

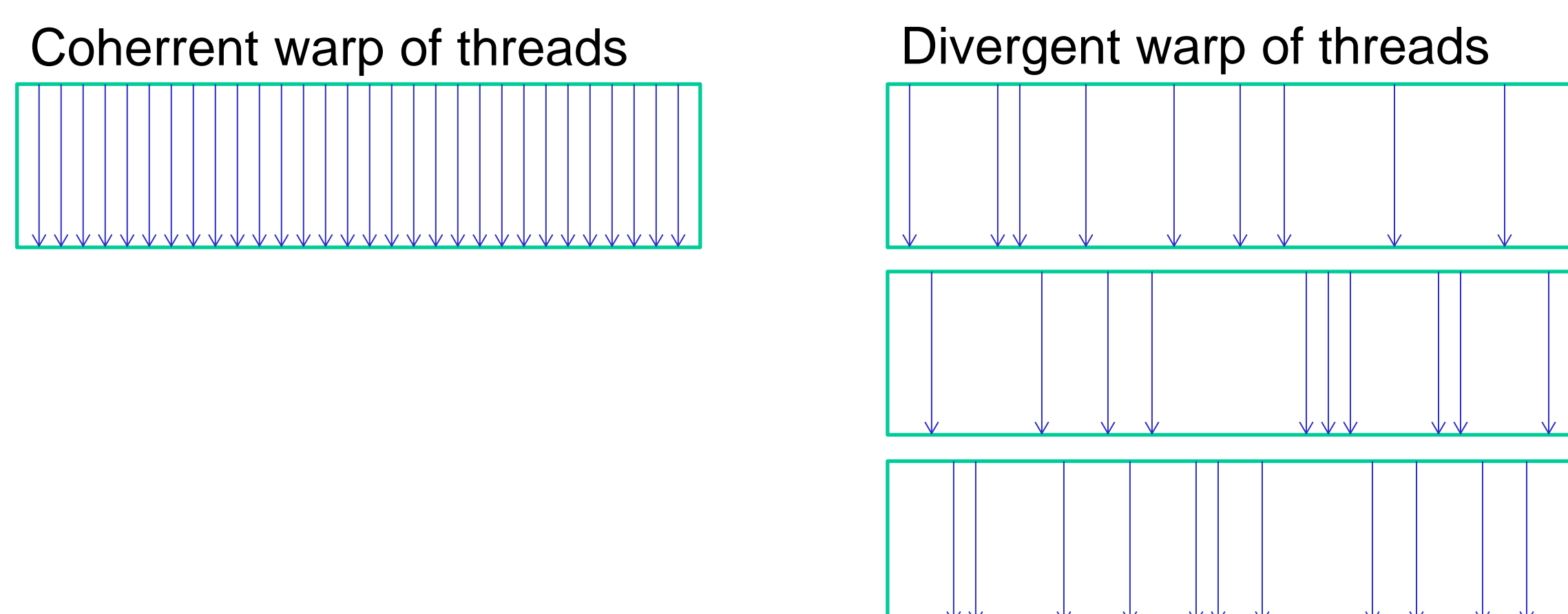
## Objectives

Recent advances in the general computing capabilities of readily available graphics processing units (GPUs) make them appropriate tools for the solution of nonlinear mixed effects problems. The architecture of the GPU presents challenges for their general use for population pharmacokinetic / pharmacodynamic (popPK/PD) applications. The solution of ordinary differential equation (ODE) models can cause the execution paths of the concurrently executing threads to diverge, destroying the computational efficiency and obliterating the benefits of the GPU. Thread divergence is aggravated by adjustable step-size ODE solvers and different event times (observations, doses, time-lags, reflux, etc...) between subjects. In this work we demonstrate an implementation that circumvents these issues and lays the groundwork for professional application of GPUs to popPK/PD problems.

## Methods

To derive suitable benefit from the GPU, the number of individuals in objective function evaluation should be much greater than the number of cores on the GPU. Sampling based algorithms (e.g., IMPEM, MCPEM, QRPEM) are well suited to massively parallel computation because each sample can be treated as an independent subject. With the typical number of samples per subject being quite high (e.g., on the order of 1000), the total number of samples is large even for small problems.

