A Matlab Framework for Estimation of NLME Models using Stochastic Differential Equations

Applications for estimation of insulin secretion rates

— POSTPRINT —

Authors: Stig B. Mortensen[1,*], Søren Klim[1,2], Bernd Dammann[1], Niels R. Kristensen[2], Henrik Madsen[1] and Rune V. Overgaard[2]

[1] Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark
[2] Novo Nordisk A/S, Denmark
[*] Corresponding author: sbm@imm.dtu.dk

Abstract: The non-linear mixed-effects model based on stochastic differential equations (SDEs) provides an attractive residual error model, that is able to handle serially correlated residuals typically arising from structural mis-specification of the true underlying model. The use of SDEs also opens up for new tools for model development and easily allows for tracking of unknown inputs and parameters over time. An algorithm for maximum likelihood estimation of the model has earlier been proposed, and the present paper presents the first general implementation of this algorithm. The implementation is done in Matlab and also demonstrates the use of parallel computing for improved estimation times. The use of the implementation is illustrated by two examples of application which focus on the ability of the model to estimate unknown inputs facilitated by the extension to SDEs. The first application is a deconvolution-type estimation of the insulin secretion rate based on a linear two-compartment model for C-peptide measurements. In the second application the model is extended to also give an estimate of the time varying liver extraction based on both C-peptide and insulin measurements.

Keywords: non-linear mixed-effects modelling; SDE; Kalman smoothing; deconvolution; state-estimation; parameter tracking; MatlabMPI; PK/PD.

Published in:

Journal of Pharmacokinetics and Pharmacodynamics (2007) 34:623-642.

INTRODUCTION

The non-linear mixed-effects (NLME) model based on ordinary differential equations (ODEs) is a widely used method for modelling pharmacokinetic/pharmacodynamic (PK/PD) data [1], since the model enables the variation to be split into inter- and intra-individual variation. It is, however, a well known problem that this model class has a too restricted residual error structure, as it assumes that the residuals are uncorrelated white noise. This assumption applies well to the expected distribution of assay error, but it is unfortunately a crude simplification to assume that the assumption also applies to the remaining sources of error [2]. Other important sources of error may arise from structural model mis-specification or unpredictable random behavior of the underlying process, which both result in serially correlated residual errors. Previous work with simulation of more complex error structures has shown that ignoring the serial correlation may lead to biased estimates of the variance components of the model or all population parameter estimates depending on the error structure [3].

A powerful way to deal with these problems is to introduce stochastic differential equations (SDEs) in the model setup. SDEs are an extension to ODEs and facilitate the ability to split the intra-individual error into two fundamentally different types: (1) *serially uncorrelated measurement error*, which is typically mainly caused by assay error and (2) *system error*, which may be caused by model mis-specifications, simplifications or true random behavior of the system.

Apart from providing a statistically more adequate model setup, the SDEs also allow new tools for the modeller. The SDE approach results in a quantitative estimate of the amount of system and measurement noise, and it can therefore also be used as a tool for model validation. If no significant system noise is found to be present, this indicates that the proposed model structure gives a suitable description of the data. However, if significant system noise is found, it can be estimated and may be used to identify a possible remaining model structure, since aspects which are not explicitly modelled will give rise to system noise. It is important to emphasize that this relation does not hold the other way around, since system noise may also arise from true unmodellable random behavior of the system, and estimated system noise may thus not be seen as evidence of an insufficient model structure. A detailed iterative scheme for model development based on SDEs has been described in [4]. Another important advantage of the SDE approach is the inherent confidence intervals for system states. This is facilitated by the estimation of system noise, and thus follows as a natural part of the model specification.

