Gaining Efficiency by Combining Analytical and Numerical Methods to Solve ODEs: Implementation in Stan and Application to Bayesian PK/PD Modeling

Charles C. Margossian[1] and William R. Gillespie[2]

[1]Department of Statistics, Columbia University, New York, NY (formally Metrum Research Group)

[2]Metrum Research Group, Tariffville, CT

mc-stan.org

Introduction

Objective: Fitting ODE based models is computationally expensive, especially when we do a full bayesian inference. Often times, an ODE system contains a subsystem we can solve analytically; for instance, the PK may consist of a simple linear two compartment model. The *mixed solver* combines this analytical solution with a numerical integrator to solve the full system. We implement the method in the open-source Bayesian inference software Stan [1]. Our goal is to measure how well the mixed solver performs compared to a regular numerical integrator in Stan.

Method: We fit simulated PK and neutrophil data to a Friberg-Karlsson semi-mechanistic model [2] using both the full numerical and the mixed solvers. For each case, we fit the models to simulated data sets 100 times, and measure how accurately Stan estimates the parameters and the time required to produce 1000 effective independent samples.

Results: Both methods do comparably well when estimating parameters. We however observe an average speedup of $49 \pm 14\%$ with the mixed solver. This is in good agreement with our theoretical calculations. While mix solving is more efficient, it requires a greater coding effort.

Software Implementation: To make mix solving more readily available, new functions are coded in C++ and added to Torsten [3] a pharmacometrics extension for Stan.

Motivating Problem

The Friberg-Karlsson model describes the absorption of a drug which as a side-effect causes neutropenia. The PK ODES describe a linear two compartment model:

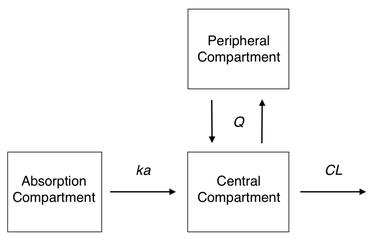


Figure 1: Linear Two Compartment Model

$$\begin{aligned} y'_{\text{gut}} &= -k_a\, y_{\text{gut}} \\ y'_{\text{cent}} &= k_a\, y_{\text{gut}} - \left(\frac{CL}{V_{\text{cent}}} + \frac{Q}{V_{\text{cent}}}\right) y_{\text{cent}} + \frac{Q}{V_{\text{peri}}}\, y_{\text{peri}} \\ y'_{\text{peri}} &= \frac{Q}{V_{\text{cent}}}\, y_{\text{cent}} - \frac{Q}{V_{\text{peri}}}\, y_{\text{peri}} \end{aligned} \tag{1}$$

This system is relatively simple and has a closed form solution.

The PD ODEs describe how the drug perturbs the feedback mechanism that keeps the neutrophil count at a baseline.

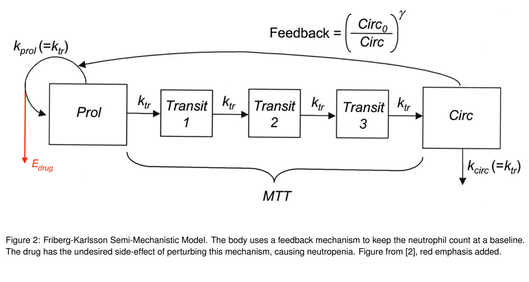


Figure 2: Friberg-Karlsson Semi-Mechanistic Model. The body uses a feedback mechanism to keep the neutrophil count at a baseline. The drug has the undesired side-effect of perturbing this mechanism, causing neutropenia. Figure from [2], red emphasis added.

$$\begin{aligned} y'_{\text{prol}} &= k_{\text{tr}}\, y_{\text{prol}} (1 - E_{\text{drug}}) \left(\frac{Circ_0}{y_{\text{circ}}}\right)^{\gamma} - k_{\text{tr}}\, y_{\text{prol}} \\ y'_{\text{tr1}} &= k_{\text{tr}}\, y_{\text{prol}} - k_{\text{tr}}\, y_{\text{tr1}} \\ y'_{\text{tr2}} &= k_{\text{tr}}\, y_{\text{tr1}} - k_{\text{tr}}\, y_{\text{tr2}} \\ y'_{\text{tr3}} &= k_{\text{tr}}\, y_{\text{tr2}} - k_{\text{tr}}\, y_{\text{tr3}} \\ y'_{\text{circ}} &= k_{\text{tr}}\, y_{\text{tr3}} - k_{\text{tr}}\, y_{\text{circ}} \end{aligned} \tag{2}$$

These ODEs are nonlinear and need to be solved numerically. Note the dependency on the PK ODEs, through $E_{\text{drug}} = \alpha\, y_{\text{cent}} / V_{\text{cent}}$.