Computations on the GPU proceed in blocks of threads that are in turn divided into "warps" of 32. Each warp receives the same instruction, but multiple warps and blocks can be interleaved to keep the processors busy while longer operations (e.g., memory read/write, computation of math functions like exp). If the threads within a warp take different paths at a branching statement (e.g., if, while, for, do types of conditionals and loops) then the warp is essentially split and much of the parallelism on the GPU is lost.



A novel version of the Burlisch-Stoer stiff solver (routine stifbs, Numerical Recipes in C) was implemented as a GPU kernel in using the CUDA C programming language (Nvidia Corporation). To integrate individual sample predictions, we group each subject's samples into separate thread blocks to ensure that the dose and observation times are the same within a block. To overcome step-size differences between samples, we treat the samples within a block as one large block diagonal system. This ensures that all samples in a block integrate together, albeit at the rate of the slowest sample. Each thread computes the derivative and Jacobian functions for its assigned sample, as needed, while the entire block of threads is used to solve the resulting linear system of equations.

The entire process of generating individual predictions is run on the GPU, with the CPU handling the parameter estimation process with a Quasi-Random Parametric Expectation Maximization (QRPEM) algorithm. An OpenMP implementation of the same ODE solver on the CPU was used for comparison.

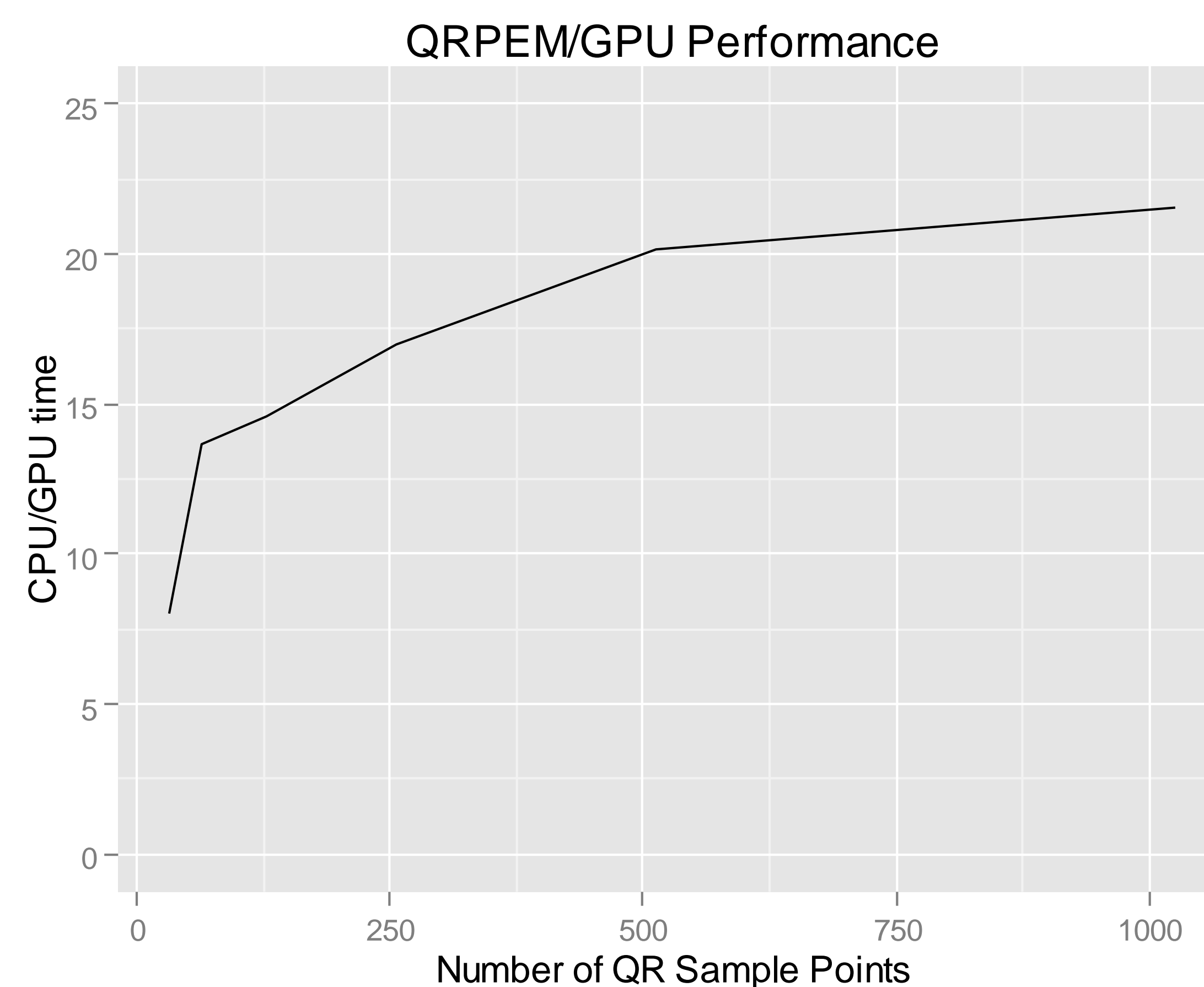
The test problem was a single dose, intravenous bolus, saturating clearance, simulated dataset, generated from the following model:

