Torsten: Stan functions for pharmacometric applications

New functionality including within chain parallel computation

Yi Zhang, William R. Gillespie
Metrum Research Group

Introduction

Stan is a widely used, open-source, probabilistic programming language and Bayesian inference engine [3, 4]. It provides a general model specification language and uses HMC simulation for fully Bayesian data analysis. Torsten is a library of Stan functions that simplifies implementation of pharmacometric (PMX) models and extends the range of models that may be implemented [2]. The objective of the work presented here is to improve and extend Torsten. This includes the addition of new functions and enhancements to existing ones.

Currently Torsten provides following functions for single-subject PMX modeling.

```
/* Functions for single-subject pharmacometric analysis */
matrix pmx_solve_onecpt(...)      /* 1-cpt model with 1st order absorption */
matrix pmx_solve_twocpt(...)      /* 2-cpt model with 1st order absorption */
matrix pmx_solve_linode(...)      /* linear ODE model specified through coefficient matrix */
matrix pmx_solve_adams(...)       /* general ODE model solved using Adams-Moulton scheme */
matrix pmx_solve_bdf(...)         /* general ODE model solved using BDF scheme */
matrix pmx_solve_rk45(...)        /* general ODE model solved using Runge-Kutta scheme */
matrix pmx_solve_onecpt_bdf(...)  /* general ODE model coupled with 1-cpt model */
matrix pmx_solve_onecpt_rk45(...) /* general ODE model coupled with 2-cpt model */
matrix pmx_solve_twocpt_bdf(...)  /* general ODE model coupled with 1-cpt model */
matrix pmx_solve_twocpt_rk45(...) /* general ODE model coupled with 2-cpt model */
```

Torsten's ODE integrators

To facilitate design for within-chain parallelization, Torsten v0.86 added its own implementation of ODE integrators

```
real[,] pmx_integrate_ode_adams(...);
real[,] pmx_integrate_ode_bdf(...);
real[,] pmx_integrate_ode_rk45(...);
```

based on the same backend libraries(`0deint` & `CVODES` [5]) as Stan's, and they share the same signature. Additionally, Torsten's ODE integrators allow solution times `ts` to be parameters. Table 1-3 show performance comparison of the integrators based on two ODE systems(Torsten v0.86, Stan v2.19.1)

integrator	run 1	run 2	run 3
`integrate_ode_bdf`	1.41	1.47	1.42
`pmx_integrate_ode_bdf`	1.16	1.18	1.15

Table 1: wall time(s) of solving a chemical kinetics ODE using BDF integrators

integrator	run 1	run 2	run 3
`integrate_ode_adams`	2.09	2.11	2.07
`pmx_integrate_ode_adams`	1.65	1.66	1.64

Table 2: wall time(s) of solving a chemical kinetics ODE using Adams integrators

integrator	run 1	run 2	run 3
`integrate_ode_adams`	8.12	8.75	8.22
`pmx_integrate_ode_adams`	7.25	7.92	7.24

Table 3: wall time(s) of solving Lorenz system ODE using Adams integrators

Torsten's within-chain parallel integrators and solvers

Torsten now includes versions of Stans ordinary differential equation (ODE) solvers that have been revised to improve performance and provide MPI-based parallel computing, We have implemented new population functions that calculate the model states for a group of individuals in a single function call. Those functions also employ MPI to distribute those calculations over multiple processors, thus providing efficient within chain parallel computation.

Torsten provides an alternative to Stan's `map_rect` framework that avoids use of `boost.mpi` and serialization, simplifies user programming tasks by providing a simplified function signature and automating computation load distribution. Unlike `map_rect`, Torsten's parallel computing capability is available through functions for specific heavy-lifting tasks. Currently we focus on tasks involving numerical solution of ODEs.

ODE group integrators

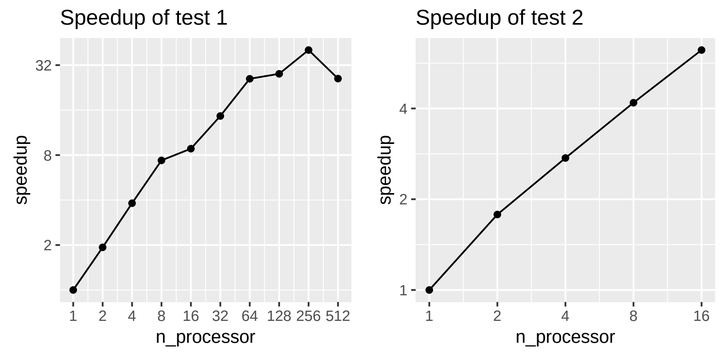
The ODE group integrators solve a set of systems governed by a *same* ODE but with different parameters/data. Function arguments `y0`, `theta`, `x_r`, and `x_i` are 2D arrays with 1st dimension indicating subject within the group. `ts` is a ragged array for each subject's solution time, and `len` consists of length of each subject's record in `ts`.

```
matrix pmx_integrate_ode_group_adams(...);
matrix pmx_integrate_ode_group_bdf(...);
matrix pmx_integrate_ode_group_rk45(...);

/* all three functions share same signature */
matrix pmx_integrate_ode_group_rk45(ODE_system, real[,] y0, real t0, int[] len, real[] ts, real[,] theta, real[,]
  ↪  x_r, int[,] x_i, real rtol, real atol, int max_step);
```

We demonstrate the MPI performance of the group integrators using two tests.