Several programs exist for modelling based on SDEs. The first implementations focused on single subject modelling, such as CTSM (Continuous Time Stochastic Modelling) [5]. CTSM is in fact also able to use multiple individuals for estimation of structural parameters, but this is done using a naive pooled likelihood function where no inter-individual variance components are estimated. Later research has also made it possible to include SDEs in population modelling by using an approximation algorithm of the likelihood function with SDEs for the widely used

non-linear mixed-effects model. This algorithm is described in [6] and is based on the use of the Extended Kalman Filter to estimate conditional densities of each observation to form the individual likelihood function. The population likelihood function is then approximated based on the First-Order Conditional Estimation (FOCE) method. It has been shown in [7] how this algorithm for estimation of SDEs can be used in NONMEM [8], but this is by no means a trivial programming task to set up for a given model. It requires a modified data file and an implementation of the Kalman filter within the NONMEM control stream. Moreover, the NONMEM implementation cannot be used to form Kalman smoothing estimates, which is an important feature of the SDE approach, where all data is used to give optimal estimates at each sampling point.

This paper will present the first prototype implementation of a general software tool for estimation of NLME models based on SDEs. The implementation has been made in Matlab and it makes experimentation with the new modelling approach readily available. The flexibility of the modelling approach will be demonstrated by two examples of applications. In the first example the model is used for stochastic deconvolution to estimate insulin secretion rates in 12 type II diabetic patients and in the second example the model is used to estimate/track the time variant behavior of the liver extraction rate for the same individuals.

THEORY

This section contains an overview of the theory for population modelling using non-linear mixed-effects models based on SDEs. It will present the state space model for individual modelling and how this can be extended to incorporate SDEs. The parameters of the population model are estimated with a maximum likelihood (ML) approach by first defining an individual likelihood function, which forms the basis for the population likelihood function. The individual likelihood function is evaluated on the basis of the Extended Kalman Filter (EKF), and this will also be outlined. A more detailed description of the estimation algorithm can be found in [6]. To ease notation, all vectors and matrices are written using a bold font.

A mixed-effects model is used to describe data with the following general structure

$$\boldsymbol{y}_{ij}, \quad i = 1, ..., N, \quad j = 1, ..., n_i \tag{1}$$

where $\boldsymbol{y}_{ij}$ is a vector of measurements at time t_{ij} for individual i, N is the number of individuals and n_i is the number of measurements for individual i. Note that the number of measurements for each individual may vary. In a mixed-effects model the variation is split into intra-individual variation and inter-individual variation, which is modelled by a first and second stage model.

First stage model

The first stage model for an NLME model with ODEs can be written in the form of a state space model. A state space model consists of two parts, namely a set of continuous state equations defining the dynamics of the system and a set of discrete measurement equations, which defines a functional relationship between the states of the system and the measurements obtained. In the general form the state space equations are written as

$$d\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t, t, \boldsymbol{\phi}_i)dt \quad (2)$$

$$\boldsymbol{y}_{ij} = \boldsymbol{h}(\boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}, t_{ij}, \boldsymbol{\phi}_i) + \boldsymbol{e}_{ij} \quad (3)$$

where t is the continuous time variable and the states of the model and the optional inputs at time t are denoted $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$ respectively. The input $\boldsymbol{u}_{ij}$ is typically frequently sampled covariates such as body temperature etc. which may affect the system, or a variable indicating an interaction with the system such as an intravenous infusion. Both the state, measurement and input can be multi-dimensional, and are in such cases thus represented by a vector at time t_{ij}. The individual model parameters are denoted $\boldsymbol{\phi}_i$ and finally $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$ are the two possibly non-linear functions defining the model. Measurements are assumed observed with an uncorrelated Gaussian measurement error. The variance of the error may depend on both state, input, time and individual parameters, that is $\boldsymbol{e}_{ij} \in N(\boldsymbol{0}, \boldsymbol{\Sigma}(\boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}, t_{ij}, \boldsymbol{\phi}_i))$.

It is important to draw attention to the concept of states, as this is essential to the understanding of the model setup. States are generally not directly observable or at best only observable through measurement noise. The actual relation between measurements and states is defined in the measurement equation by the function $\boldsymbol{h}(\cdot)$. A state can represent many different aspects of the system of interest, e.g. concentrations or amounts in compartments, a volume, a parameter with unknown time varying behavior, or an input to the system that we wish to estimate. The state space formulation is thus a very flexible form of model specification, and the use of the state space model will be illustrated with the applications presented later on in this paper.