$$\begin{aligned} \frac{dA_a}{dt} &= -k_a A_a & C &= \frac{A_1}{V} \\ \frac{dA_1}{dt} &= k_a A_a - \frac{v_{\max} C}{K_m + C} & C_{obs} &= C(1 + \varepsilon) \end{aligned}$$

The problem was run with various numbers of samples to test the scaling of the GPU and CPU implementations.

## Results

The GPU implementation was between 5 and 20 times faster than the CPU implementation baselined to a single core. The CPU and GPU both exhibited linear scaling with number of subjects for large numbers of samples, but the GPU showed superlinear scaling with small numbers of samples. That is, the runtime for the GPU does not increase proportionally with sample size until its computational capacity becomes saturated.



## Discussion

Previous work had shown tremendous performance gains using QRPEM on GPU with closed form models. This implementation of a stiff ODE solver on the GPU illustrates the feasibility of using this type of computing hardware for non-linear-mixed effects modeling for a much more general class of models. There are still some factors, however, that may limit its application.

The GPU consists of several Streaming Multiprocessor Units (SMUs), each of which have multiple cores. In the case of the GTX 580 used here there are 16 SMUs with 32 cores each, for a total of 512 cores. Thread blocks are assigned to the SMUs, where they must divide the shared memory. In addition, there are a limited number of registers available to the running threads. The register and shared memory (totaling 64kb) are faster than the GPU global memory, but overuse of either can limit the total number of active thread blocks that can run.

There are complex tradeoffs to be made in order to optimize memory performance vs. occupancy of active blocks. Memory usage increases with the number of equations, so for large models it may not be possible to make much use of the shared memory. But that loss of performance in memory transactions may be counterbalanced by increasing occupancy (more blocks can be active). The current code is heavily weighted towards utilizing shared memory, which may be hurting performance. Additional work could be done to further optimize the routines and achieve a higher occupancy (currently 8.3%).

The results here show a roughly 20 fold increase in performance between our GPU and a single CPU. It can be argued that an 8 core machine would lower the benefit substantially and that the GPU is no match for a cluster of such machines. It should be noted, however, that multiple GPUs can be hosted inside modest CPUs (2 core) raising the performance of a single machine to roughly 60-80 fold faster than a single core. These machines can also be used in a cluster, so the GPU maintains a performance edge.

Computing the objective function for popPK/PD problems on the GPU is a feasible and possibly worthwhile opportunity. For particularly large and/or computationally intensive problems the cost of custom coding could result in tremendous time savings. However, the specialized coding skills and specific hardware required to utilize the GPU present a significant barrier to all but the most determined pharmacometricians. Therefore, commercially viable GPU solutions for popPK/PD are needed to make these benefits generally accessible to the pharmacometrics community.

## Contact

Jason Chittenden, Certara/Pharsight Corporation,  
Ph: +1 919.852.4630, Email: jason.chittenden@certara.com