RPEM: Randomized Monte Carlo parametric expectation maximization algorithm

Rong Chen\textsuperscript{1,2} | Alan Schumitzky\textsuperscript{2,3} | Alona Kryshchenko\textsuperscript{4} | Keith Nieforth\textsuperscript{1} | Michael Tomashevskiy\textsuperscript{1} | Shuhua Hu\textsuperscript{1} | Romain Garreau\textsuperscript{5,6} | Julian Otalvaro\textsuperscript{2} | Walter Yamada\textsuperscript{2} | Michael N. Neely\textsuperscript{2,7,*}

\textsuperscript{1}Certara, Inc., Princeton, New Jersey, USA
\textsuperscript{2}Laboratory of Applied Pharmacokinetics and Bioinformatics, Children’s Hospital Los Angeles, Keck School of Medicine, University of Southern California, Los Angeles, California, USA
\textsuperscript{3}Department of Mathematics, University of Southern California, Los Angeles, California, USA
\textsuperscript{4}Department of Mathematics, California State University Channel Islands, Camarillo, California, USA
\textsuperscript{5}UMR CNRS 5558, Laboratoire de Biométrie et Biologie Evolutive, Université de Lyon, Université Lyon 1, Villeurbanne, France
\textsuperscript{6}Hospices Civils de Lyon, GH Nord, Service de Pharmacie, Lyon, France
\textsuperscript{7}Pediatric Infectious Diseases, Children’s Hospital Los Angeles, Keck School of Medicine, University of Southern California, Los Angeles, California, USA

Abstract
Inspired from quantum Monte Carlo, by sampling discrete and continuous variables at the same time using the Metropolis–Hastings algorithm, we present a novel, fast, and accurate high performance Monte Carlo Parametric Expectation Maximization (MCPEM) algorithm. We named it Randomized Parametric Expectation Maximization (RPEM). We compared RPEM with NONMEM’s Importance Sampling Method (IMP), Monolix’s Stochastic Approximation Expectation Maximization (SAEM), and Certara’s Quasi-Random Parametric Expectation Maximization (QRPEM) for a realistic two-compartment voriconazole model with ordinary differential equations using simulated data. We show that RPEM is as fast and as accurate as the algorithms IMP, QRPEM, and SAEM for the voriconazole model in reconstructing the population parameters, for the normal and log-normal cases.

Study Highlights
WHAT IS THE CURRENT KNOWLEDGE ON THE TOPIC?
NONMEM, Phoenix NLME, and Monolix are among the most widely used software packages which contain Monte Carlo parametric expectation maximization (MCPEM) algorithm engines. All of them tend to apply the importance sampling functions designed for the expectation step (E-step) to the maximization step (M-step), and use biased Monte Carlo estimators in the M-step. This may influence the speed and accuracy when dealing with certain models and datasets.

WHAT QUESTION DID THIS STUDY ADDRESS?
Is there a general, simple, and efficient way to use unbiased estimators in both the E-step and the M-step, and possibly further improve the speed and the accuracy of the MCPEM algorithms?

Correspondence
Rong Chen, Certara, Inc., Princeton, NJ 08540, USA.
Email: rong.chen@certara.com

*Principal Investigator: Michael N. Neely (mneely@chla.usc.edu).
WHAT DOES THIS STUDY ADD TO OUR KNOWLEDGE?

Randomized Parametric Expectation Maximization (RPEM) provides a novel Metropolis-Hastings algorithm inspired from quantum Monte Carlo. In the M-step of the MCPEM algorithm, by treating the discrete and continuous variables the same, we make both the E-step and the M-step use unbiased estimators. For normal and log-normal distributed parameters, we observe speed and/or accuracy improvement of RPEM over other MCPEM engines, such as Importance Sampling Method (IMP), Quasi-Random Parametric Expectation Maximization (QRPEM), and Stochastic Approximation Expectation Maximization (SAEM), for a voriconazole model which contains three ordinary differential equations, and with rich sampling data.

HOW MIGHT THIS CHANGE DRUG DISCOVERY, DEVELOPMENT, AND/OR THERAPEUTICS?

RPEM may be an alternative and/or additional solution to the current MCPEM algorithms. Especially when for certain reasons one must use a truncated Gaussian distribution, yet still want reasonably accurate estimations of the population parameters as well as the predictions.

INTRODUCTION

The expectation maximization (EM) method\(^1\) is a widely used powerful algorithm in machine learning, including population modeling of pharmacokinetic (PK) and pharmacodynamic (PD) systems. EM treats data in terms of “complete” and “missing.” By using Bayes Theorem, the missing data are integrated out, and the parameters of the parametric model are learned automatically through iterations between the expectation steps (E-steps) and the maximization steps (M-steps). An EM algorithm based, exact maximum likelihood solution to the parametric population modeling problem was proposed by Schumitzky\(^2\) in 1995 and fully implemented by Walker\(^3\) in 1996 for non-mixture models. For mixture models, the corresponding formulas were derived in ref. \([4]\).

In general, the EM algorithm needs to address two important problems. One is from the theoretical aspect, namely the convergence problem. EM algorithms converge to a stationary point\(^4-6\) of the likelihood function from the given initial conditions. However, a stationary point can be a local maximum, a local minimum, or a saddle point, and therefore it may not be the global maximum of the likelihood function. A standard method to increase the probability of finding the global maximum is to repeatedly run the algorithm, initializing each run with different conditions. This, of course, is computationally intense, so developing efficient methods to avoid converging on solutions which are not truly maximally likely is an important task.

We focus in this paper on the second problem which faces EM algorithms from the computational aspect, namely how to evaluate all the integrals efficiently and accurately in the E-step and the M-step. This is important because the integrals must be evaluated at each iteration. If the integrals can be evaluated efficiently, we can reach the stationary point rapidly, which will enable us to search from more initial conditions per unit time, and therefore also help addressing the first problem of convergence described above. Parametric EM methods typically use Monte Carlo algorithms to evaluate the integrals, and such methods are called Monte Carlo parametric EM methods (MCPEM). Therefore, fast, accurate and robust Monte Carlo algorithms are always desirable for MCPEM.

There are successful and widely used Monte Carlo engines implemented in software tools commonly used for PK/PD population modeling, such as those used in ADAPT from the University of Southern California\(^7\), S-ADAPT\(^8\), Importance sampling method (IMP) in ICON’s NONMEM program\(^9\), Quasi-random parametric EM (QRPEM) in Certara’s Phoenix software\(^10\), and R speaks Non-Linear Mixed Effects (RsNLME)\(^11,12\) and Stochastic Approximation EM (SAEM)\(^13-16\) in Lixoft’s Monolix software. As far as we know, each of these MCPEM algorithms strictly sums over the Monte Carlo integrals for all the \(n\) subjects, and, when applicable, for all the possible \(K\) components of a mixture model to describe the joint model parameter value probability distributions as sums of normal distributions.

Inspired from quantum Monte Carlo methods\(^17-21\), which is one of the most precise, reliable, and successful computational methods widely used in physics and chemistry, in this work, we approach the Monte Carlo integrals in a novel way by developing our randomized parametric expectation maximization (RPEM) algorithm. It uses an unbiased Metropolis–Hastings method to efficiently sample from the subjects and mixture models...
instead of strictly summing over all of them. The unique Metropolis–Hastings method in the M-step allows us to efficiently get unbiased estimations for both analytic models and models with ordinary differential equations (ODEs) without taking many samples in the E-step, which is time-consuming. Therefore, as we will show, RPEM is fast without compromising accuracy.

The paper is organized as follows. In “Methods” section, we briefly introduce the two-stage nonlinear random effects mixture model and the EM algorithm. Then, we describe the algorithm of RPEM. In “Results and Discussions” section, we report the results of RPEM by using a concrete example of the two-stage nonlinear random effects two-mixture model as well as a realistic voriconazole model with ODEs. We compare RPEM with IMP, QRPEM and SAEM, and show that RPEM is not only fast but also accurate to varied starting conditions, no matter the population parameters are normal or log-normally distributed. We also report how RPEM scales on the supercomputer. In “Summary and Outlook” section, we summarize the novelty, speed, and accuracy of RPEM. Possible future directions are considered in closing.

**METHODS**

**Two-stage nonlinear random effects mixture model**

We consider a two-stage nonlinear mixture model. At stage one, given $\theta_i$, which is the parameter vector describing the unobserved random effects ($\theta_i \in \mathbb{R}^p$), as well as $\beta$ which describes the unobserved fixed effects ($\beta \in \mathbb{R}^q$), then the $m_i$-dimensional observation vector for the $i$th individual $Y_i = (Y_{i1}, \ldots, Y_{im_i})^T$ is sampled from a Gaussian distribution such that:

$$Y_i | \theta_i, \beta \sim N(h_i(\theta_i), G_i(\theta_i, \beta)), \quad i = 1, \ldots, n. \quad (1)$$

where $n$ represents the number of subjects, $N(\mu, \Sigma)$ is the multivariate Gaussian distribution with mean vector $\mu$ and the covariance matrix $\Sigma$, $h_i(\theta_i)$ is the function defining the PK/PD model, and $G_i(\theta_i, \beta)$ is a positive definite covariance matrix ($G_i \in \mathbb{R}^{m_i \times m_i}$). In this paper, we consider the following important case:

$$G_i(\theta_i, \beta) = \sigma^2 H_i(\theta_i), \quad (2)$$

where $H_i(\theta_i)$ is a known function and $\beta = \sigma^2$.

In our usage, a random effects parameter can change from subject to subject; whereas a fixed effects parameter is constant over the population.

At stage two, each of the $n$ parameter vectors $\theta_1, \ldots, \theta_n$ is sampled from a Gaussian distribution with $K$ mixing components:

$$\theta_1, \ldots, \theta_n \sim_{iid} \sum_{k=1}^{K} w^{(k)} N(\mu^{(k)}, \Sigma^{(k)}), \quad (3)$$

where the non-negative number $w^{(k)}$ (normalized by $\sum_{k=1}^{K} w^{(k)} = 1$) is the weight for the $k$th Gaussian distribution $N(\mu^{(k)}, \Sigma^{(k)})$, $\mu^{(k)}$ is the mean vector ($\mu^{(k)} \in \mathbb{R}^p$) and $\Sigma^{(k)}$ is the positive definite covariance matrix ($\Sigma^{(k)} \in \mathbb{R}^{p \times p}$).

Given the observation $\{Y_1, \ldots, Y_n\}$, we can define $p(Y_i | \theta_i, \beta)$ as the likelihood function of $Y_i$ given $\theta_i$ and $\beta$.

We use $p(\theta_i | \mu^{(k)}, \Sigma^{(k)})$ to denote the multivariate Gaussian distribution of $\theta_i$ with mean vector $\mu^{(k)}$ and covariance matrix $\Sigma^{(k)}$. We want to estimate $\phi$ which represents the collection of parameters $\{\beta, \mu^{(k)}, \Sigma^{(k)}, \theta_i, k = 1, \ldots, K\}$ by maximizing the overall data likelihood $L(\phi)$ which is written as:

$$L(\phi) = \prod_{i=1}^{n} \sum_{k=1}^{K} p(Y_i | \theta_i, \beta) w^{(k)} p(\theta_i | \mu^{(k)}, \Sigma^{(k)}) d\theta_i. \quad (4)$$

This is called the maximum likelihood estimate (MLE). The MLE of $\phi$ is defined as $\phi_{ML}$ such that $L(\phi_{ML}) \geq L(\phi)$ for all $\phi$ within the parameter space.

We want to point out that the notation used in this paper is for mathematical convenience. They should not be confused with those widely used notations in NONMEM, Monolix, and RsNLME. For example, the mean of the Gaussian $\mu^{(k)}$, is usually called a “fixed-effect” parameter, and is referred to as “THETA” or “theta” in NONMEM, Monolix, and RsNLME. The variance of the Gaussian, $\Sigma^{(k)}$, is usually called a “random-effect” parameter which describes the (L1) random effects, and is referred to as “OMEGA” or “omega” in NONMEM and RsNLME. For Monolix, “OMEGA” or “omega” means the standard deviation of the Gaussian. In our notation, for subject $i$, the vector $\theta_i$ is the fixed effect plus the random effect $\eta_i$, that is, $\theta_i = \mu + \eta_i$, where $\eta_i$ is usually called “ETA” or “eta” in NONMEM, Monolix, and RsNLME.