This produces a system of 8 nonlinear ODEs to solve. If we wish to use a gradient-based algorithm to fit our model, such as gradient-descent Maximum Likelihood Estimation or Hamiltonian Monte Carlo sampling [4], we also need to compute sensitivities. In particular, we must calculate the Jacobian of the solution with respect to the initial states, y_0, and the model parameters, θ:

$$\theta = \{CL, Q, V_{\text{cent}}, V_{\text{peri}}, k_a, MTT, Circ_0, \gamma, \alpha\}$$

To calculate the Jacobian, we use automatic differentiation, a method which has proven more efficient than finite and symbolic differentiation [5], and has been implemented in Stan [6].

Mix Solving

The above ODE system has the form:

$$\begin{aligned} y'_{\text{PK}} &= f_{\text{PK}}(y_{\text{PK}}, t) \\ y'_{\text{PD}} &= f_{\text{PD}}(y_{\text{PK}}, y_{\text{PD}}, t) \end{aligned}$$

When doing a full integration, we integrate the coupled system:

$$y(t) = \int f(f_{\text{PK}}, f_{\text{PD}}) \mathrm{d}t$$

When mix solving, we solve for y_{PK} analytically and only integrate:

$$y_{\text{PD}}(t) = \int f(y_{\text{PK}}, f_{\text{PD}}) \mathrm{d}t$$

Mix solving presents some trade-offs:

- We reduce the number of states we integrate numerically. Our theoretical calculations show the run time ratio between the mixed solver and the full integrator is then $\mathscr{R}_{\text{theory}} = 0.42$.
- These calculations do however not account for the fact the integrant is more complex, because y_{PK} is often a much more sophisticated expression than f_{PK}. Thus $\mathscr{R}_{\text{theory}}$ should be treated as a lower bound.
- Coding a mixed solver is more difficult. We need to deal with some overhead (hand-coding the analytical solution and a lot of bookkeeping). The latter gets taken care of by Torsten under the hood.

Simulation Study Method

We simulate plasma concentration and neutrophil count for one patient using *mrgsolve* [7], according to the parameter values:

$$\begin{aligned} (CL, Q, V_{\text{cent}}, V_{\text{peri}}, ka) &= (10, 15, 35, 105, 2.0) \qquad \sigma^2 = 0.001 \\ (MTT, Circ_0, \alpha, \gamma) &= (125, 5, 3 \times 10^{-4}, 0.17) \quad \sigma^2_{\text{PD}} = 0.001 \end{aligned}$$

The following protocol describes the clinical trial our hypothetical patient undergoes:

- 1 subject
- Multiple doses: 80,000 μg administered every 12 hours, 15 times
- PK: plasma concentration of drug (c)
- PD response: Neutrophil count (y_{circ})
- PK measured 0.083, 0.167, 0.5, 0.5, 0.75, 1, 2, 3, 4, 6, and 8 hours after the first, second and last doses, and once every 12 hours.
- PD measured once every two days for 28 days

We use strongly informative priors. While this makes for a less realistic setting, it insures the Markov chains are exploring the target posterior, and reduces variance during the run time of the warm-up phase:

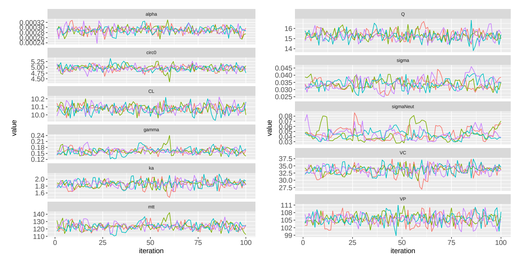
$$\begin{aligned} CL &\sim \text{lognormal}(10, 0.1) & Q &\sim \text{lognormal}(15, 0.2) & V_{\text{cent}} &\sim \text{lognormal}(35, 0.14) \\ V_{\text{peri}} &\sim \text{lognormal}(105, 0.14) & ka &\sim \text{lognormal}(2, 0.17) & Circ0 &\sim \text{lognormal}(5, 0.2) \\ MTT &\sim \text{lognormal}(125, 0.2) & \gamma &\sim \text{lognormal}(0.17, 0.2) & \alpha &\sim \text{lognormal}(2 \times 10^{-4}, 0.2) \end{aligned}$$

Simulation Study Results

We write two models in Stan: one using full integration and the other using a mixed solver. Both methods use a Runge Kutta $4^{\text{th}}/5^{\text{th}}$ order integrator.

We first write unit tests in C++ and find the solvers agree within a relative error of 10^{-6}. They are also in agreement with *mrgsolve*. The Jacobians the two solvers produce are in agreement within a relative error of 7×10^{-5}. Since the full integrator was fully tested before its implementation in Stan, we consider it to be reliable.