Extending the first stage model with SDEs

In the ordinary state space model, noise is only allowed to enter through the measurement equation, see Eq. (3). The result is that error due to model mis-specification or true random fluctuations of the states is absorbed into the measurement error term and hence may give rise to correlated residuals. To allow for error to originate from the system specification, a stochastic term is added to the system equation. This results in a stochastic state space equation defined as follows

$$d\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t, t, \boldsymbol{\phi}_i)dt + \boldsymbol{\sigma}_\omega(\boldsymbol{u}_t, t, \boldsymbol{\phi}_i)d\boldsymbol{\omega}_t \quad (4)$$

$$\boldsymbol{y}_{ij} = \boldsymbol{h}(\boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}, t_{ij}, \boldsymbol{\phi}_i) + \boldsymbol{e}_{ij} \quad (5)$$

where $\boldsymbol{\omega}_t$ is a standard Wiener process defined by $\boldsymbol{\omega}_{t_2} - \boldsymbol{\omega}_{t_1} \in N(\boldsymbol{0}, |t_2 - t_1|\boldsymbol{I})$. The entire part $\boldsymbol{\sigma}_\omega(\boldsymbol{u}_t, t, \boldsymbol{\phi}_i)d\boldsymbol{\omega}_t$ is called the diffusion term and describes the stochastic part of the system and $\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t, t, \boldsymbol{\phi}_i)dt$ is called the drift term and describes the deterministic part. Together the drift and diffusion terms define the stochastic dynamics of the system.

By looking at the formulation of the extended first stage model, it is seen that noise is now allowed to enter in two places, namely as system noise via the diffusion term and as measurement noise. It is noted that if no system noise is present, the model will reduce into the standard ODE case, and this also ensures that physiological interpretation of structural parameters is preserved with the use of an SDE model.

Second stage model

The second stage model for the individual parameters describes the variation of the individual parameters $\boldsymbol{\phi}_i$ between individuals and can be defined in a number of ways, each with different properties. In the present work it has been chosen to use

$$\boldsymbol{\phi}_i = g(\boldsymbol{\theta}, \boldsymbol{Z}_i) \cdot exp(\boldsymbol{\eta}_i) \quad (6)$$

where $\boldsymbol{\eta}_i$ is the multivariate random effect parameter for the ith individual, which is assumed Gaussian distributed with mean zero and covariance $\boldsymbol{\Omega}$: $\boldsymbol{\eta}_i \in N(\boldsymbol{0}, \boldsymbol{\Omega})$. The fixed effect parameter of the NLME model is $\boldsymbol{\theta}$, which is also sometimes referred to as the structural parameter or population parameter. The second stage model in Eq. (6) includes an optional covariate $\boldsymbol{Z}_i$. This can be used to include individually measurable covariates such as height, weight etc. that could affect $\boldsymbol{\phi}_i$. The chosen formulation of the 2nd stage model restricts variations in $\boldsymbol{\eta}_i$ from changing the sign of $g(\boldsymbol{\theta}, \boldsymbol{Z}_i)$ which is typically an advantage as $\boldsymbol{\phi}_i$ may be used as parameter for a variance or other sign-sensitive parameters. Moreover the resulting distribution of the individual parameters is log-Gaussian, as is often the case when dealing with PK/PD models. The second stage model in Eq. (6) may easily be replaced if other model structures are needed, and this can be done without yielding any changes to the final population likelihood function as long as $\boldsymbol{\eta}_i$ is still assumed to have a Gaussian distribution.

Maximum likelihood estimation of the NLME model with SDEs

The full set of parameters to be estimated for the final NLME model with SDEs are the matrices $\boldsymbol{\Sigma}$, $\boldsymbol{\sigma}_\omega$, $\boldsymbol{\Omega}$ and the fixed effect parameters in the vector $\boldsymbol{\theta}$. The three matrices are usually fixed

to some degree so that only the diagonals or other partial structure remains to be estimated.