**EM algorithm**

In the EM algorithm, we define complete data and missing data. In this model, we define $\{\theta_i, z_i\}$ as missing data, where $z_i$ is a $K$-dimensional vector whose $k$th component $z_i(k)$ is 1 or 0 depending on whether $\theta_i$ belongs to the $k$th mixing in Equation 3 or not. The complete data is defined as $Y_c = \{(Y_i, \theta_i, z_i), i = 1, \ldots, n\}$. The purpose of the EM algorithm is to start with $\phi^{(0)}$ and iterate from $\phi^{(r)}$ to $\phi^{(r+1)}$...
at the $r$th iteration, continue the process until we find the desired parameters $\phi^{(r+1)}$ such that $\phi^{(r+1)} = \arg\max_{\phi} Q(\phi, \phi^{(r)})$, where $Q(\phi, \phi^{(r)})$ is defined by Equation 5 and calculated by Equation 7. This process is guaranteed to converge to a stationary point of the likelihood, and, typically, a number of starting positions are suggested in an effort to ensure convergence to a global maximum.

In the E-step, the function $Q(\phi, \phi^{(r)})$ is defined as:

$$Q(\phi, \phi^{(r)}) = E \{ \log L_c(\phi) | Y, \phi^{(r)} \},$$

(5)

where the complete data likelihood $\log L_c(\phi)$ is given by:

$$\log L_c(\phi) = \sum_{i=1}^{n} \sum_{\theta_1} \log p \left( Y_i, \theta_1 | \sigma^2, \mu(\theta), \Sigma(\theta) \right) Z_i(\theta).$$

(6)

By using Bayes Theorem, Equation 5 can be written as:

$$Q(\phi, \phi^{(r)}) = \frac{\sum_{\theta_1} g_{i\theta}(\theta_1, \phi^{(r)}) \log p \left( Y_i, \theta_1 | \sigma^2, \mu(\theta), \Sigma(\theta) \right) d\theta_1}{\sum_{\theta_1} g_{i\theta}(\theta_1, \phi^{(r)})},$$

(7)

where

$$g_{i\theta}(\theta_1, \phi) = \frac{w(\theta_1)p(Y_i | \Sigma(\theta_1)) p(\theta_1 | \mu(\theta), \Sigma(\theta))}{\sum_{\theta_1} w(\theta_1)p(Y_i | \Sigma(\theta_1)) p(\theta_1 | \mu(\theta), \Sigma(\theta)) d\theta_1},$$

(8)

and

$$\log p \left( Y_i, \theta_1 | \sigma^2, \mu(\theta), \Sigma(\theta) \right) = C - \frac{1}{2} \log \left( \sigma^2 \right) - \frac{1}{2\sigma^2} (Y_i-h_1(\theta_1))^T H_1(\theta_1)^{-1} (Y_i-h_1(\theta_1))$$

$$- \frac{1}{2} \left( \theta_1 - \mu(\theta_1) \right)^T \Sigma(\theta_1)^{-1} \left( \theta_1 - \mu(\theta_1) \right) - \frac{1}{2} \log | \Sigma(\theta_1) |,$$

(9)

for some constant $C$. Note that the probability that the $i$th individual belongs to the $\theta$th mixing component can be defined as a function $r_i(\theta)$ such that:

$$r_i(\theta) = E \{ Z_i(\theta) | Y, \phi \} = p r \{ Z_i(\theta) = 1 | Y, \phi \} = \int g_{i\theta}(\theta_1, \phi) d\theta_1,$$

(10)

In the M-step, it is sufficient to find the unique solution of $\phi^{(r+1)}$ such that:

$$\frac{\partial}{\partial \phi} Q(\phi, \phi^{(r)}) \bigg|_{\phi^{(r+1)}} = 0,$$

(11)

where we define $\phi' = \{ \beta, \mu(\theta), \Sigma(\theta) \}; \theta = 1, \ldots, K \}$. Equation 11 leads to unique solutions of $\mu(\theta)^{(r+1)}$,

$$[\mu(\theta)^{(r+1)}, \sigma^2(\theta)]^{(r+1)}.$$ The updating of $w(\theta)$ can be calculated as the average of the contributions from each subject to the $\theta$th mixing, that is,

$$[w(\theta)]^{(r+1)} = \frac{1}{n} \sum_{i=1}^{n} r_i(\theta) = \frac{1}{n} \sum_{i=1}^{n} g_{i\theta}(\theta_1, \phi^{(r)}) d\theta_1.$$  (12)

The EM iterates $\phi^{(r)}$ have the important property that the corresponding likelihoods $L(\phi^{(r)})$ are non-decreasing, that is, $L(\phi^{(r+1)}) \geq L(\phi^{(r)})$ for all $r^{[6]}$. It is also worthwhile to mention that, under certain assumptions, in SAEM, by splitting the E-step into a simulation-step and a stochastic approximation step, or using an improved SAEM which adds a simulation-step before E-step and adds a stochastic approximation step after E-step, it is possible to show that the algorithm converges to a local maximum of the likelihood function. Besides SAEM, there are other algorithms using stochastic methods in order to converge to a local maximum, see recent references such as refs. [24-26].

**RPEM algorithm**

From a close look at $g_{i\theta}(\theta_1, \phi^{(r)})$ in Equation 8, we find that:

$$\sum_{\theta_1} g_{i\theta}(\theta_1, \phi^{(r)}) d\theta_1 = 1,$$

(13)

$$\sum_{i=1}^{n} \sum_{\theta_1} g_{i\theta}(\theta_1, \phi^{(r)}) d\theta_1 = n.$$ (14)

Therefore, the unique solutions of Equation 11 can be written as:

$$[\mu(\theta)]^{(r+1)} = \frac{\sum_{i=1}^{n} r_i(\theta_1, \phi^{(r)}) d\theta_1}{\sum_{\theta_1} g_{i\theta}(\theta_1, \phi^{(r)}) d\theta_1},$$

(15)

$$[\Sigma(\theta)]^{(r+1)} = \frac{\sum_{i=1}^{n} \left[ z_i - [\mu(\theta)]^{(r+1)} \right] \left[ z_i - [\mu(\theta)]^{(r+1)} \right]^T}{\sum_{i=1}^{n} m_i/n} g_{i\theta}(\theta_1, \phi^{(r)}) d\theta_1.$$ (16)

$$\left[\sigma^2(\theta)\right]^{(r+1)} = \frac{\sum_{i=1}^{n} \sum_{\theta_1} [Y_i-h_1(\theta_1)]^T H_1(\theta_1)^{-1} [Y_i-h_1(\theta_1)] g_{i\theta}(\theta_1, \phi^{(r)}) d\theta_1}{\sum_{i=1}^{n} \sum_{\theta_1} g_{i\theta}(\theta_1, \phi^{(r)}) d\theta_1}.$$ (17)

In a particularly useful case, we can partition the parameter $\theta_i$ into two components: $\theta_i = \{ \alpha, \xi \}$. Such that $\alpha$
is from a mixture of multivariate Gaussians and $\zeta_i$ is from one single multivariate Gaussian. In such a case, the EM updates from Equation 11 are given by:

$$\left[ \mu^{(\hat{\theta})}_i \right]^{(r+1)} = \frac{\sum_{i=1}^{n} f_i g_{i\ell} (\theta_i, \phi^{(r)}) d \theta_i}{\sum_{i=1}^{n} g_{i\ell} (\theta_i, \phi^{(r)}) d \theta_i},$$

(18)

$$\left[ \Sigma^{(\hat{\theta})}_i \right]^{(r+1)} = \frac{\sum_{i=1}^{n} \left\{ \alpha_i - \left[ \mu^{(\hat{\theta})}_i \right]^{(r+1)} \right\}^T g_{i\ell} (\theta_i, \phi^{(r)}) d \theta_i}{\sum_{i=1}^{n} g_{i\ell} (\theta_i, \phi^{(r)}) d \theta_i},$$

(19)

$$\left[ \Sigma^{(\hat{\theta})}_i \right]^{(r+1)} = \frac{\sum_{i=1}^{n} \sum_{k=1}^{K} \xi_i g_{i\ell} (\theta_i, \phi^{(r)}) d \theta_i}{\sum_{i=1}^{n} \sum_{k=1}^{K} g_{i\ell} (\theta_i, \phi^{(r)}) d \theta_i},$$

(20)

RPEM is a Monte Carlo parametric EM algorithm. The most important feature of RPEM is that it updates the parameters in Equations 15–21 by performing a Metropolis–Hastings algorithm based on an “overall” randomized sampling for both discrete labels, such as $i$ and $\ell$, and the continuous variables $\theta_j$. Next, we describe the basic idea of RPEM in the E-step and M-step.

**E-Step of RPEM**

In order to prepare for the RPEM Monte Carlo integrations in the M-step, we first evaluate the denominator of $g_{i\ell} (\theta_i, \phi^{(r)})$, which does not depend on $\ell$. We define it as $N_i$ such that:

$$N_i = \sum_{\ell=1}^{K} \omega^{(\ell)} n_{i\ell},$$

(22)

where

$$n_{i\ell} \equiv \int p(Y_i | \sigma^2, \theta_i) p(\theta_i | \mu^{(\hat{\theta})}, \Sigma^{(\hat{\theta})}) d \theta_i.$$

(23)

For $n_{i\ell}$ we sample $\theta_i$ from Gaussian $p(\theta_i | \mu^{(\hat{\theta})}, \Sigma^{(\hat{\theta})})$, then we evaluate Equation 23 by taking the average of the samples of $p(Y_i | \sigma^2, \theta_i)$, that is,

$$\langle n_{i\ell} \rangle = \frac{1}{m_{\text{Gauss}}} \sum_{m=1}^{m_{\text{Gauss}}} p(Y_i | \sigma^2, \theta_i^{(m)}) | \theta_i^{(m)} \in p(\theta | \mu^{(\hat{\theta})}, \Sigma^{(\hat{\theta})}),$$

(24)

where the number of samples $m_{\text{Gauss}}$ is typically set between 200 and 3000 on a single CPU core. Once we obtain $n_{i\ell}$ and $N_i$, the $\tau_i(\ell)$ in Equation 10 can be immediately evaluated by:

$$\tau_i(\ell) = \frac{\omega^{(\ell)} n_{i\ell}}{N_i}.$$

(25)

We also evaluate the log of the likelihood function $L(\phi)$ in Equation 4 as

$$\ln L(\phi) = \sum_{i=1}^{n} \ln (N_i).$$

(26)

The program is iterated until $\ln L(\phi)$ stabilizes.

When dealing with models with ODEs, the E-step can be the most time-consuming part of the algorithm. We typically store the samples of $p(Y_i | \sigma^2, \theta_i)$ calculated from ODEs and obtained for each $i$ and $\ell$ in Equation 24, and reuse them in the M-step.

**M-Step of RPEM**

The M-step is important because it is at the M-step that we truly estimate the parameters. Inaccurate estimation at each iteration of the M-step may cumulate into inaccurate estimations of the parameters and result in unnecessarily long iteration time.

In the M-step, we update all the parameters. The weight in Equation 12 is calculated from the $n_{i\ell}$ obtained in the E-step,

$$\left[ \omega^{(\ell)} \right]^{(r+1)} = \frac{1}{n} \sum_{i=1}^{n} g_{i\ell} (\theta_i, \phi^{(r)}) d \theta_i = \frac{\omega^{(\ell)} n_{i\ell}}{n} \sum_{i=1}^{n} \frac{n_{i\ell}}{N_i}.$$

(27)

The rest of the parameters from Equations 15 to 21 can be cast into one type of integral, which can be generalized as:

$$f = \frac{\sum_{i=1}^{n} \sum_{\ell=1}^{K} \int f_{i\ell} (\theta_i) g_{i\ell} (\theta_i, \phi^{(r)}) d \theta_i}{\sum_{i=1}^{n} \sum_{\ell=1}^{K} \int g_{i\ell} (\theta_i, \phi^{(r)}) d \theta_i}.$$

(28)

For example, from Equations 15 to 21, $f_{i\ell} (\theta_i)$ can be any of the following expressions: $\theta_i$, $\left\{ \theta_i - \left[ \mu^{(\hat{\theta})} \right]^{(r+1)} \right\}$, $\frac{[Y_i - h_i(\theta_i)]^T H_i(\theta_i)^{-1} [Y_i - h_i(\theta_i)]}{\sum_{i=1}^{m_i / n}}$.