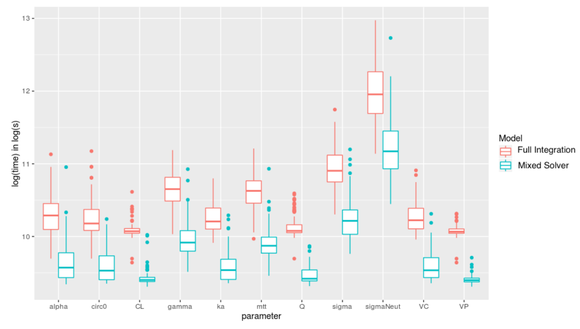
For each model, we run 100 fits. Each fit uses 4 Markov chains with 200 iterations, equally divided between the warm-up and sampling phases. Convergence is assessed by looking at the trace plots, the $\hat{R}$ statistics, and the effective number of samples. We here show example trace plots from a fit obtained when using the mixed solver:



	mean	n_eff	Rhat		mean	n_eff	Rhat
CL	10.08	393.32	1.01	Circ0	4.96	274.71	1.01
Q	15.31	339.76	1.01	MTT	123.55	400	0.99
VC	33.58	263.58	1.00	gamma	0.162	204.45	1.01
VP	105.42	400	1.00	alpha	2.90e-4	134.45	1.01
ka	1.86	285.31	1.00	sigmaNeut	0.044	99.78	1.02
sigma	0.034	244.92	1.00				

The $\hat{R}$ statistics is always below 1.1 which is a good indicator the chains are converging to a common distribution.

The estimated posteriors is approximately the same for both models across all fits. We however observe differences in the computation speed, when looking at the time required to produce 1000 effective independent samples:



The mixed solver model is faster. Averaging over all runs and parameters, the ratio for time to compute 1000 independent samples between the two models is:

$$\mathscr{R} = 51.11 \pm 13.51\%$$

This is above our theoretical lower bound and gives us some sense of how expensive computing y_{PK} analytically at each step the integrator takes is. Code used for the simulated study can be found at `https://github.com/charlesm93/MixedSolver`.

Implementation in Torsten

Stan is a very flexible open source probabilistic programming language designed primarily to perform Bayesian data analysis (`mc-stan.org`). Torsten is a **prototype** extension of Stan for pharmacometrics. It provides new built-in functions which can be used directly in the Stan language. Torsten includes compartment PK/PD models and schedules of discrete events, e.g. dosing.

As of version 0.83, Torsten has a module to do mix solving, where y_{PK} can be:

- A linear One Compartment Model with a first-order absorption
- A linear Two Compartment Model with a first-order absorption

The user no longer needs to hand-code analytical solutions and the overhead coding is now taken care of under the hood. A call to the function may look as follow:

```
x = mixOde2CptModel_rk45(f_PD, nOde_PD,
                         time, amt, rate, ii, evid, cmt, addl, ss,
                         theta, F, tlag,
                         rel_tol, abs_tol, max_num_steps)
```

where:

- `x`: the drug mass in each compartment at each event
- `f_PD`: a function that encodes the PD ODEs
- `nOde_PD`: the number of PD ODEs
- `time, amt, rate, ...`: the event schedule
- `theta, F, tlag`: the model parameters
- `rel_tol, abs_tol, max_num_steps`: the tuning parameters for the ODE integrator.

In particular, `f_PD` is coded as follow:

```
real [] f_PD (real t, real [] y, real[] yPK, real [] theta, real [] x_r, real [] x_i) {

. . .

  Edrug = alpha * yPK[2] / VC;

  dydt[1] = ktr * prol * ((1 - Edrug) * ((circ0 / circ)^gamma) - 1);
  dydt[2] = ktr * (prol - transit1);
  dydt[3] = ktr * (transit1 - transit2);
  dydt[4] = ktr * (transit2 - transit3);
  dydt[5] = ktr * (transit3 - circ);

  return dydt;
}
```

For more details, please consult our manual at: `https://github.com/metrumresearchgroup/example-models`.

References

[1] Stan Development Team. *Stan Modeling Language Users Guide and Reference Manual*. http://mc-stan.org, version 2.16.0 edition (2017).

[2] Friberg, L.E., Henningsson, A., Maas, H., Nguyen, L. and Karlsson, M.O. Model of chemotherapy-induced myelosuppression with parameter consistency across drugs. *J Clin Oncol* **20** (2002):4713–21.

[3] Margossian, C.C. and Gillespie, W.R. Stan functions for pharmacometrics modeling. In *Journal of Pharmacokinetics and Pharmacodynamics*, volume 43 (2016).

[4] Neal, R.M. *MCMC using Hamiltonian Dynamics*. Handbook of Markov Chain Monte Carlo (Chapman & Hall / CRC Press, 2010).

[5] Baydin, A.G., Pearlmutter, B.A., Radul, A.A. and M., S.J. Automatic differentiation in machine learning: a survey. *arXiv:1502.05767v2* (2015).

[6] Carpenter, B., Hoffman, M.D., Brubaker, M.A., Lee, D., Li, P. and Betancourt, M.J. The stan math library: Reverse-mode automatic differentiation in c++. *arXiv 1509.07164*. (2015).

[7] Baron, K.T., Hindmarsh, A.C., Petzold, L.R., Gillespie, B., Margossian, C. and Pastoor, D. *mrgsolve: Simulate from ODE-Based Population PK/PD and Systems Pharmacology Models*. Metrum Research Group, http://mrgsolve.github.io/user_guide/ (2017). R package version 0.8.6.

 Copies of this poster are available at www.metrumrg.com/publications

ACoP 8th Annual Conference, Fort Lauderdale, Florida, October 13–15, 2017