The estimation of model parameters is based on a first stage likelihood function, which is formed as a product of probabilities for each measurement. Due to the assumption of correlated residuals with the inclusion of the Wiener process, it is necessary to condition on the previous measurements to define the probability density of each measurement. In the approach chosen here, this is done by assuming that the conditional densities for the states are Gaussian and thus fully described by the state-prediction and the state prediction variance for each observation. These can be found using the Extended Kalman filter, which gives the unbiased minimum variance estimate of the evolution of the model states [9]. This will hold exactly for the linear case but only as an approximation in the non-linear case. The assumptions for the EKF can be examined by testing for a Gaussian distribution of the residuals and by testing for correctness of the estimated stochastic differential equations [10]. The prediction from the EKF is defined by

$$\hat{\boldsymbol{y}}_{i(j|j-1)} = E(\boldsymbol{y}_{ij}|\boldsymbol{\phi}_i, \boldsymbol{\Sigma}, \boldsymbol{\sigma}_\omega, \boldsymbol{u}_i, \mathcal{Y}_{i(j-1)}) \tag{7}$$

$$\boldsymbol{R}_{i(j|j-1)} = V(\boldsymbol{y}_{ij}|\boldsymbol{\phi}_i, \boldsymbol{\Sigma}, \boldsymbol{\sigma}_\omega, \boldsymbol{u}_i, \mathcal{Y}_{i(j-1)}) \tag{8}$$

where $\mathcal{Y}_{ij} = [\boldsymbol{y}_{i1}, ..., \boldsymbol{y}_{ij}]$ and this gives the conditional distribution of the one-step prediction error

$$\boldsymbol{\epsilon}_{ij} = \boldsymbol{y}_{ij} - \hat{\boldsymbol{y}}_{i(j|j-1)} \in N(\boldsymbol{0}, \boldsymbol{R}_{i(j|j-1)}) \tag{9}$$

The first stage likelihood function is calculated as the simultaneous density function for the ith individual

$$p_1(\mathcal{Y}_{in_i}|\boldsymbol{\phi}_i, \boldsymbol{\Sigma}, \boldsymbol{\sigma}_\omega, \boldsymbol{u}_i) = \left(\prod_{j=2}^{n_i} p(y_{ij}|\mathcal{Y}_{i(j-1)}, \cdot)\right) p(y_{i1}|\cdot) \tag{10}$$

$$\approx \prod_{j=1}^{n_i} \frac{\exp\left(-\frac{1}{2}\boldsymbol{\epsilon}_{ij}^T \boldsymbol{R}_{i(j|j-1)}^{-1} \boldsymbol{\epsilon}_{ij}\right)}{\sqrt{|2\pi \boldsymbol{R}_{i(j|j-1)}|}} \tag{11}$$

where conditioning on $\boldsymbol{\phi}_i$, $\boldsymbol{\Sigma}$, $\boldsymbol{\sigma}_\omega$ and $\boldsymbol{u}_i$ is denoted "$\cdot$".

Based on the first and second stage model density functions, the full non-linear mixed-effects likelihood function can now be defined. The second stage distribution is simply a multivariate Gaussian density denoted $p_2(\boldsymbol{\eta}_i|\boldsymbol{\Omega})$, and combining this with the first stage distribution results in the population likelihood function

$$L(\boldsymbol{\theta}, \boldsymbol{\Sigma}, \boldsymbol{\sigma}_\omega, \boldsymbol{\Omega}) = \prod_{i=1}^{N} \int p_1(\mathcal{Y}_{in_i}|\boldsymbol{\eta}_i, \boldsymbol{\theta}, \boldsymbol{\Sigma}, \boldsymbol{\sigma}_\omega, \boldsymbol{u}_i) p_2(\boldsymbol{\eta}_i|\boldsymbol{\Omega}) d\boldsymbol{\eta}_i \tag{12}$$

$$= \prod_{i=1}^{N} \int \exp(l_i) d\boldsymbol{\eta}_i \tag{13}$$

where l_i is the *a posteriori* log-likelihood function for the random effects of the ith individual given by

$$l_i = -\frac{1}{2} \sum_{j=1}^{n_i} \left(\boldsymbol{\epsilon}_{ij}^T \boldsymbol{R}_{i(j|j-1)}^{-1} \boldsymbol{\epsilon}_{ij} + \log |2\pi \boldsymbol{R}_{i(j|j-1)}| \right) - \frac{1}{2} \boldsymbol{\eta}_i^T \boldsymbol{\Omega}^{-1} \boldsymbol{\eta}_i - \frac{1}{2} \log |2\pi \boldsymbol{\Omega}| \tag{14}$$