$^*$Note that, if $\Sigma^{(\hat{\theta})}$ contains nonzero off diagonal elements, we can still sample $\theta_i$ from the multivariate Gaussian distribution $p(\theta_i | \mu^{(\hat{\theta})}, \Sigma^{(\hat{\theta})})$. This is usually done by using the Cholesky factorization.

$^\dagger$When the integral does not involve looping over the mixture label $\ell$, we simply treat $K$ in Equation 28 as 1.
\( \alpha_i \), \( \{ \alpha_i - [\mu_{\alpha}^{(\text{r}+1)}] \} \{ \alpha_i - [\mu_{\alpha}^{(\text{r}+1)}] \}^T \), \( \beta_i \),

\[
\begin{align*}
\beta_i - (\mu_\beta^{(\text{r}+1)})^T, \\
\beta_i - (\mu_\beta^{(\text{r}+1)})^T.
\end{align*}
\]

Note that,

\[
\sum_{k=1}^{K} \sum_{=1}^{n} d\theta_i \left[ \frac{\sum_{k=1}^{K} \sum_{=1}^{n} g_i(t_i, \phi^{(r)}) d\theta_i}{\sum_{k=1}^{K} \sum_{=1}^{n} g_i(t_i, \phi^{(r)}) d\theta_i} \right] = 1, \quad (29)
\]

so the integrand of Equation 29 can be treated as a target distribution \( \pi(s) \),

\[
\pi(s) = \frac{g_i(t_i, \phi^{(r)})}{\mathcal{N}}
\]

whose normalization factor is \( \mathcal{N} = \sum_{k=1}^{K} \sum_{=1}^{n} g_i(t_i, \phi^{(r)}) d\theta_i \).

RPEM distinguishes itself from other MCPEM algorithms, by treating continuous variables \( \theta_i \) and discrete variables \( i \) and \( \bar{\epsilon} \) the same in the Metropolis–Hastings algorithm. This is because the Monte Carlo method applies not only to continuous variables, but also to discrete variables. The same techniques have already been used in quantum Monte Carlo for several decades, to overcome the difficulty in summing over a formidable number of quantum states without sacrificing accuracy.

RPEM samples \( \pi(s) \) with the Metropolis–Hastings sampling. We denote \( s = \{ i, \bar{\epsilon}, \theta_i \} \) as the current state, \( s' = \{ i', \bar{\epsilon}', \theta_i' \} \) as the proposed state. A common choice of the acceptance probability \( A(s \rightarrow s') \) is

\[
A(s \rightarrow s') = \min \left[ 1, \frac{\pi(s') T(s' \rightarrow s)}{\pi(s) T(s \rightarrow s')} \right], \quad (31)
\]

where \( \pi(s') \) is the new target distribution which equals to \( \frac{g_i(t_i', \phi^{(r)'})}{\mathcal{N}'} \). The proposed transition probability is denoted by \( T(s \rightarrow s') \), which can be further decomposed by \( T(s \rightarrow s') = T(i \rightarrow i') T(\bar{\epsilon} \rightarrow \bar{\epsilon}') T(\theta_i \rightarrow \theta_i') \). The normalization factor \( \mathcal{N}' \) cancels in Equation 31.

The Metropolis–Hastings algorithm proceeds as follows. We randomly sample \( "i" \) from the uniform distribution over the integers from 1 to \( n \) (the number of the subjects), and we randomly sample \( "\bar{\epsilon}" \) from the uniform distribution over the integers from 1 to \( K \) (the number of terms in the mixture). This means both \( T(i \rightarrow i') \) and \( T(\bar{\epsilon} \rightarrow \bar{\epsilon}') \) are constant, so \( T(i \rightarrow i') = T(i' \rightarrow i) \) and \( T(\bar{\epsilon} \rightarrow \bar{\epsilon}') = T(\bar{\epsilon}' \rightarrow \bar{\epsilon}) \). We sample \( \theta_i' \) from \( p(\theta_i' | \mu^{(\bar{\epsilon}')}, \Sigma^{(\bar{\epsilon}')}) \), which means \( T(\theta_i \rightarrow \theta_i') = p(\theta_i | \mu^{(\bar{\epsilon})}, \Sigma^{(\bar{\epsilon})}) \), and \( T(\theta_i' \rightarrow \theta_i) = p(\theta_i' | \mu^{(\bar{\epsilon}')}, \Sigma^{(\bar{\epsilon}')}) \). After we propose the new state \( s' \), the acceptance probability \( A(s \rightarrow s') \) in Equation 31 becomes

\[
A(s \rightarrow s') = \min \left[ 1, \frac{g_i(t_i', \phi^{(r)'}) T(i \rightarrow i') T(\bar{\epsilon} \rightarrow \bar{\epsilon}') T(\theta_i \rightarrow \theta_i')}{g_i(t_i, \phi^{(r)}) T(i \rightarrow i') T(\bar{\epsilon} \rightarrow \bar{\epsilon}') T(\theta_i \rightarrow \theta_i')} \right] \quad (32)
\]

We judge if the new state \( s' \) is accepted by Equation 32. If accepted, we keep the new state \( s' \) and set \( s' \) as the current state \( s \); if not, we keep the current state \( s \) and continue to propose the new state \( s' \). In evaluating Equation 28, to judge if the new state \( s' \) is accepted by Equation 32, we first generate a uniform random variable \( x \) between 0 and 1, then we compare \( x \) with \( \frac{p(Y_i | \sigma^2, \theta_i)}{p(Y_i | \sigma^2, \theta_i)} \times \frac{N_i}{N_i} \times \frac{w^{(s')}}{w^{(s)}} \). If \( x \) is smaller, we accept the new state \( s' \), otherwise we reject \( s' \).

We continue the Metropolis–Hastings process until the target distribution is formed and we get \( m \) sufficient independent samples of \( f_{\theta_i}^{(s')}(\theta_i) \), where \( j \) denotes the label of a sample of \( (\bar{\epsilon}, \bar{\epsilon}', \theta_i) \) combination. The expectation of Equation 28 is evaluated as,

\[
\langle f \rangle \approx \frac{1}{m} \sum_{j=1}^{m} f_{\theta_i}^{(s')} \left( \{ \bar{\epsilon}, \bar{\epsilon}', \theta_i \} \in \pi(s') \right), \quad (33)
\]

and according to the central limit theorem, the error bar of Monte Carlo (also called standard error in statistics) is the standard deviation of the mean of the independent samples. A pseudo Fortran 90 code with descriptions illustrating RPEM in calculating Equation 33 using the Metropolis–Hastings algorithm Equation 32, is provided in the Appendix S1.

RPEM may have advantages over other MCPEM algorithms in terms of low variance and low bias, and these can lead to rapid convergence. About variance, other MCPEM algorithms apply the importance sampling functions designed for Equation 23 in the E-step to Equation 28 in the M-step. This may increase the variance of the estimators when evaluating Equation 28. Because the integrand in Equation 23 and Equation 28 are different, a good importance sampling function which mimics the shape of the integrand in Equation 23, may not mimic the shape of the integrals of Equation 28 hence lead to high variance. In RPEM we do not have this issue, because we evaluate Equation 28 by directly sampling from the target distribution \( \pi(s) \) in Equation 30. About bias, other MCPEM algorithms use biased estimators in evaluating Equation 28, by estimating its numerator and denominator separately, and then calculate their ratio. In RPEM, by directly sampling from the target distribution \( \pi(s) \), we are able to use the unbiased estimator.
that is, **Equation 33**, to estimate **Equation 28**. This can lead to more accurate estimations of **Equation 28** than using biased estimators. The unique Metropolis–Hastings sampling as shown in **Equation 32** makes RPEM not only fast but also accurate, as will be shown in the following section.

## RESULTS AND DISCUSSIONS

### Software and hardware

RPEM is written in modern Fortran and is fully parallelized using MPI. It can be run on both PCs and supercomputers. In this paper, we used Intel Fortran and MPI provided in the free Intel OneAPI 2022.1.3. For the comparison among RPEM, IMP, SAEM, and QRPEM, all the runs are done using six threads and the platform is Windows 10.\(^5\) In RPEM, we use FLINT,\(^34\) which is written in modern Fortran as our non-stiff ODE solver, and when stiff ODEs are detected, we use the latest version of the well-known LSODA solver in ODEPACK.\(^35\) The absolute tolerance (ATOL) and relative tolerance (RTOL) in the ODE solvers are set to \(10^{-6}\) in RPEM. For comparison, NONMEM 7.5 is used, and the parameters TOL and ATOL are set to 6 for the IMP method. SAEM is in Monolix 2019R1 and with default settings. QRPEM is in Certara RsNLME\(^12\) with NLME Engine version 21.11.2, ATOL and RTOL are set to \(10^{-6}\).

For the results in Section “Results with log-normal transformed parameters”, we used a ThinkPad T14 Gen 3 laptop with NONMEM license, which uses Intel 17-1260P CPU (efficiency cores are disabled) and 40GB DDR4-3200 memory. For RPEM, the Monte Carlo sampling size \(m_{\text{Gauss}}\) in **Equation 24** is set to 600. For IMP and QRPEM, 300 samples per subject are used. SAEM uses default settings.

For all the other results, we used a ThinkPad P72 laptop with Intel Xeon-2186M CPU and 64GB DDR4-2666 ECC memory. The Monte Carlo sampling size \(m_{\text{Gauss}}\) is set to 1000 for RPEM. SAEM uses default settings. QRPEM’s sample size is set to 500, and the maximum ODE step is set to 500.

### Model with analytic solution

We first report the results of RPEM by using the same two-stage model as in ref. [4] with the useful case as indicated by **Equations 18–21**. It is a one-compartment PK model with \(K=2\) mixing whose plasma concentration is given by

\[
y_{ij} = \frac{D}{V_i} e^{-k_{ij} t} (1 + \epsilon_{ij}),
\]  

where \(j\) ranges from 1 to the number of samples per subject, \(m_i\), and \(i\) ranges from 1 to the number of subjects, \(n\). The residual errors are denoted by \(\epsilon_{ij}\), which is a normally distributed random variable with mean of zero and standard deviation \(\sigma\). \(D\) is a bolus drug administration with the unit of dose, \(V_i\) is the volume, and \(k_i\) is the elimination rate constant.

At the first stage, \(Y_i \sim N(h_i(\theta_i), \sigma^2 H_i(\theta_i))\), we have,

\[
h_i(\theta_i) = \frac{D}{V_i} \left( e^{-k_{1i} t_1} + \ldots + e^{-k_{ni} V_i} \right),
\]

and therefore,

\[
p(Y_i | \sigma^2, \theta_i) = \prod_{j=1}^{m_i} \frac{1}{\sigma V_i \sqrt{2\pi}} \exp \left\{ -\frac{1}{2} \frac{\left( \frac{D}{V_i} \exp(-k_{ij} t_j) \right)^2}{\sigma^2 V_i \exp(-k_{ij} V_i)} \right\}.
\]

At the second stage, we have

\[
\theta_i = (k_i, V_i)^T \sim \mathcal{N}_R \left( \mu^{(\delta)}, \Sigma^{(\delta)} \right),
\]

where \(\mu^{(\delta)} = \left( \mu_k, \mu_V \right)^T\) and we assume \(\Sigma^{(\delta)} = \text{diag} \left( \sigma_k^2, \sigma_V^2 \right)^2\). Therefore,

\[
p(\theta_i | \mu^{(\delta)}, \Sigma^{(\delta)}) = \frac{1}{\sigma_k \sqrt{2\pi}} \exp \left( -\frac{1}{2} \frac{\left( \frac{\mu_k - \mu^{(\delta)}_k}{\sigma_k} \right)^2}{\frac{1}{\sigma^2}} \right) \times \frac{1}{\sigma_V \sqrt{2\pi}} \exp \left( -\frac{1}{2} \frac{\left( \frac{\mu_V - \mu^{(\delta)}_V}{\sigma_V} \right)^2}{\frac{1}{\sigma^2}} \right).
\]

For the \(K=2\) case used in this paper, \(k_i\) is sampled from two mixtures of Gaussians whereas \(V_i\) is sampled from just one Gaussian, so \(\mu^{(\delta)} \) and \(\Sigma^{(\delta)} \) can be written as \(\mu^{(1)} = \left( \mu_k^{(1)}, \mu_V \right)^T, \mu^{(2)} = \left( \mu_k^{(2)}, \mu_V \right)^T\), \(\Sigma^{(1)} = \text{diag} \left( \sigma_k^{(1)^2}, \sigma_V^2 \right)^2\) and \(\Sigma^{(2)} = \text{diag} \left( \sigma_k^{(2)^2}, \sigma_V^2 \right)^2\). We fix \(D=100\), and we set \(m_i=5\) such that \(t_1\) to \(t_5\) are 1.5, 2, 3, 4, and 5.5, correspondingly. For **Equation 34**, all the parameters used to generate the data \(Y_i\) are listed in the “True Values” column in **Table 1**.

