of Bayesian inference of population models, we designed a multilevel parallel scheme that combines a cross-chain warmup algorithm with within-chain parallelisation and demonstrated that this approach significantly improves large PKPD model simulation efficiency.

CROSS-CHAIN WARMUP

The standard practice of Stan is to perform a fixed number of warmup iterations. With this practice, the efficacy of the warmup is unknown *a priori* and often warmup is unncessarily long as user oversubscribe warmup iterations. The proposed warmup algorithm tries to avoid this by checking potential scale reduction coefficients ($\hat{R}$) and effective sample sizes (ESS) [4] . Specifically, for warmup we propose

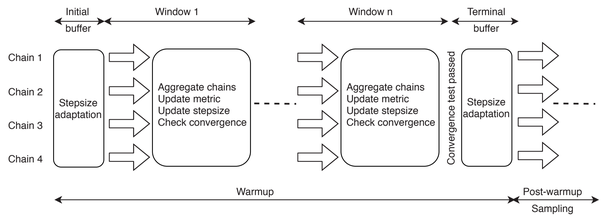


Figure 1: my caption of the figure

1. Given a fixed window size w(default 100 iterations), the sampler iterates during warmup with stepsize adapted as in regular warmup runs.
2. At the end of a window, joint posterior probability from all the chains are aggregated and used to calculate corresponding $\hat{R}$ and ESS. For example, with default window size $w = 100$, when warmup reaches iteration 300, we calculate $\hat{R}^i$ and ESS^i for $i = 1, 2, 3$, so that $\hat{R}^1$ and ESS^1 are based on warmup iteration 1 to 300, $\hat{R}^2$ and ESS^2 are based on warmup iteration 101 to 300, and $\hat{R}^3$ and ESS^3 are based on warmup iteration 201 to 300.
3. At the end of window n, with predefined target value $\hat{R}^0$ and ESS^0, from $1, \ldots, n$, we select j with maximum ESS^j, and a new metric is calculated by aggregating samples from corresponding windows. If, in addition, j satisfies $\hat{R}^j < \hat{R}^0$ and $\text{ESS}^j > \text{ESS}^0$, the warmup is considered complete(*converges*). Otherwise warmup continues until the end of the next window and step 2-3 are repeated.

Unlike current warmup scheme, the above proposal requires communication among the chains, hence we call it *cross-chain warmup*.
In benchmark, the above warmup scheme is compared against standard Stan warmup(1000 iterations) on several models for their

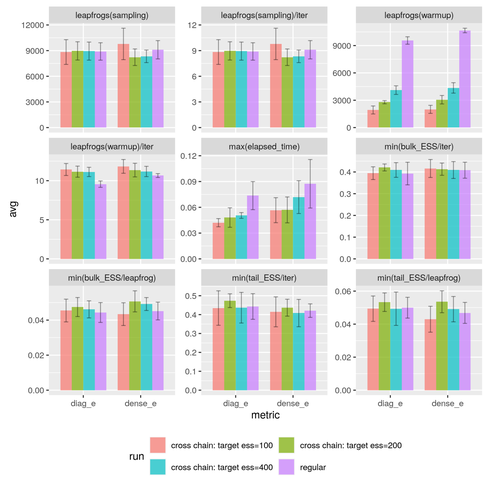
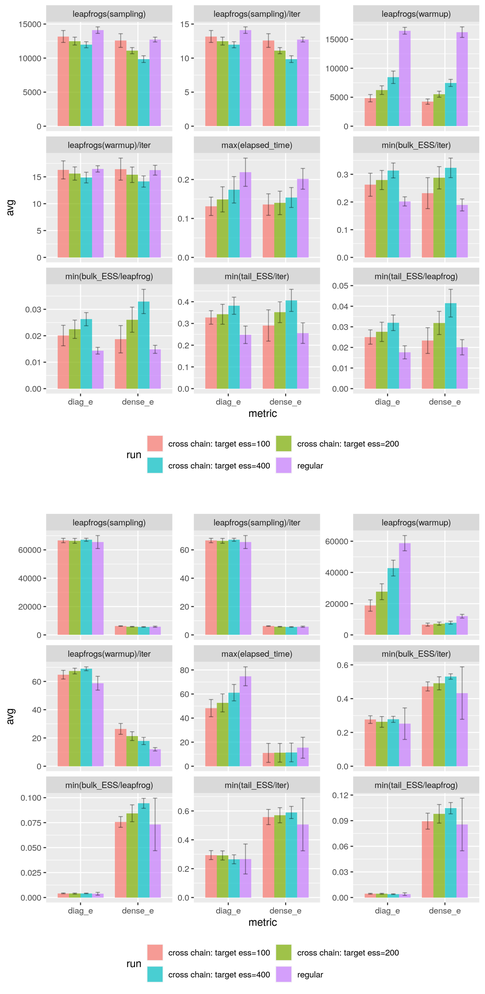