The population likelihood function in Eq. (13) cannot be evaluated analytically, and therefore l_i is approximated by a second-order Taylor expansion, where the expansion is made around the value $\hat{\boldsymbol{\eta}}_i$ that maximizes l_i. At this optimum the first derivative $\boldsymbol{\nabla} l_i \big|_{\hat{\boldsymbol{\eta}}_i} = 0$ and the population likelihood function therefore reduces to

$$L(\boldsymbol{\theta}, \boldsymbol{\Sigma}, \boldsymbol{\sigma}_\omega, \boldsymbol{\Omega}) \approx \prod_{i=1}^{N} \left| \frac{-\boldsymbol{\Delta} l_i}{2\pi} \right|^{-\frac{1}{2}} \exp(l_i) \Big|_{\hat{\boldsymbol{\eta}}_i} \tag{15}$$

as shown in App. A. The approximation of the 2nd derivative $\boldsymbol{\Delta} l_i$ is done using the First-Order Conditional Estimation (FOCE) method, as it is also normally done in the NLME model based on ODEs. The objective function for parameter estimation is chosen as the negative log-likelihood function given as

$$-\log L(\boldsymbol{\theta}, \boldsymbol{\Sigma}, \boldsymbol{\sigma}_\omega, \boldsymbol{\Omega}) \approx \sum_{i=1}^{N} \left(\frac{1}{2} \log \left| \frac{-\boldsymbol{\Delta} l_i}{2\pi} \right| - l_i \right) \tag{16}$$

Kalman filtering

The Extended Kalman Filter plays a central role for working with the NLME model with SDEs as seen from the previous section. Therefore a brief introduction to the EKF will be given here, as well as to the three new types of state estimates made available. For a detailed description of the EKF algoritm please refer to [5, 6, 11, 12].

For linear state estimation problems the Kalman Filter will give an unbiased minimum variance state estimate. The solution can be derived explicitly using simple linear algebra, and hence the algorithm runs efficiently in a computer implementation. For non-linear problems it is necessary

to use another method for state estimation like that obtained by the Extended Kalman Filter, which has been used here. The Extended Kalman Filter is for the main part identical to the Kalman Filter, except for the state prediction which requires a solution to the non-linear differential system equations. This solution is obtained by a point-wise first order approximation and therefore, for non-linear systems, the EKF will only provide an approximate minimum variance estimate of the states. The EKF also runs slower due to the need for a numerical algorithm to solve the non-linear differential equations.

The Kalman Filter is a two-part algorithm consisting of *prediction* and *updating*, which iterates through all observations. In the prediction part the current estimated states and covariances are used to create predictions of the two first moments of the state and observation to a time point t_{ij} given the information at time $t_{i(j-1)}$. These predictions are denoted $\hat{\boldsymbol{x}}_{i(j|j-1)}$, $\hat{\boldsymbol{P}}_{i(j|j-1)}$, $\hat{\boldsymbol{y}}_{i(j|j-1)}$ and $\hat{\boldsymbol{R}}_{i(j|j-1)}$ respectively. Updating is performed at measurement time points, where the states and covariances are updated accordingly.

The updating is based on a compromise between the observation and current model state. In a situation where the model is good but the observations are dominated by measurement error, the state estimate should rely more on the model as opposed to fitting the observations. On the other hand, if the model is incomplete the states should rely more on the observations than the model. This trust in model versus observations is balanced by the Kalman gain, which is dependent on the magnitude of system noise $\boldsymbol{\sigma}_\omega$ and observation noise $\boldsymbol{\Sigma}$.