The initial condition of the parameters is listed in the “Initial Values” column. Starting from some initial condition, our

\(^5\) We also tested the speed of RPEM using GFortran and MPICH on Linux and Mac with the M1 chip. The speeds are consistent and similar with using Intel OneAPI.
The purpose is to use RPEM to find the true values of the parameters.

In Table 1, we report the estimated parameters using RPEM for 50 iterations on one CPU core, and for different number of subjects \( n = 100, n = 1000 \) and \( n = 10,000 \). In all cases, \( m_{\text{Gauss}} \) for each subject at the E-step is set to 1000, and the number of Metropolis–Hastings trials for each mixture is set to 1000 at the M-step are set to 100\( n \) (the autocorrelation time is about 100 steps). For the \( n = 100 \) case with 50 iterations, RPEM took 1.3 seconds with results comparable to those obtained by the tradition MCPEM. The speed of RPEM enables us to deal with large datasets efficiently. For the \( n = 1000 \) case, it took 13.4 s, and for the \( n = 10,000 \) case it took 137 s. Besides speed and accuracy, we also tested RPEM’s robustness. We extensively assessed it with varying initial conditions, and RPEM consistently converged within the first 50 iterations.

The standard errors are calculated by collecting all the converged Gaussian samples from iteration 45 to 50 (1000 samples per subject per iteration, so 6000 samples for each subject) and by using the Fisher score method described in section 5 in ref. [4]. As pointed out in ref. [37], the standard error is proportional to \( 1/\sqrt{n} \). Indeed, our standard error calculation confirmed this. For example, we can see that the standard error of each parameter for the \( n = 10,000 \) case is about 10 times smaller than those for the \( n = 100 \) case.

### Table 1: Parameter values estimated by RPEM for varying subject numbers, \( n \), each using 50 iterations.

| Parameter values estimated by RPEM for varying subject numbers, \( n \), each using 50 iterations. | \( n = 100 \) | \( n = 10^3 \) | \( n = 10^4 \) |
|---|---|---|---|
| \( \mu_V \) | 20 | 19.95 | 20.00 |
| | \( \pm 0.3 \) | \( \pm 0.09 \) | \( \pm 0.03 \) |
| \( \mu_k^{(1)} \) | 0.3 | 0.304 | 0.3013 |
| | \( \pm 0.008 \) | \( \pm 0.003 \) | \( \pm 0.0008 \) |
| \( \mu_k^{(2)} \) | 0.6 | 0.592 | 0.602 |
| | \( \pm 0.01 \) | \( \pm 0.007 \) | \( \pm 0.002 \) |
| \( \omega^{(1)} \) | 0.8 | 0.80 | 0.802 |
| | \( \pm 0.04 \) | \( \pm 0.02 \) | \( \pm 0.004 \) |
| \( \omega^{(2)} \) | 0.2 | 0.20 | 0.198 |
| | \( \pm 0.04 \) | \( \pm 0.02 \) | \( \pm 0.004 \) |
| \( \sigma_V \) | 2 | 1.96 | 1.97 |
| | \( \pm 0.4 \) | \( \pm 0.09 \) | \( \pm 0.03 \) |
| \( \sigma_k^{(1)} \) | 0.06 | 0.058 | 0.0595 |
| | \( \pm 0.008 \) | \( \pm 0.002 \) | \( \pm 0.0008 \) |
| \( \sigma_k^{(2)} \) | 0.06 | 0.061 | 0.058 |
| | \( \pm 0.01 \) | \( \pm 0.005 \) | \( \pm 0.002 \) |
| \( \omega \) | 0.1 | 0.1000 | 0.09985 |
| | \( \pm 0.0003 \) | \( \pm 0.00003 \) | \( \pm 0.00002 \) |

Note: True values are simulated, and initial conditions for each experiment are indicated. Time to completion is included in the column headers. Data are presented as the mean. The estimated values are on the first row of each parameter. The estimated standard errors are on the second row of each parameter and begin with \( \pm \) (e.g., for the \( n = 100 \) case, the estimated standard error for \( \mu_V \) is 0.3), they are calculated by collecting all the converged Gaussian samples from iteration 45 to 50 (so 6000 samples for each subject) and by using the method described in section 5 in ref. [4].

Abbreviation: RPEM, Randomized Parametric Expectation Maximization.

In this section, our task is to use RPEM, SAEM, and QRPEM to reconstruct the \( \mu^{(\theta)} \) and \( \Sigma^{(\theta)} \) from the simulated data based on a realistic voriconazole model with ODEs, and compare RPEM with SAEM and QRPEM in terms of speed and accuracy.

### Voriconazole model

In this voriconazole model, we follow the data and model format for Pmetrics. The seven primary (structural) parameter vector for subject \( i \) is given by,

\[
\theta_i = (K_a, V_{max}, K_m, V_c, F_A1, K_{cp}, K_{pc})^T,
\]

and the population mean is
\[
\mathbf{\mu}(\vec{k}) = \left( \mu_{K_a}, \mu_{V_{max0}}, \mu_{K_m^i}, \mu_{V_C0}, \mu_{F_A1}, \mu_{K_{cp}^i}, \mu_{K_{pc}^i} \right)^T, \tag{41}
\]

and again, we assume \( \mathbf{\Sigma}(\vec{k}) = \text{diag}(\sigma^2_{K_a^i}, \sigma^2_{V_{max0}^i}, \sigma^2_{K_m^i}, \sigma^2_{V_C0^i}, \sigma^2_{F_A1}, \sigma^2_{K_{cp}^i}, \sigma^2_{K_{pc}^i}) \). The covariate is weight (wt). The secondary parameters which are obtained from primary parameters and covariate are \( V_m \) and \( V \),

\[
V_m = V_{max0} \times \text{wt}^{0.75},
\]

\[
V = V_{c0} \times \text{wt}.
\]

For any subject \( i \), the ODEs are listed as below:

\[
\frac{d x_1}{dt} = -K_a \times x_1, \tag{44}
\]

\[
\frac{d x_2}{dt} = -K_a \times x_1 + r_V(i(t)) - \frac{V_m(i)}{K_m(i)} \times x_2 - K_{cp} \times x_2 + K_{pc} \times x_3, \tag{45}
\]

\[
\frac{d x_3}{dt} = K_{cp} \times x_2 - K_{pc} \times x_3, \tag{46}
\]

where for subject \( i \), \( r_V(i(t)) \) is the ratio between dose and duration at time \( t \). If at time \( t \) the dose is non-zero and duration is zero, it means a bolus and \( x_1(t) \) needs to be added by an additional dose \( \times F_A1 \). \( V_m(i) \) and \( V(i) \) are the secondary parameters \( V_m \) and \( V \) for subject \( i \). Because the ODEs represent the same structural model for all individuals, for simplicity, we omit the superscript \( (i) \) on \( x_1 \) to \( x_3 \), and on the seven primary parameters in Equation 66. However, we put superscript \( (i) \) on \( r_V(i(t)), V_m(i) \) and \( V \), to emphasize that they depend on the covariates for subject \( i \).

The concentration for subject \( i \) at time \( t_j \) is given by,

\[
y_{ij} = \frac{x_2(i(t_j))}{V(i)} + \epsilon_{ij}, \tag{47}
\]

and we assume the noise \( \epsilon_{ij} \) is a Gaussian random variable whose standard deviation \( \sigma_{y_{ij}} \) has the following form,

\[
\sigma_{y_{ij}} = c_0 + c_1 \times \frac{x_2(i(t_j))}{V}. \tag{48}
\]

where \( c_0 \) to \( c_1 \) are known non-negative constants. This is the combined additive and multiplicative error model\(^{41} \) we used in LAPK\(^{3} \) for many years and we find it reasonable in most cases.

\(^{3}\) Laboratory of Applied Pharmacokinetics and Bioinformatics (LAPK) is part of the Children’s Hospital Los Angeles (CHLA) affiliated with University of Southern California (USC).

Similar with what has been described in Section “Model with analytic solution”, at the first stage, \( Y_i \sim N(h_i(\theta_i), H_i(\theta_i)) \), and for this model we have

\[
h_i(\theta_i) = \frac{1}{V(i)} \left[ x_2(i(t_1)), …, x_2(i(t_{m_i})) \right]^T, \tag{49}
\]

\[
H_i(\theta_i) = \text{diag}(\sigma^2_{1i}, …, \sigma^2_{m_{i}i}), \tag{50}
\]

and, therefore,

\[
p(Y_i | \boldsymbol{\sigma}_{y_{ij}}, \theta_i) = \prod_{j=1}^{m_i} \frac{1}{\sqrt{2\pi \sigma_{y_{ij}}}} \exp \left\{-\frac{1}{2} \frac{Y_{ij} - x_2(i(t_j))}{\sigma_{y_{ij}}}^2 \right\}. \tag{51}
\]

At the second stage, again we have

\[
\theta_i \sim_{i.i.d} \sum_{k=1}^{K} w(\vec{k}) N(\mathbf{\mu}(\vec{k}), \mathbf{\Sigma}(\vec{k})).
\]

Simulated data

For the simulated data, we set the number of subjects \( n = 50 \). We assume \( K = 1 \), so there is no Gaussian mixture, the superscript \( (\vec{k}) \) can be removed or simply set as \( 1 \), the covariate is weight. For each subject, we set observation time \( t \) (unit is hour) as 2, 4, 6, 8, …, 48 so the number of observations is \( m_i = 24 \). At \( t = 0 \), we set dose (unit is mg) as 180 and duration time as 2. At \( t = 24 \) we set dose as 180 and duration time as 0. Similarly, with what was done in ref. [42], the covariate weights are all set to 16.5 (unit is kg), to minimize the effects/noise from covariates when reconstructing the population parameters \( \mathbf{\mu}(\vec{k}) \) and \( \mathbf{\Sigma}(\vec{k}) \) (in real data situations each subject will have its own weight). For the noise in Equation 48, we set \( c_0 = 0.02, c_1 = 0.1 \).

The simulated data file for the voriconazole ODE model can be found in ref. [43] (provided in the Appendix S1). The observation concentration data \( y_{ij} \) is provided as the “OUT” column. They can be generated by using the 50 sets of primary parameters in ref. [44] (provided in the Appendix S1), which are directly sampled from the Gaussian whose true values of \( \mathbf{\mu}(\vec{k}) \) and \( \mathbf{\Sigma}(\vec{k}) \) are listed as the “True” rows in Table 2. No log transformations of the primary parameters are needed for our data. The Monolix project files for the model and dataset are also provided in the Appendix S1.

Stopping criterion

RPEM’s stopping criterion is described as follows. As the iteration is going on, we take the lastest consecutive
and obtain our final estimations of stop the iteration, and we take the samples of the latest sample from these stabilized samples using Equation 32 consecutive 30 runs as stabilized samples.

None of the stabilized samples are wasted. We resample from these stabilized samples using Equation 32 and obtain our final estimations of \( \mu^{(E)} \) and \( \Sigma^{(E)} \). Besides, because the stabilized samples of \( \theta_i \) in RPEM are approximately distributed from \( \sum_{k=1}^{K} w_k^{(E)} N\left( \mu^{(E)}, \Sigma^{(E)} \right) \), we also included a fast Gaussian mixture model (GMM) clustering algorithm which directly finds the optimum \( \mu^{(E)} \) and \( \Sigma^{(E)} \) from the stabilized samples of \( \theta_i \) as well. Such results are marked as RPEM-GMM. So, when RPEM finishes, it will have RPEM-GMM results also. When report the speed of RPEM, the cost of RPEM-GMM is already included.