Figure 2: Cross-chain warmup performance comparison: eight schools model



total number of leapfrog integration steps in warmup	
total number of leapfrog integration steps in sampling	
number of leapfrog integration steps in per each warmup iteration	
number of leapfrog integration steps in per each sampling iteration	
minimum ESS_{bulk} per iteration	
minimum ESS_{tail} per iteration	
minimum ESS_{bulk} per leapfrog step	
minimum ESS_{tail} per leapfrog step	
maximum wall time height	

Each setup is run with 10 PRNG seeds and the quantities' average(barplot) and standard deviation(error bar) are shown. All wall time are in seconds.

MULTILEVEL PARALLELIZATION: CROSS-CHAIN WARMUP + WITHIN-CHAIN PARALLELIZATION

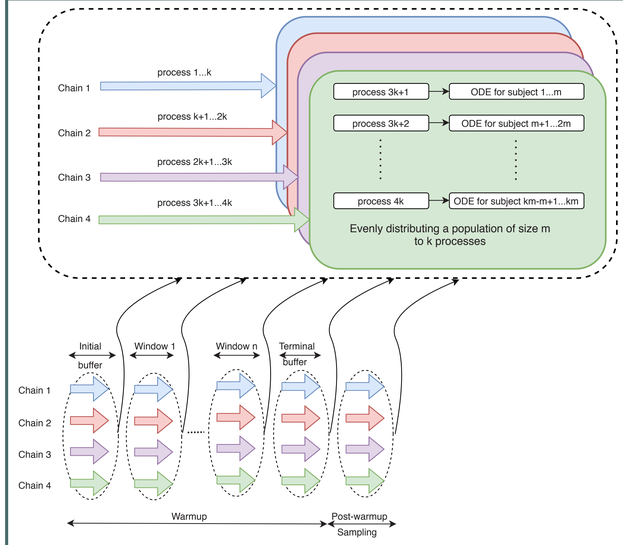


Figure 3: Multilevel parallelism for ODE-based population models. A simplified version of Figure 1, the lower diagram shows the cross-chain warmup through multiple windows. In within-chain parallelization, as shown in the upper diagram, each chain has its own parameter samples(indicated by different colors), and dedicated processes for solving the population model.

Level	Parallel operation	Parallel communication
1	parallel chains with cross-chain warmup	at the end each warmup window
2	within-chain parallel group ODE solver	when likelihhood is evaluated

Table 1: A framework of *multilevel parallism* for Bayesian inference of population models.

A time-to-event model for the time to the first grade 2+ peripheral neuropathy (PN) event in patients treated with an antibody-drug conjugate (ADC) delivering monomethyl auristatin E (MMAE). We call it Time-To-PN(TTPN) model, and analyze data using a simplified version of the model reported in [2]. We consider three treatment arms: fauxlatuzumab vedotin 1.2, 1.8 and 2.4 mg/kg IV boluses q3w x 6 doses, with 20 patients per treatment arm. In this model, each patient's PK is described by an effective compartment model(one-compartment), and PD by a linear model. The likelihood for time to first 2+ PN event is described by a hazard function that depends on the concentration effect through Weibull distribution. Two unknowns from PK model and the cumulative hazard form a three-component ODE system. Each evaluation of likelihood requires solving this 3-system for every patient.
In Torsten's model, ODEs corresponding to the entire population can be solved by a single call of `pmx_solve_group_rk45` function. The three parameters of the model are:

- k_{e0} in effective compartment model.
- α the coefficient of linear PD model.
- β Weibull distribution scale parameter.

Similar to previous section, Figure shows performance of cross-chain and regular runs based on target ESS = 400. Unlike in previous models, we did not performe runs with multiple seed or target ESS to avoid long computing time. One can make conclusion consistent with the other models, that the cross-chain warmup reduce total run time without compromising ESS.