The initial conditions of the state and state covariance ($\hat{\boldsymbol{x}}_{i(1|0)}$ and $\hat{\boldsymbol{P}}_{i(1|0)}$) need to be specified for the Kalman filtering algorithm. The initial state can either be fixed or included in the likelihood function, whereas $\hat{\boldsymbol{P}}_{i(1|0)}$ for this implementation has been chosen to be estimated as the integral of the Wiener process and system dynamics over the first sample interval in accordance with the method used in [11].

A key feature of the SDE approach to population modelling is the ability to give improved estimates of the system states given the individual parameters and also to provide confidence bands for the states. Confidence bands at a timepoint t are directly given by the estimated state covariance matrix $\hat{\boldsymbol{P}}_{i(t|\ldots)}$ from the EKF, where t can be both at or between measurements. There are four types of state and state covariance estimates available when using the EKF, each of which differs in the way data is used. The four types are:

- Simulation estimate: $\hat{\boldsymbol{x}}_{i(j|0)}$, $\hat{\boldsymbol{P}}_{i(j|0)}$
  Provides an estimate of the state evolution for a repeated experiment, without updating based on measurements. This is an ODE-like estimate, but it also yields a confidence band for the state evolution.

- Prediction estimate: $\hat{\boldsymbol{x}}_{i(j|j-1)}$, $\hat{\boldsymbol{P}}_{i(j|j-1)}$
  The prediction is used here to give the conditional density for the next observation at time t_{ij} given the observations up to $t_{i(j|j-1)}$.

- Filtering estimate: $\hat{\boldsymbol{x}}_{i(j|j)}$, $\hat{\boldsymbol{P}}_{i(j|j)}$
  Best estimate at time t_{ij} given the observations up to time t_{ij}.
- Smoothing estimate: $\hat{\boldsymbol{x}}_{i(j|N)}$, $\hat{\boldsymbol{P}}_{i(j|N)}$
  Optimal estimate at time t_{ij} utilizing all observations both prior to and after time t_{ij}.

For a conventional ODE model the state is found by the simulation estimate, which is entirely given by the (possibly ML-estimated) initial state of the system. The covariance matrix for the states is $\mathbf{0}$ since no system noise is estimated. In other words the ODE model assumes that a new experiment will yield an identical outcome of the underlying system apart from observed measurement noise. By moving to SDEs, system noise is separated from measurement noise, thereby enabling the model to provide confidence bands for the realization of the states in a new experiment. By improving the model, the confidence bands for the states will become narrower and theoretically be zero if the true model is used and no random fluctuations in system states are present.

With SDEs three new types of estimates, apart from the simulation estimate, also become available. In the present setup the prediction estimate is used to give conditional Gaussian densities to form the likelihood function. The filter estimate is the best obtainable state estimate during the experiment, where the subsequent observations are not present. The third type of state estimate is the smoothed estimate. This provides the optimal state and state covariance estimate ($\hat{\boldsymbol{x}}_{i(j|N)}$ and $\hat{\boldsymbol{P}}_{i(j|N)}$) based on all obtained observations, both prior and subsequent to the time of interest. The smoothed estimate is therefore often the natural estimate of choice when studying the behavior of the system in *post hoc* analysis.

SOFTWARE IMPLEMENTATION

The estimation algorithm outlined in the previous section has been implemented in a Matlab framework called PSM (Population Stochastic Modelling). It is intended that this should work as a software prototype, in order to make further experimentation with the model setup easily available. The program may be obtained by addressing an email to the corresponding author.

Features

The implementation is designed to handle any non-linear mixed effects models using SDEs based on the general model definition in the previous section. The model specification is achieved through a set of Matlab functions written in m-files. A complete model specification consists of state dynamics $\boldsymbol{f}$ and diffusion term magnitude $\boldsymbol{\sigma}_\omega$, output function $\boldsymbol{h}$ and uncertainty $\boldsymbol{\Sigma}$, derivatives of state $d\boldsymbol{f}/dt$ and output $d\boldsymbol{h}/dt$, initial state $\boldsymbol{x}_0$, second stage model $\boldsymbol{g}$ and finally a variance function $\boldsymbol{\Omega}$ for the random effects. Each function is prepared to use all input arguments as specified by the model definition.