30 iterations’ log likelihood, calculate their slope based on least square method. Before convergence, this slope must be positive. After convergence, the log likelihood is flattened, and this slope cannot be always positive anymore. Because the Monte Carlo integrals have uncertainties, the slope will fluctuate around zero, that is, it can be slightly positive or slightly negative. Once we detect at which iteration a negative slope first occurs, we stop the iteration, and we take the samples of the latest consecutive 30 runs as stabilized samples.

None of the stabilized samples are wasted. We resample from these stabilized samples using Equation 32 and obtain our final estimations of \( \mu^{(E)} \) and \( \Sigma^{(E)} \). Besides, because the stabilized samples of \( \theta_i \) in RPEM are approximately distributed from \( \sum_{k=1}^{K} w_k^{(E)} N\left( \mu^{(E)}, \Sigma^{(E)} \right) \), we also included a fast Gaussian mixture model (GMM) clustering algorithm which directly finds the optimum \( \mu^{(E)} \) and \( \Sigma^{(E)} \) from the stabilized samples of \( \theta_i \) as well. Such results are marked as RPEM-GMM. So, when RPEM finishes, it will have RPEM-GMM results also. When report the speed of RPEM, the cost of RPEM-GMM is already included.

### Table 2: The parameter reconstruction comparisons among RPEM, SAEM, and QRPEM.

| Method (% error/parameter) | \( \mu_{K_e} \) | \( \mu_{V_{max}} \) | \( \mu_{K_{in}} \) | \( \mu_{V_{in}} \) | \( \mu_{K_{cp}} \) | \( \mu_{V_{cp}} \) |
|---------------------------|----------------|----------------|----------------|----------------|----------------|----------------|
| True                      | 2.26 ± 0.48    | 9.23 ± 1.64   | 10.32 ± 2.59   | 1.16 ± 0.14    | 0.73 ± 0.01    | 1.75 ± 0.13    |
| RPEM (14.9%)              | 2.46 ± 0.55    | 9.54 ± 1.66   | 11.32 ± 2.34   | 1.21 ± 0.17    | 0.74 ± 0.01    | 1.59 ± 0.04    |
| RPEM-GMM (14.9%)          | 2.52 ± 0.78    | 9.47 ± 2.55   | 10.69 ± 3.66   | 1.34 ± 0.16    | 0.74 ± 0.01    | 1.21 ± 0.17    |
| SAEM (26.9%)              | 3.65 ± 1.65    | 9.47 ± 2.55   | 10.69 ± 3.66   | 1.34 ± 0.16    | 0.74 ± 0.01    | 1.21 ± 0.17    |
| QRPEM (35.9%)             | 3.20 ± 0.78    | 13.94 ± 1.69  | 17.22 ± 2.47   | 1.45 ± 0.17    | 0.73 ± 0.01    | 0.99 ± 0.10    |

| Method (% error/sigma)    | \( \sigma_{K_e} \) | \( \sigma_{V_{max}} \) | \( \sigma_{K_{in}} \) | \( \sigma_{V_{in}} \) | \( \sigma_{K_{cp}} \) | \( \sigma_{V_{cp}} \) |
|---------------------------|----------------|----------------|----------------|----------------|----------------|----------------|
| True                      | 0.76 ± 0.19    | 3.96 ± 1.13   | 4.45 ± 1.49    | 0.17 ± 0.09    | 0.07 ± 0.01    | 0.77 ± 0.18    |
| RPEM (37.4%)              | 0.46 ± 0.21    | 3.83 ± 0.60   | 6.09 ± 1.07    | 0.22 ± 0.08    | 0.04 ± 0.01    | 0.48 ± 0.08    |
| RPEM-GMM (31.5%)          | 0.55 ± 0.83    | 3.53 ± 0.95   | 5.33 ± 1.94    | 0.27 ± 0.06    | 0.07 ± 0.01    | 0.51 ± 0.12    |
| SAEM (40.6%)              | 1.40 ± 0.55    | 3.95 ± 0.95   | 5.06 ± 1.94    | 0.27 ± 0.64    | 0.07 ± 0.01    | 0.51 ± 0.12    |
| QRPEM (43.0%)             | 1.24 ± 0.41    | 5.55 ± 1.09   | 8.04 ± 1.13    | 0.25 ± 0.13    | 0.06 ± 0.01    | 0.58 ± 0.11    |

Note: The results are obtained by averaging the results from the 21 randomly picked initial conditions. The averaged values are listed at the first row of each method, and the standard deviation (which may be served as the estimation of the standard error for the parameter) are listed at the second row of each method and begin with ± (e.g., for RPEM, the averaged \( \mu_{K_e} \) is 2.46, the standard deviation is 0.48). For each method, the overall averaged percentage errors for all the \( \mu^{(E)} \) and \( \Sigma^{(E)} \) are listed in the parentheses correspondingly.

Abbreviations: QRPEM, Quasi-Random Parametric Expectation Maximization; RPEM, Randomized Parametric Expectation Maximization; SAEM, Stochastic Approximation Expectation Maximization.

Comparison among RPEM, SAEM, and QRPEM

We randomly picked 21 initial conditions from ref. [44], which is used to generate the simulated data. The 21 initial conditions are picked from ID numbers 1, 10, 13, 15, 17, 20, 23, 25, 27, 3, 30, 33, 35, 37, 40, 43, 45, 47, 5, 50, and 7, by setting the corresponding primary parameters as the initial \( \mu^{(E)} \) and \( \Sigma^{(E)} \). We let RPEM, SAEM, and QRPEM start from these 21 initial conditions and compare their speed and accuracy in reconstructing true population parameters \( \mu^{(E)} \) and \( \Sigma^{(E)} \). We use \( \sigma_{K_e}, \sigma_{V_{max}}, \sigma_{K_{in}}, \sigma_{V_{in}}, \sigma_{F_{A1}}, \sigma_{K_{cp}}, \sigma_{V_{cp}} \) to represent \( \Sigma^{(E)} \).

We list the speed comparison among RPEM, SAEM, and QRPEM in Figure 1. We see that most RPEM runs finished at around 20 to 25s (about 42 iterations, each RPEM iteration takes about 0.6s). The rest of RPEM runs finished between

\(^{1}\)In this example, we do not use NONMEM’s importance sampling Monte Carlo EM method (IMP) here, because IMP cannot finish the runs, due to the difficulties in handling the Gaussian distributed parameters.
30 to 60 s. For SAEM, which are represented by the red symbols, most of them finished between 90 and 120 s, and few of them finished between 130 and 144 s. The results of QRPEM are more scattered, the fast runs finish around 64 seconds, whereas the slow ones take around 192 s. On average, we find that for the voriconazole model, the speed of SAEM and QRPEM are comparable (QRPEM is slightly faster), and RPEM is about three or four times faster than both.

In Figure 2, we show RPEM, SAEM, and RPEM’s abilities in reconstructing the true population parameters $\mu^{(6)}$ and $\Sigma^{(6)}$. We find that almost all the true $\mu^{(6)}$ and $\Sigma^{(6)}$ are covered by RPEM within the 90% confidence interval (CI). Out of the total 14 population parameters, RPEM only missed two of them, namely $\sigma_{V_{ct}}$ and $\sigma_{K_{pc}}$, outside the 90% CI. SAEM on the other hand, missed seven out of 14, $\mu_{F_{A1}}, \mu_{K_{tp}}, \sigma_{V_{ct}}, \mu_{K_{pc}}, \sigma_{V_{ct}}, \sigma_{K_{pc}}$, and $\sigma_{V_{ct}}$. Within the 50% CI, RPEM only missed six out of 14, $\sigma_{V_{ct}}, \sigma_{F_{A1}}, \sigma_{K_{pc}}, \mu_{K_{pc}}, \sigma_{V_{ct}}, \sigma_{K_{pc}}$. Whereas SAEM missed 10 out of 14, namely $\mu_{K_{pc}}, \sigma_{V_{ct}}, \sigma_{F_{A1}}, \sigma_{K_{pc}}, \mu_{K_{pc}}, \sigma_{K_{pc}}, \mu_{K_{pc}}, \sigma_{V_{ct}}, \sigma_{K_{pc}}$, and $\sigma_{F_{A1}}$. For QRPEM, we find it nearly missed all the true $\mu^{(6)}$ and $\Sigma^{(6)}$. However, we find some ODE difficulties in QRPEM due to the additive parameterizations for the voriconazole model, therefore, we suspect its accuracy is compromised because of that.

To quantitatively compare the accuracy among RPEM, QRPEM, and SAEM, in Figure 3, for all of them, we plot the averaged percentage error for all the $\Sigma^{(6)}$ versus averaged percentage error for all the $\mu^{(6)}$, for each of the 21 runs.**

The symbols to the lower and the left corner are better. We found that both RPEM and RPEM-GMM are more concentrated toward the lower and the left corner than SAEM and QRPEM. Both SAEM and QRPEM’s averaged percentage error for all the $\mu^{(6)}$ are bigger than RPEM’s. The overall averaged percentage error for all the $\mu^{(6)}$†† for both RPEM and RPEM-GMM are about 14.9%, for SAEM is 26.9%, and for QRPEM is 35.9%. The overall averaged percentage error for all the $\Sigma^{(6)}$ for RPEM and RPEM-GMM are about 37.4% and 31.5%, for SAEM is 40.6%, and for QRPEM is 43.0%. We also notice that SAEM and QRPEM’s outliers can be about two times of RPEM’s. In Table 2, we list the averaged $\mu^{(6)}$ and $\Sigma^{(6)}$ reconstructed from the 21 initial conditions of RPEM, RPEM-GMM, SAEM, and QRPEM.

Based on the analysis of Figure 1, Figure 2, Figure 3, as well as Table 2, we can qualitatively conclude that, for the voriconazole model, RPEM is faster and more accurate than SAEM and QRPEM.

We need to point out that, for RPEM, by using the Metropolis–Hastings algorithm Equation 32, we did not neglect any of the 50 subjects in any iterations in any of the RPEM runs. All 50 subjects are always almost equally sampled, that is, in the M-step in each iteration in each RPEM run, each of the 50 subjects shares about 2% of all the samples in Equation 33.

**In each run, each of the $\sigma_{K_{pc}}, \sigma_{V_{ct}}$, $\sigma_{F_{A1}}$, $\sigma_{V_{ct}}$, $\sigma_{K_{pc}}$, and $\sigma_{K_{pc}}$ has a corresponding percentage error compared with their true values. We sum up these seven percentage errors and divide it by seven, and that is how we obtain the averaged percentage error for all the $\Sigma^{(6)}$. Averaged percentage error for all the $\mu^{(6)}$ is similarly obtained.

††The overall averaged percentage error for all the $\mu^{(6)}$†† means that we averaged the 21 runs’ averaged percentage error for all the $\mu^{(6)}$. Same thing applies to the overall averaged percentage error for all the $\Sigma^{(6)}$. 

**In each run, each of the $\sigma_{K_{pc}}, \sigma_{V_{ct}}$, $\sigma_{F_{A1}}$, $\sigma_{V_{ct}}$, $\sigma_{K_{pc}}$, and $\sigma_{K_{pc}}$ has a corresponding percentage error compared with their true values. We sum up these seven percentage errors and divide it by seven, and that is how we obtain the averaged percentage error for all the $\Sigma^{(6)}$. Averaged percentage error for all the $\mu^{(6)}$ is similarly obtained.

††The overall averaged percentage error for all the $\mu^{(6)}$†† means that we averaged the 21 runs’ averaged percentage error for all the $\mu^{(6)}$. Same thing applies to the overall averaged percentage error for all the $\Sigma^{(6)}$. 

**In each run, each of the $\sigma_{K_{pc}}, \sigma_{V_{ct}}$, $\sigma_{F_{A1}}$, $\sigma_{V_{ct}}$, $\sigma_{K_{pc}}$, and $\sigma_{K_{pc}}$ has a corresponding percentage error compared with their true values. We sum up these seven percentage errors and divide it by seven, and that is how we obtain the averaged percentage error for all the $\Sigma^{(6)}$. Averaged percentage error for all the $\mu^{(6)}$ is similarly obtained.

††The overall averaged percentage error for all the $\mu^{(6)}$†† means that we averaged the 21 runs’ averaged percentage error for all the $\mu^{(6)}$. Same thing applies to the overall averaged percentage error for all the $\Sigma^{(6)}$. 