	Cross-chain	Regular
`leapfrogs(warmup)`	1.002225e+04	1.588275e+04
`leapfrogs(sampling)`	1.709250e+04	1.831600e+04
`leapfrogs(warmup)/iter`	1.822227e+01	1.588275e+01
`leapfrogs(sampling)/iter`	1.709250e+01	1.831600e+01
`min(bulk_ESS/iter)`	2.805000e-01	2.340000e-01
`min(tail_ESS/iter)`	3.482500e-01	3.205000e-01
`min(bulk_ESS/leapfrog)`	1.641071e-02	1.277572e-02
`min(tail_ESS/leapfrog)`	2.037443e-02	1.749836e-02
`max(elapsed_time)`	1.702630e+03	1.979646e+03

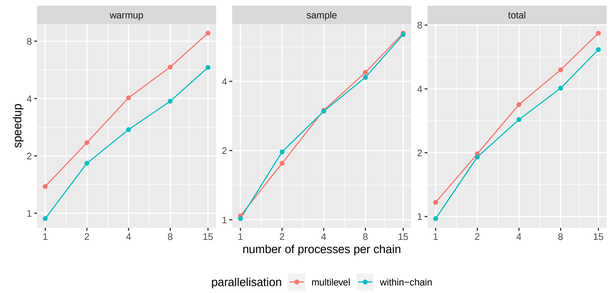


Figure 4: Multilevel scheme parallel performance based on TTPN model.

Next, we apply multilevel method to TTPN model with a fixed target ESS = 400, by running the model with 4 chains using $n_{\text{proc}} = 8, 16, 32, 60, 80$ processes. Equivalently, there are $n_{\text{proc\_per\_chain}} = 2, 4, 8, 15, 20$ processes per chain so that within-chain parallelization can be utilized. With population size 60, each process handles solution of $n_{\text{id}} = 30, 15, 7, 4, 3$ subjects' ODE system, respectively.
To show parallel scaling performance, we collect `stanfit` objects of the benchmark runs and plot their wall time speedup against regular Stan runs. With all runs having 1000 post-warmup sampling iterations, in multilevel runs the number of warmup iterations is determinted at runtime, while both within-chain parallel runs and regular Stan runs have 1000 warmup iterations. Among 4 chains in a run, we use the one with maximum total walltime(in seconds) as performance measure, as in practice usually further model evaluation becomes accessible only after all chains finish.
As shown in Figure **??**, both multilevel and within-chai-only parallel runs exhibit good scaling up to 60 processes(15 processes per chain $\times$ 4 chains)

CONCLUSION AND FUTURE WORK

Multilevel parallelism using in Stan and Torsten significantly improves computational efficiency and extends the range of models that may be practically implemented. A natual follow-up of this study is to seek higher efficiency by maintaining target ESS while increasing the number of parallel chains during warmup.

REFERENCES

[1] B. CARPENTER, A. GELMAN, M. D. HOFFMAN, D. LEE, B. GOODRICH, M. BETANCOURT, M. BRUBAKER, J. GUO, P. LI, AND A. RIDDELL, *Stan: A Probabilistic Programming Language*, Journal of Statistical Software, 76 (2017), pp. 1–32.

[2] D. LU, W. R. GILLESPIE, S. GIRISH, P. AGARWAL, C. LI, J. HIRATA, Y.-W. CHU, M. KAGEDAL, L. LEON, V. MAIYA, AND J. Y. JIN, *Time-to-Event Analysis of Polatuzumab Vedotin-Induced Peripheral Neuropathy to Assist in the Comparison of Clinical Dosing Regimens*, CPT: pharmacometrics & systems pharmacology, 6 (2017), pp. 401–408.

[3] TORSTEN DEVELOPMENT TEAM, *Torsten: library of C++ functions that support applications of Stan in Pharmacometrics*. https://github.com/metrumresearchgroup/Torsten.

[4] A. VEHTARI, A. GELMAN, D. SIMPSON, B. CARPENTER, AND P.-C. BÜRKNER, *Rank-normalization, folding, and localization: An improved $\hat{R}$ for assessing convergence of MCMC*, arXiv:1903.08008 [stat], (2019). arXiv: 1903.08008.