The implementation has been made in two versions. The first is able to handle the general non-linear case, and is thus based on the use of an algorithm for solving the differential equations in the EKF. It has been chosen to use `ode15s`, which is a Matlab built-in ODE solver. The second version is only able to solve linear systems, which will run significantly faster since it is based on an explicit solution of the differential equations.

The population parameters are estimated by maximizing of the population likelihood function given in Eq. (15). Maximization is performed using a publicly available Matlab implementation `ucminf` of a gradient search BFGS method with soft-line search and trust-region type monitoring of step length [13]. For additional performance it is possible to guide the optimization by providing an initial guess and boundaries for the parameters. The implementation is also able to assess parameter variance and correlation based on a numerical approximation of the Hessian of the likelihood functions [14].

Implementation details

In the evaluation of the population likelihood function it is necessary to evaluate the individual *a posteriori* log-likelihood function for each individual at its optimum, since a Taylor expansion is made around this point. Hence for one evaluation of the population likelihood function an optimization must be performed on each individual likelihood function. These optimizations only share the given population parameters and are therefore evaluated independently. This observation can be used to employ the use of parallel computing, where the individual optimizations are distributed to a number of CPUs.

Matlab does not have the option for parallel computing by default[1], but this can be made possible using external software. MatlabMPI[2] is a package developed at MIT and it enables parallel computing in Matlab by creating a set of scripts that is executed in separate processes. MatlabMPI uses message passing but it was found faster to pass all data and parameters through files. The individual calculation extracts its unique part of data by using its identifier number. The individual log-likelihood result is passed back into the leader thread by proper message passing to avoid deadlocks or race conditions. A shared memory environment is beneficial as message passing is implemented through shared files.

In order to illustrate the effect of parallel computation for population modelling, a model was setup and estimated on the basis of simulated data for 20 subjects. The resulting computation time is found in Table I and it can be seen that the computation time is reduced to a little less than one-fifth of the original using five CPUs. For this example overhead begins to dominate when using more than five CPUs, however for more computationally intensive models, the benefit of adding more CPUs is expected to be less affected by overhead.

For non-linear models a significant part of the computation time is spent in the prediction part

[1] A distributed toolbox for Matlab is under development by The MathWorks.
[2] J. Kepner, Parallel Programming with MatlabMPI, http://www.ll.mit.edu/MatlabMPI/, 2006.

of the Extended Kalman Filter when solving the differential system equations. The prediction includes both state and state covariance, and these differential equations are coupled and must therefore be solved simultaneously. To account for this coupling, the two prediction equations have been collected into one system with a combined input vector Z which stores both the states and the covariance matrix. The symmetry in the covariance matrix is exploited so only the upper part is transferred, i.e.

$$Z = \begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{x}}_{t|k} \\ U(\boldsymbol{P}_{t|k}) \end{pmatrix} \tag{17}$$

where $U()$ is a column vector containing the upper matrix. Conversion to the vector Z is then used in conjunction with `ode15s` and the output is converted back into a state vector and covariance matrix. The use of a single vector Z complies with the Matlab standard conventions for ODE solving algorithms, and the chosen algorithm `ode15s` may thus easily be substituted to suit the dynamic properties of a given model.

Validation of implementation

The implementation of PSM has been validated with CTSM and NONMEM. The comparison with CTSM has been used to verify correct implementation of the Kalman Filter and Extended Kalman Filter by comparison with CTSM's individual likelihood function. The comparison was based on a model using SDEs and showed identical outcomes from the two programs.

The comparison with NONMEM was done with a model based on ODEs in order to verify the population likelihood function. The comparison showed that PSM produces identical population parameter estimates and also identical estimates of the individual random effects parameters for four simulated data sets containing 2, 4, 10 and 20 subjects.