**In each run, each of the $\sigma_{K_{pc}}, \sigma_{V_{ct}}$, $\sigma_{F_{A1}}$, $\sigma_{V_{ct}}$, $\sigma_{K_{pc}}$, and $\sigma_{K_{pc}}$ has a corresponding percentage error compared with their true values. We sum up these seven percentage errors and divide it by seven, and that is how we obtain the averaged percentage error for all the $\Sigma^{(6)}$. Averaged percentage error for all the $\mu^{(6)}$ is similarly obtained.
FIGURE 2 The comparison among QRPEM (black), SAEM (red), RPEM-GMM (blue), and RPEM (green) in terms of the ability of reconstructing $\mu(\theta)$ and $\Sigma(\theta)$. The leftmost symbol is the true value. The definition of the box with data overlap is the same as in Figure 1. QRPEM, Quasi-Random Parametric Expectation Maximization; RPEM, Randomized Parametric Expectation Maximization; SAEM, Stochastic Approximation Expectation Maximization.
condition, \( \mu = (2.03, 10.18, 11.11, 12.0, 78, 1.26, 0.78)^T \), and 
\( \Sigma = (\mu/2.5)^2 \). We show their parameter and standard error estimates. They are calculated from collecting all the Gaussian samples from the latest 30 consecutive converged iterations (1000 samples per subject per iteration, so totally 30,000 samples for each subject). The estimated standard error for each parameter begins with the symbol \( \pm \) (e.g., for RPEM, the estimated \( \mu_{K_p} \) is 2.40, the standard error is 0.34). We find that, in general, the true parameters are within the range of the estimated parameters plus and minus their standard errors (68% CI). Even for parameters such as \( \mu_{K_p}, \mu_{K_{p'}}, \sigma_{K_p}, \sigma_{F_{L}1}, \sigma_{K_{p'}}, \) and \( \sigma_{F_{L}1} \), their true values are outside the range of the estimated parameters plus and minus their standard errors, the true values are not off by too much. If we consider the range of the estimated parameters plus and minus twice their standard errors (95% CI), then all the true parameters are within the range.

### Predictions

Once RPEM run is finished, based on the Gaussian samples from the latest 30 consecutive converged iterations (so 30,000 samples for each subject are used). For all the subjects, population prediction and individual predictions have been calculated too.

Population prediction for subject \( i \) at time slot \( j \), that is, \( \langle y_{ij}^{\text{pred}} \rangle_{\text{pop}} \), is made by calculating the expectation of \( y_{ij}^{\text{pred}}(\theta_i) \) using the population probability density \( \sum_{\xi=1}^{K} w^{(\xi)} p(\theta_i | \mu^{(\xi)}, \Sigma^{(\xi)}) \) as follows,

\[
\langle y_{ij}^{\text{pred}} \rangle_{\text{pop}} = \sum_{\xi=1}^{K} y_{ij}^{\text{pred}}(\theta_i) w^{(\xi)} p(\theta_i | \mu^{(\xi)}, \Sigma^{(\xi)}) \, d\theta_i, \tag{52}
\]

where \( y_{ij}^{\text{pred}}(\theta_i) \) is the \( j \)th element in the \( h_i(\theta_i) \) in Equation 49. This is essentially the same as the equation 10 in ref. [47].

For the individual prediction \( \langle y_{ij}^{\text{pred}} \rangle_{\text{ind}} \), a typical way\(^{47} \) is to find the optimal parameters for each subject first, by using the normalized posterior distribution \( \sum_{\xi=1}^{K} b_{i\xi}(\theta_i, \phi) \) as follows:

\[
\langle \theta_i \rangle_{\text{opt}} = \frac{\sum_{\xi=1}^{K} \theta_i b_{i\xi}(\theta_i, \phi) \, d\theta_i}{\sum_{\xi=1}^{K} \int p(Y_i | \sigma^2, \theta_i) p(\theta_i | \mu^{(\xi)}, \Sigma^{(\xi)}) \, d\theta_i},
\]

\[
= \frac{\sum_{\xi=1}^{K} \int p(Y_i | \sigma^2, \theta_i) w^{(\xi)} p(\theta_i | \mu^{(\xi)}, \Sigma^{(\xi)}) \, d\theta_i}{\sum_{\xi=1}^{K} \int p(Y_i | \sigma^2, \theta_i) w^{(\xi)} \, d\theta_i}.
\tag{53}
\]
then plug the obtained optimal parameters \( \langle \theta_i \rangle_{\text{opt}} \) into the model \( y_{ij}^{\text{pred}}(\theta_i) \) to make individual predictions, that is,

\[
\langle y_{ij}^{\text{pred}} \rangle_{\text{ind}} = y_{ij}^{\text{pred}}(\langle \theta_i \rangle_{\text{opt}}).
\] (54)

We would like to mention that, from mathematical point of view, there may be other ways to calculate the individual prediction \( \langle y_{ij}^{\text{pred}} \rangle_{\text{ind}} \). For example, it can be the expectation of \( y_{ij}^{\text{pred}}(\theta_i) \) using the posterior distribution \( g_{\Phi}(\theta_i, \phi) \) in Equation 8. We denote this individual prediction using \( \langle y_{ij}^{\text{pred}} \rangle_{\text{ind}}^* \), to differ it from the symbol of \( \langle y_{ij}^{\text{pred}} \rangle_{\text{ind}} \) in Equation 54. The equation is:

\[
\langle y_{ij}^{\text{pred}} \rangle_{\text{ind}}^* = \sum_{k=1}^{K} \int y_{ij}^{\text{pred}}(\theta_i) g_{\Phi}(\theta_i, \phi) d\theta_i = \sum_{k=1}^{K} \int y_{ij}^{\text{pred}}(\theta_i) p(\theta_i | \mu^{(k)}, \Sigma^{(k)}) d\theta_i \]

\[
= \frac{1}{\sum_{k=1}^{K} \int p(\theta_i | \mu^{(k)}, \Sigma^{(k)}) d\theta_i} \sum_{k=1}^{K} \int y_{ij}^{\text{pred}}(\theta_i) p(\theta_i | \mu^{(k)}, \Sigma^{(k)}) d\theta_i.
\]

Note that the likelihood \( p(Y_i | \sigma^2, \theta_i) \) can be written in the form of a product, that is,

\[
p(Y_i | \sigma^2, \theta_i) = \prod_{j=1}^{m_i} p_j(Y_i | \sigma^2, \theta_i),
\] (56)

where \( p_j(Y_i | \sigma^2, \theta_i) \) is the likelihood at time \( t_j \) which is the \( j \)th time slot. For the voriconazole model, from Equation 51 we can find that,
Dealing with the truncated Gaussian

For the voriconazole PK model, all the components of the parameter $\theta_i$ must be positive. From the true values listed in Table 2, it can be estimated and determined later that, if we require all the seven parameters in $\theta_i$ to be positive, the seven-dimensional population distribution, that is, the Gaussian $N(\mu, \Sigma)$, will be truncated slightly by about 4%.

When we simulate the dataset, we generate a sample of the seven primary parameters $\theta_i$ from the seven-dimensional Gaussian $N(\mu, \Sigma)$, and we use this sample to generate the concentration versus time for a subject. Indeed, to make each of the subject associate with seven non-negative primary parameters, we need to drop about 4% of $\theta_i$ samples which contain at least one negative parameter. So, compared with the true Gaussian, the effective Gaussian used to generate the dataset, can be taken as is slightly truncated.

However, our job is to reconstruct the true Gaussian $N(\mu, \Sigma)$ which is not truncated. To achieve this, all we need to do in the integrals in the RPEM algorithm is, we sample $\theta_i$ freely without setting any lower or upper bound, and whenever the sample of $\theta_i$ contains any negative parameters, the predicted concentration in the corresponding likelihood $p(Y_i | \sigma^2, \theta_i)$ is unrealistic, so the likelihood $p(Y_i | \sigma^2, \theta_i)$ is set to zero, and there is no need to call the ODE solver. With this reason, we can think of those dropped subjects which contain negative parameters are still there in the simulated dataset, it is just that some of their simulated concentration values $Y_{ij}$ in $Y_i$ are so unrealistic, such that their likelihood $p(Y_i | \sigma^2, \theta_i)$ for any $\theta_i$ as shown in Equation 51 is zero or approximately zero. In this way, the unrealistic subjects in the dataset have no influence in the EM algorithm, because their contributions are zero in calculating the total likelihood in the E-step and updating the population parameters in the M-step. Therefore, we can imagine those unrealistic subjects are still in the dataset, so we are actually dealing with the full data set instead of the truncated dataset. The Gaussian distribution we reconstruct should be approximately the true Gaussian distribution which is used to simulate the dataset. In fact, the results in Table 2 and Table 3 confirm this.

Note that, when calculating the integrals $\int \gamma^{\text{pred}}_{ij}(\theta_i) p(\theta_i | \mu(\theta_i, \Sigma(\theta_i)) \ d\theta_i$, in the population prediction Equation 52, we will need to set the lower bound of $\theta_i$ as zero, because $\gamma^{\text{pred}}_{ij}(\theta_i)$ is unrealistic for $\theta_i$ containing negative elements, and unlike the individual predictions, in population predictions there is no likelihood $p(Y_i | \sigma^2, \theta_i)$ to be set to zero to handle such cases. So, instead of evaluating $\int \gamma^{\text{pred}}_{ij}(\theta_i) p(\theta_i | \mu(\theta_i, \Sigma(\theta_i)) \ d\theta_i$, we will be evaluating the truncated integral $\frac{1}{N(\theta_i)} \int_0^{+\infty} \gamma^{\text{pred}}_{ij}(\theta_i) p(\theta_i | \mu(\theta_i, \Sigma(\theta_i)) \ d\theta_i,$ which can be further written as,

$$\frac{1}{N(\theta_i)} \int_0^{+\infty} \gamma^{\text{pred}}_{ij}(\theta_i) p(\theta_i | \mu(\theta_i, \Sigma(\theta_i)) \ d\theta_i = \frac{1}{N(\theta_i)} \int_0^{+\infty} \gamma^{\text{pred}}_{ij}(\theta_i) p(\theta_i | \mu(\theta_i, \Sigma(\theta_i)) \ d\theta_i,$$  \hspace{1cm} (59)

where $N(\theta_i)$ is the normalization factor such that $\frac{1}{N(\theta_i)} \int_0^{+\infty} p(\theta_i | \mu(\theta_i, \Sigma(\theta_i)) \ d\theta_i = 1$, and $\gamma^{\text{pred}}_{ij}(\theta_i)$ is defined as,

$$\gamma^{\text{pred}}_{ij}(\theta_i) = \begin{cases} \gamma^{\text{pred}}_{ij}(\theta_i), & \text{if all the element in } \theta_i \text{ are positive} \\ 0, & \text{otherwise.} \end{cases} \hspace{1cm} (60)$$

To evaluate $\int \gamma^{\text{pred}}_{ij}(\theta_i) p(\theta_i | \mu(\theta_i, \Sigma(\theta_i)) \ d\theta_i,$ in Equation 59, we just need to take the average of the samples of $\gamma^{\text{pred}}_{ij}(\theta_i)$, in which $\theta_i$ are sampled from the untruncated Gaussian $p(\theta_i | \mu(\theta_i, \Sigma(\theta_i))$. Then, we multiply the average value by the factor $1 / N(\theta_i)$, to obtain the expected value of $\frac{1}{N(\theta_i)} \int_0^{+\infty} \gamma^{\text{pred}}_{ij}(\theta_i) p(\theta_i | \mu(\theta_i, \Sigma(\theta_i)) \ d\theta_i$, which is what we want. When we sample $\theta_i$ from $p(\theta_i | \mu(\theta_i, \Sigma(\theta_i))$, $N(\theta_i)$ can be approximately evaluated by the ratio between the number of $\theta_i$ samples within the range $(0, \infty)$ and the total number of $\theta_i$ samples. For the simple analytic model in Section “Model with analytic solution”, $N(\theta_i=1,2)$ are all 1. For the voriconazole model in Section “Model with ordinary differential equations”, there is just one Gaussian mixture, $N(\theta_i=1)$ is about 0.96.