- Test 1: Neutropenia PKPD ODE group of size 1000 solved using `BDF` group integrator. Each ODE is of size 8 and has 9 parameters.
- Test 2: predator-prey model parameter inference for a set of 16 Lotka-Volterra equations, with the ODE group solved using `RK45` integrator.



Pharmacometric population solvers

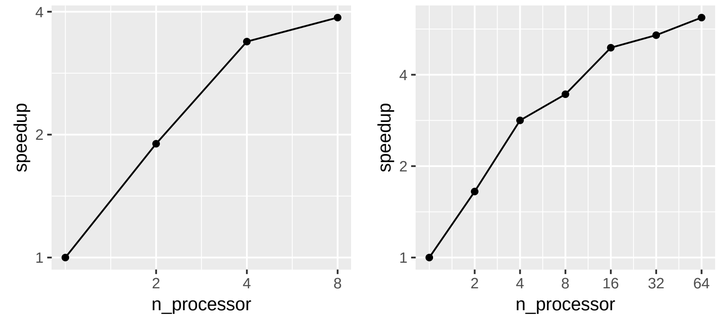
```
matrix pmx_solve_group_adams(...);
matrix pmx_solve_group_bdf(...);
matrix pmx_solve_group_rk45(...);

/* all three functions share same signature */
matrix pmx_solve_group_rk45(ODE_system, int nCmt, int[] len, real[] time, real[] amt, real[] rate, real[] ii, int[]
  ↪  evid, int[] cmt, real[] addl, int[] ss, real[] theta, real[] biovar, real[] tlag, real rel_tol, real abs_tol,
  ↪  int max_step);
```

Similar to single-subject solvers, the group/population PMX solvers use a model specification and data format based on NONMEM®/NMTRAN/PREDPP conventions. Function arguments specify dosing events for a population using ragged arrays, with each subject's data located through array `len` that contains the subject record length.

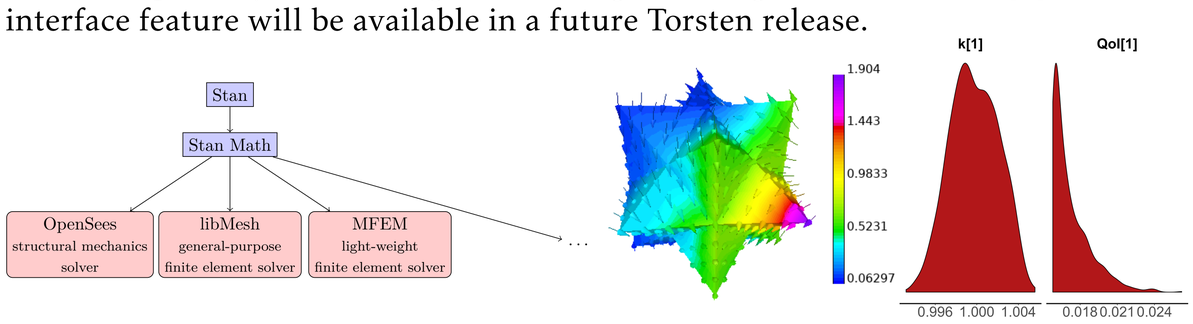
We demonstrate the MPI performance of the population PMX solvers using two tests.

- Test 3: Two-compartment model with first-order absorption among a population of 8. Stan model parameters are the PK model parameters for each subject. The population PK model is solved numerical using `pmx_solve_group_bdf`.
- Test 4: Parametric time-to-event model for the time to the first grade 2+ peripheral neuropathy event in 60 patients treated with an antibody-drug conjugate delivering monomethyl auristatin E. The hazard ODE system is of size 3 and has 5 parameters.



Interface to external PDE libraries

We have also implemented interface to external PDE libraries for applications that involve PDE parameter inference. The methodology has been tested using multiple PDE libraries. Example on the right shows Darcy's flow velocity in an inference model for porosity k of in irregular flow region, using `Torsten` + `MFEM` [1]. The PDE interface feature will be available in a future Torsten release.



Conclusions and future work

Recent developments in Stan and Torsten significantly improve computational efficiency and extend the range of models that may be implemented. The addition of within chain parallel computation to Stan/Torsten makes fully Bayesian analysis with Stan an increasingly practical option for PMX applications.

Some near-future work planned for Torsten:

- Further improve MPI performance.
- Multi-level and hybrid parallel computing based on MPI and multithreading.

References

[1] MFEM: Modular finite element methods library. `mfem.org`.

[2] Torsten: library of C++ functions that support applications of Stan in Pharmacometrics. `https://github.com/metrumresearchgroup/Torsten`.

[3] Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A Probabilistic Programming Language. *Journal of Statistical Software*, 76(1):1–32, January 2017.

[4] Bob Carpenter, Matthew D. Hoffman, Marcus Brubaker, Daniel Lee, Peter Li, and Michael Betancourt. The Stan Math Library: Reverse-Mode Automatic Differentiation in C++. *arXiv:1509.07164 [cs]*, September 2015. arXiv: 1509.07164.

[5] Alan C. Hindmarsh, Peter N. Brown, Keith E. Grant, Steven L. Lee, Radu Serban, Dan E. Shumaker, and Carol S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software*, 31(3):363–396, September 2005.

yiz@metrumrg.com billg@metrumrg.com

The authors would like to thank Office of Naval Research for funding this research under grant N6833518C0110.