Results with log-normal transformed parameters

We realize that it is very common in PKs to use log-normal distributions, which avoids the truncated Gaussian issue. For the same voriconazole model in Section “Voriconazole Model”, that is to say, we do analysis for all the seven primary (structural) normal distributed
parameters \((K_a, V_{max0}, K_m, V_{c0}, F_{A1}, K_{cp}, K_{pc})\). This means Equations 42–46 become,

\[
V_m = e^{V_{max0} \times wt^{0.75}},
\]

\[
V = e^{V_{c0} \times wt},
\]

and the ODEs for any subject \(i\) are:

\[
\frac{dx_1}{dt} = -e^{K_a} \times x_1,
\]

\[
\frac{dx_2}{dt} = -e^{K_a} \times x_1 + \mu(n(t)) \times V_m (t) \times x_2 - e^{K_{cp}} \times x_2 + e^{K_{pc}} \times x_3,
\]

\[
\frac{dx_3}{dt} = e^{K_{cp}} \times x_2 - e^{K_{pc}} \times x_3.
\]

We define the seven log-normal transformed primary parameter vector for subject \(i\) as,

\[
\hat{\theta}_i = (e^{K_a}, e^{V_{max0}}, e^{K_m}, e^{V_{c0}}, e^{F_{A1}}, e^{K_{cp}}, e^{K_{pc}})^T,
\]

and we created 50 samples of \(\hat{\theta}_i\) such that their mean and standard deviation are about the same as the ‘true’ mean and standard deviation in Table 2. We re-simulated the new dataset\(^5\) based on these 50 samples.\(^5\) Note that in this model we do not use Gaussian mixture, so from hereafter we will neglect the superscript \(\hat{\theta}\) for the mixture model.

Note that the often called “typical values” (we use \(tv\) as prefix) in this case, are the exponential of the Gaussian mean \(\mu = (\mu_{K_a}, \mu_{V_{max0}}, \mu_{K_m}, \mu_{V_{c0}}, \mu_{F_{A1}}, \mu_{K_{cp}}, \mu_{K_{pc}})\), that is,

\[
e^\mu = (e^{\mu_{K_a}}, e^{\mu_{V_{max0}}}, e^{\mu_{K_m}}, e^{\mu_{V_{c0}}}, e^{\mu_{F_{A1}}}, e^{\mu_{K_{cp}}}, e^{\mu_{K_{pc}}})
\]

\(FIGURE 4\) VPC plots of RPEM using the \(x\)-variable binning and binless methods\(^4\) provided in the vpcResultsUI function in the \(R\) tidyvpc package,\(^9\) with 3000 replicas, for the voriconazole model and data described in Section “Model with ordinary differential equations”, and from the initial condition described in Section “Model with ordinary differential equations”. CI, confidence interval; RPEM, Randomized Parametric Expectation Maximization; VPC, Visual Predictive Check.
\[ \begin{align*}
\textbf{The “typical values” and the standard deviation for the new dataset are listed as the “True” rows in Table 4.}
\end{align*} \]

We randomly picked 22 initial conditions from ref. [52]. The 22 initial conditions are picked from ID number 1, 2, 10, 13, 15, 17, 20, 23, 25, 27, 3, 30, 33, 35, 37, 40, 43, 45, 47, 5, 50, and 7, by setting the corresponding primary parameters as the initial values and the covariance matrix \( \Sigma \) as an identity matrix. For this dataset NONMEM’s IMP method has no problem, so we include it in the comparison. Similar with what have been discussed in Sec. “Comparison among RPEM, SAEM, and QRPEM”, we let RPEM, IMP, SAEM, and QRPEM start from the 22 initial conditions and compare their speed and accuracy in reconstructing true typical values and the true \( \Sigma \). We use \( \sigma_{V_{\text{max}}}, \sigma_{K_{m}}, \sigma_{V_{c}}, \sigma_{F_{A1}}, \sigma_{K_{cp}}, \sigma_{K_{a}} \) to represent the standard deviations which are the square root of the diagonal elements in \( \Sigma \).

\[ \begin{align*}
\textbf{TABLE 4 The parameter reconstruction comparisons among RPEM, IMP, QRPEM, and SAEM.}
\end{align*} \]

\[ \begin{align*}
\textbf{Method (% error/parameter)} & \quad n_{V_{\text{max}}} & n_{V_{c}} & n_{K_{m}} & n_{F_{A1}} & n_{K_{cp}} & n_{K_{a}} \\
\textbf{True} & 2.14 & 8.48 & 9.48 & 1.15 & 0.73 & 1.60 & 1.19 \\
\textbf{RPEM (15.3%)} & 2.63 & 8.69 & 10.44 & 1.13 & 0.74 & 1.86 & 1.23 \\
& \pm 0.53 & \pm 1.96 & \pm 2.77 & \pm 0.11 & \pm 0.01 & \pm 0.48 & \pm 0.12 \\
\textbf{IMP (10.7%)} & 2.34 & 8.53 & 10.23 & 1.10 & 0.74 & 2.02 & 1.30 \\
& \pm 0.20 & \pm 0.83 & \pm 1.17 & \pm 0.08 & \pm 0.01 & \pm 0.33 & \pm 0.08 \\
\textbf{QRPEM* (11.0%)} & 2.41 & 8.62 & 10.37 & 1.10 & 0.74 & 2.02 & 1.30 \\
& \pm 0.21 & \pm 0.76 & \pm 1.08 & \pm 0.08 & \pm 0.01 & \pm 0.31 & \pm 0.07 \\
\textbf{SAEM (35.2%)} & 4.56 & 10.26 & 12.62 & 1.28 & 0.74 & 1.41 & 1.17 \\
& \pm 5.83 & \pm 3.09 & \pm 4.40 & \pm 0.14 & \pm 0.01 & \pm 0.46 & \pm 0.20 \\
\textbf{Method (% error/sigma)} & \sigma_{V_{\text{max}}} & \sigma_{K_{m}} & \sigma_{V_{c}} & \sigma_{F_{A1}} & \sigma_{K_{cp}} & \sigma_{K_{a}} \\
\textbf{True} & 0.33 & 0.41 & 0.41 & 0.15 & 0.10 & 0.42 & 0.55 \\
\textbf{RPEM (27.8%)} & 0.33 & 0.37 & 0.35 & 0.11 & 0.05 & 0.24 & 0.56 \\
& \pm 0.11 & \pm 0.05 & \pm 0.09 & \pm 0.02 & \pm 0.01 & \pm 0.13 & \pm 0.10 \\
\textbf{IMP (27.8%)} & 0.29 & 0.46 & 0.22 & 0.08 & 0.08 & 0.19 & 0.60 \\
& \pm 0.04 & \pm 0.01 & \pm 0.01 & \pm 0.01 & \pm 0.01 & \pm 0.07 & \pm 0.06 \\
\textbf{QRPEM* (27.3%)} & 0.31 & 0.46 & 0.22 & 0.08 & 0.08 & 0.19 & 0.61 \\
& \pm 0.04 & \pm 0.01 & \pm 0.01 & \pm 0.01 & \pm 0.01 & \pm 0.06 & \pm 0.05 \\
\textbf{SAEM (24.9%)} & 0.36 & 0.33 & 0.46 & 0.13 & 0.09 & 0.34 & 0.54 \\
& \pm 0.22 & \pm 0.08 & \pm 0.07 & \pm 0.03 & \pm 0.01 & \pm 0.18 & \pm 0.15 \\
\end{align*} \]

Note: The results are obtained by averaging the results from the 22 randomly picked initial conditions. The averaged values are listed at the first row of each method, and the standard deviation (which may be served as the estimation of the standard error for the parameter) are listed at the second row of each method and begin with \( \pm \) (e.g., for RPEM, the averaged \( n_{V_{\text{max}}} \) is 2.63, the standard deviation is 0.53). The overall averaged percentage errors from all the true typical values and the corresponding standard deviations are listed in the parentheses correspondingly. \( QRPEM^* \) means we manually stop QRPEM at the 50th iteration.

Abbreviations: IMP, Importance Sampling Method; QRPEM, Quasi-Random Parametric Expectation Maximization; RPEM, Randomized Parametric Expectation Maximization; SAEM, Stochastic Approximation Expectation Maximization.

In Figure 5, we list the speed comparison among RPEM, IMP, QRPEM, and SAEM. The timing of the runs is represented as data by solid symbols. We see that the speed of RPEM and IMP are comparable. On average RPEM took 32 s, IMP took 36 s, and both of them mostly finished in less than 40 s. IMP’s results are concentrated around 36 s, this is because due to its stopping criterion, for this model and dataset, it always stops at the 50th iterations. For RPEM, it mostly stops between the 40th and 70th iteration, so its time cost varies a little bit more than IMP. For the SAEM runs, they finished between 90 s and 210 s. For QRPEM, its stopping criterion not only checks the convergence of the likelihood, but also the convergence of each parameter. For this model and dataset, in terms of the likelihood, QRPEM always converge at about 50th iteration just like IMP. However, some parameters, such as \( K_{0}, K_{cp}, \) and \( V_{c0} \) continue to change even if the likelihood already converged at around 1300. Therefore, QRPEM continues to run for several hundred iterations, and this made QRPEM runs took longer than RPEM, IMP, and SAEM. Considering QRPEM’s convergence criterion...
might be too strict, we manually stopped the runs at the 50th iteration where the likelihood were already converged, and we report the time cost of them which are labeled as QRPEM*. We can see that for QRPEM*, it only took around 15s. This can be understood as RPEM uses 600 samples per subject whereas QRPEM uses 300. Therefore, per iteration, QRPEM should take only half of the time RPEM takes. RPEM usually took 50 iterations to finish, and QRPEM* took 50 iterations, so for the same number of iterations, QRPEM* should take about half of the time RPEM takes. Overall, we find that if we let all the algorithms stop naturally based on their own stopping criterion, the speed of RPEM and IMP are very similar and are among the fastest, whereas SAEM and QRPEM on average can be about 5 to 10 times slower. However, if we manually stop QRPEM at the 50th iteration while the convergence of the likelihood is already reached, QRPEM only took about half of the time RPEM and IMP took.

In Figure 6, we show RPEM, IMP, SAEM, and QRPEM’s abilities in reconstructing the true typical parameters and the corresponding standard deviations. Here, we use the results of QRPEM*, which means we manually stop QRPEM at the 50th iteration because the convergence of likelihoods are already reached. We find that almost all the true typical parameters and the standard deviations are covered by RPEM within the 90% CI. Out of the total 14 population parameters, namely $t_v K_{c_p}$, $t_v V_{c}$, $t_v K_c$, $t_v K_{p_c}$, $t_v K_{a}$, $t_v V_{max}$, $t_v K_m$, $t_v V_{c_0}$, $t_v K_{a_1}$, $t_v K_{p_c}$, $K_c$, $K_{max}$, $K_m$, $V_{c_0}$, $F_{Al}$, $K_{p_c}$, $F_{A_1}$, and $K_{p_c}$, RPEM missed three of them, that is, $t_v F_{A_1}$, $t_v V_{c_0}$, and $F_{A_1}$ are outside the 90% CI. IMP and QRPEM* are very similar. Within the 90% CI, they missed seven out of 14, that is, $t_v F_{A_1}$, $t_v K_{p_c}$, $t_v V_{max}$, $t_v F_{Al}$, $t_v K_m$, $t_v V_{c_0}$, $F_{A_1}$, and $K_{p_c}$. For SAEM, within the 90% CI, it only missed $F_{A_1}$ out of the 14. However, the results of SAEM represent more randomness than other methods, as can be seen from the data points are more scattered than RPEM, IMP, and QRPEM*. This indicates that, to get the correct typical values and the standard deviations, one may need to run SAEM for more times than RPEM, IMP, and QRPEM*. Considering each SAEM run may take the order of 100s, multiple SAEM runs may take the order of 1000s.

In Table 4, similar with Table 2 and the discussion in Section “Comparison among RPEM, SAEM, and QRPEM”, we quantify the results obtained in Figure 6, and we show the parameter reconstruction comparisons among RPEM, IMP, QRPEM*, and SAEM. Again, QRPEM* means we manually stop QRPEM at the 50th iteration. The results for each of the 14 population parameters, are obtained by averaging the results from the 22 randomly picked initial conditions. The averaged values for each of the 14 population parameters, are listed at the first row of each method, and the standard deviation (they may be served as the estimation of the standard error for each of the population

**FIGURE 5** The speed comparison among RPEM, IMP, QRPEM, and SAEM for the voriconazole model. QRPEM* means we manually stop QRPEM at the 50th iteration. The timing of the runs is represented as data by solid symbols. The definition of the box with data overlap is the same as in Figure 1. IMP, Importance Sampling Method; QRPEM, Quasi-Random Parametric Expectation Maximization; RPEM, Randomized Parametric Expectation Maximization; SAEM, Stochastic Approximation Expectation Maximization.

**FIGURE 6** The comparison among QRPEM* (black), SAEM (red), IMP (blue), and RPEM (green) in terms of the ability of reconstructing the typical values $\theta_\text{av}$ (labels begin with $t_v \cdot$) and the standard deviation of $\Sigma$ (labels begin with $\sigma_\cdot$). QRPEM* means we manually stop QRPEM at the 50th iteration. The leftmost symbol is the true value. The definition of the box with data overlap is the same as in Figure 1. IMP, Importance Sampling Method; QRPEM, Quasi-Random Parametric Expectation Maximization; RPEM, Randomized Parametric Expectation Maximization; SAEM, Stochastic Approximation Expectation Maximization.
parameters) are listed at the second row of each method and begin with ±. The overall averaged percentage errors from all the true typical values and the corresponding true standard deviations are listed in the parentheses after the name of each method. We can see that RPEM, IMP, and QRPEM* are similar. For the estimated typical values, IMP and QRPEM* are about 11% off (10.7% and 11.0%) from the true typical values, while RPEM is 15.3% off. For the estimated standard deviations of the population parameters, RPEM, IMP, and QRPEM* are all around 27% (27.8%, 27.8%, and 27.3%) off. For SAEM, due to its randomness in evaluating the population parameters, its estimated typical values are on average 35.2% off from the true values, this is less accurate than RPEM, IMP, and QRPEM*. However, SAEM’s estimated standard deviation is 24.9% off, and this is slightly better than RPEM, IMP, and QRPEM*.

The standard error estimation and the predictions are similar, with those already discussed in Section “Standard error” and Section “Predictions”, so we do not repeat them here. Note that for standard error of the typical values, one need to first obtain the standard error for each of the seven elements in the Gaussian mean \( \mu \) as discussed in Section “Standard error”, and then do error propagation and get the standard error for all the seven typical values.

### Scalability

Because RPEM is equipped with MPI, we also run RPEM from an initial condition on Agave supercomputer cluster at Arizona State University to test its scalability. Overall, when the number of CPU cores are less than 100 (which covers the range from a laptop to high-end personal desktop nowadays), the efficiency of RPEM is around 90%. As the number of cores goes beyond 100, because the samples at E-step needs to be calculated on each core become fewer and fewer, the real computation time in solving ODEs is decreased, so the percentage of MPI communication time in the total time increased. Therefore, the efficiency decreased to about 70% when the number of CPU cores beyond 200. However, if the ODEs are complicated enough and require many samples, such that the computing time on each core is much more than the MPI communication time, RPEM will always have reasonably high efficiency. Therefore, RPEM is ready for models which are too complicated for a personal computer.

### SUMMARY AND OUTLOOK

In this paper, we presented our quantum Monte Carlo inspired novel MCPEM algorithm, which we call RPEM. RPEM distinguishes itself from other MCPEM algorithms mostly in the M-step, which uses the Metropolis–Hastings algorithm and samples both continuous variables and discrete variables at the same time efficiently. Therefore, unlike other MCPEM algorithms which uses biased estimators, both RPEM’s E-step and the M-step uses unbiased estimators which will lead to fast convergence and accurate results.

With concrete examples of a one-compartment two-mixture analytic model and a voriconazole model with ODEs, and by comparing RPEM with IMP, QRPEM, and SAEM, we show that no matter if the population parameters are normal distributed or log-normal distributed, RPEM is indeed not only a fast, but is also an accurate MCPEM algorithm. For the voriconazole model with normal distributed population parameters, which IMP cannot handle, on average, RPEM is about three to four times faster than QRPEM and SAEM, and more accurate than them. For the voriconazole model with log-normal distributed population parameters, RPEM, IMP, and QRPEM are more accurate than SAEM in estimating the typical values, and the speed of RPEM and IMP are among the fastest, on average, RPEM is about five to 10 times faster than SAEM and QRPEM. If we manually stop QRPEM at the 50th iteration, then the speed of QRPEM is the fastest. We also show that RPEM is a scalable high performance MCPEM which can be run on supercomputers for more complicated models.

In future work, we will further assess and validate RPEM on more complex data and models, test several approaches to rapidly arrive at the optimal number of mixing components, and further develop or implement techniques to avoid local maxima.

We wish as a newly developed MCPEM method, that RPEM can be a useful addition to the current MCPEM methods. We welcome ideas, suggestions, and cooperating opportunities from the community.

### AUTHOR CONTRIBUTIONS

R.C. wrote the manuscript. R.C., A.S., K.N., and M.N.N. designed the research. R.C., A.S., A.K., K.N., M.T., S.H., R.G., and M.N.N performed the research. R.C., A.S., A.K., K.N., M.T., S.H., R.G., J.O., W.Y., and M.N.N. analyzed the data. R.C., A.S., K.N., M.T., S.H., R.G., and M.N.N. contributed new reagents/analytical tools.

### ACKNOWLEDGMENTS

This work was funded in part by U01 1FD006549 (Neely, PI). R.C. thanks Professor Kevin E. Schmidt at Arizona State University for inspiring discussions and insights. R.C. acknowledges Research Computing at Arizona State University for providing HPC and storage resources that have contributed to the research results reported within this paper. R.C. thanks the help and discussions from James Craig and Mark Sale at Certara about using vpcResultsUI function in the R tidyvpc package\(^49\) for the visual predictive
check and the proofread of the manuscript. R.C. also thanks the help and comments from Robert J. Bauer at ICON in setting up the model file for the NONMEM’s IMP method. Furthermore, R.C. acknowledges the valuable help received from Fortran-lang community. In particular, R.C. thanks Ondřej Čertík, John Campbell, Milan Curcic, Martin Diehl, Steve Kargl, Steve Lionel (Doctor Fortran), Bharat Mahajan, Vincent Magnin, Arjen Markus, Panagiotis Papasotiriou, Ivan Pribec, Vivek Rao, Brad Richardson, Simon Rowe, Amir Shahmoradi, Michal Szymanski, Theodore-Thomas Tsikas, John S. Urban, and Yi Zhang for various and generous help in modern Fortran coding and the ODE solvers used in RPEM code. Last but not least, we thank the referees and the editors for valuable comments and suggestions.

CONFLICT OF INTEREST STATEMENT

This work has no conflict of interest/competing interests.

ORCID

Rong Chen https://orcid.org/0000-0001-8717-094X
Alona Kryshchenko https://orcid.org/0000-0002-5049-0293
Shuhua Hu https://orcid.org/0000-0002-4714-7478
Romain Garreau https://orcid.org/0000-0002-6605-4808
Julian Otalvaro https://orcid.org/0000-0001-5202-1645
Walter Yamada https://orcid.org/0000-0003-3512-9202
Michael N. Neely https://orcid.org/0000-0002-1675-8276

REFERENCES

1. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the em algorithm. J R Stat Soc B Methodol. 1977;39:1-22. doi:10.1111/j.2517-6161.1977.tb01600.x
2. Schumitzky A. EM algorithms and two stage methods in pharmacokinetic population analysis. Advanced Methods of Pharmacokinetic and Pharmacodynamic Systems Analysis. Vol 2. Plenum Press; 1995:145.
3. Walker S. An EM algorithm for nonlinear random effects models. Biometrics. 1996;52:934.
4. Wang X, Schumitzky A, D’Argenio DZ. Nonlinear random effects mixture models: maximum likelihood estimation via the em algorithm. Comput Stat Data Anal. 2007;51:6614-6623.
5. Wu CFJ. On the convergence properties of the em algorithm. Ann Stat. 1983;11:95.
6. Tseng P. An analysis of the em algorithm and entropy-like proximal point methods. Mathemat Operat Res. 2004;29:27-44. doi:10.1287/moor.1030.0073
7. D’Argenio DZ, Schumitzky A, Wang X. ADAPT 5 User’s Guide: Pharmacokinetic/Pharmacodynamic Systems Analysis Software. Biomedical Simulations Resource; 2009.
8. Bauer RJ. Advanced Population Analysis Features in the S-ADAPT/MCPEM Program. Accessed December 22, 2023. https://www.page-meeting.org/?abstract=1111 and https://bmsr.usc.edu/downloads/s-adapt/
9. Sheiner LB, Beal SL. Evaluation of methods for estimating population pharmacokinetic parameters. I. michaelis-menten model: routine clinical pharmacokinetic data. J Pharmacokinet Biopharm. 1980;8:553-571.
10. Cerbera LLP. Phoenix NLME. https://www.cerbera.com/software/pkpd-modeling-and-simulation/phoenix-nlme/ 2023.
11. Leary RH, Dunlavey M. QRPEM, A Quasi-Random Parametric EM Method. 2012.
12. Craig J, Tomashhevskiy M, Nazarov V, Fred S. Certara. RsNLME: Pharmacometric modeling in R (2021), r package version 1.1.0.
13. Delyono B, Lavielle M, Moulines E. Convergence of a stochastic approximation version of the em algorithm. Ann Stat. 1999;27:94.
14. Kuhn E, Lavielle M. Coupling a stochastic approximation version of EM with an MCMC procedure. In: Vermeulen N, de Saporta B, eds. ESAIM: Probability and Statistics, Vol 8; 2004: 115. https://www.esaim-ps.org/articles/ps/abs/2004/01/p0312/p0312.html
15. Kuhn E, Lavielle M. Maximum likelihood estimation in nonlinear mixed effects models. Comput Stat Data Anal. 2005;49:1020-1038.
16. Lavielle M, Mboogning C. An improved saem algorithm for maximum likelihood estimation in mixtures of nonlinear mixed effects models. Stat Comput. 2014;24:693-707.
17. Lomnitz-Adler J, Pandharipande V, Smith R. Monte Carlo calculations of triton and 4He nuclei with the Reid potential. Nuclear Physics A. 1981;361:399-411.
18. Ceperley DM. Path integrals in the theory of condensed helium. Rev Mod Phys. 1995;67:279-355.
19. Carlson J, Gandolfi S, Pederiva F, et al. Quantum Monte Carlo methods for nuclear physics. Rev Mod Phys. 2015;87:1067-1118.
20. Chen R. Path integral quantum Monte Carlo method for light nuclei. PhD thesis, Arizona State University 2020.
21. Chen R, Schmidt KE. Path-integral quantum monte carlo calculations of light nuclei. Phys Rev C. 2022;106:044327.
22. Wang X, Schumitzky A, D’Argenio DZ. Population pharmacokinetic/pharmacodynamic mixture models via maximum a posteriori estimation. Comput Stat Data Anal. 2009;53:3907-3915.
23. Owen JS, Fiedler-Kelly J. Introduction to Population Pharmacokinetic/Pharmacodynamic Analysis with Nonlinear Mixed Effects Models. John Wiley & Sons; 2014.
24. Wiens DP. Robust designs for dose–response studies: model and labelling robustness. Comput Stat Data Anal. 2021;158:107189.
25. Brown PT, Joshi C, Joe S, Rue H. A novel method of marginalisation using low discrepancy sequences for integrated nested laplace approximations. Comput Stat Data Anal. 2021;157:107147.
26. Li R, Reich BJ, Bondell HD. Deep distribution regression. Comput Stat Data Anal. 2021;159:107203.
27. Shahmoradi A. ParaMonte: Plain Powerful Parallel Monte Carlo and MCMC Library for Python, MATLAB, Fortran, C++. Accessed December 22, 2023. https://fortran-lang.discourse. group/t/is-there-some-good-multivariate-gaussian-generator/1905/2?u=cruquantum
28. Hammersley JM, Handscomb DC. Monte Carlo Methods. Chapman and Hall; 1983.
29. Kalos MH, Whitlock PA. Monte Carlo Methods. Wiley; 2008.
30. Negele JW, Orland H. Quantum Many-Particle Systems. Westview Press; 1998.
SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Chen R, Schumitzky A, Kryshchenko A, et al. RPEM: Randomized Monte Carlo parametric expectation maximization algorithm. CPT Pharmacometrics Syst Pharmacol. 2024;13:759-780. doi:10.1002/psp4.13113