A final validation with NONMEM was done on the objective function value. The NONMEM objective function (l_{NM}) is advertised as $-2\log L$ but in fact it lacks a constant equal to the likelihood of the data. The PSM objective function (l_{PSM}) is $-\log L$ as seen in Eq. (16) and the relation thereby becomes $l_{NM} = 2 \cdot l_{PSM} - \log(2\pi) \cdot \sum n_i$. This relation between the two objective functions was found to hold for all the estimated models on the four simulated data sets, and this demonstrates that the formulations of the objective functions are equivalent.

APPLICATIONS

The general approach of including SDEs in the NLME model as implemented in PSM has a potential of improving model development and performance for a wide range of PK/PD modelling situations, as has been discussed previously. The applications to illustrate the functionality of

PSM in the present paper have been chosen to focus on a feature inherent to the new approach. The SDEs enable a simple way to estimate unknown inputs and time-varying parameters by modelling these as a random walk. The technique works for both linear and non-linear problems, and this will be illustrated in the following by two models to estimate the insulin secretion rate and liver extraction rate.

Data

The data originates from a double-blind, placebo-controlled, randomized crossover study with a duration of 24 hours starting at 8 a.m. in the morning. Thirteen patients (5 women and 8 men) with type II diabetes were examined. Their age given as mean $\pm$ 1 standard deviation was 56.4 $\pm$ 9.2 years, BMI was 31.2 $\pm$ 3.6 kg/m^2 and the duration of diabetes was 3.0 $\pm$ 2.6 years (range 5 months to 8 years) [15].

C-peptide and insulin measurements will be used for analysis in this paper, and only the placebo data is used. This is done to focus the presented analysis on two types of application of the NLME model which are only possible by extending it with SDEs, namely stochastic deconvolution of an unknown input and continuous tracking of the behavior of a parameter.

One of the patients was discarded since the measurement times were delayed compared to the rest. The data used thus consists of 24-hour C-peptide and insulin profiles for 12 individuals, see Figure 1.

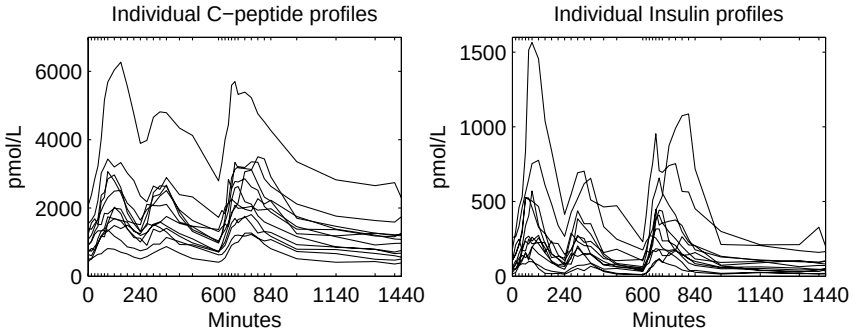

Figure 1: Individual profiles for C-peptide and Insulin.

The subjects were sampled 35 times during the 24 hours at varying time intervals, mainly concentrated after meal times. A total of 3 standard meals were given at 8 a.m., noon and 6 p.m., each to be finished within 20 minutes. These times correspond to 0, 240 and 600 minutes after the study was initiated, see Figure 2.

Deconvolution

The first example of application will illustrate how the model setup can be used for deconvolution of the insulin secretion rate (ISR) based on a standard two-compartment linear model for C-

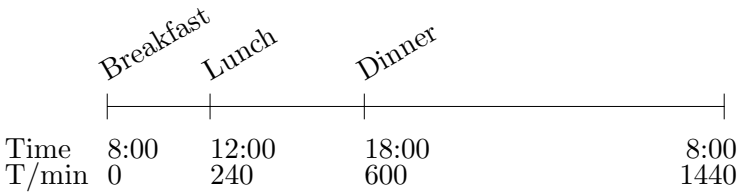

Figure 2: Meal times during 24H study period.

peptide measurements [16]. It is known that C-peptide and insulin are secreted in equi-molar amounts, and this fact is used to construct the model. The basic idea is to model the secretion rate into the central compartment as a pure random walk (Wiener process) and then estimate ISR as the realization of this random walk using the EKF to provide a smoothed estimate.

The modelling of the ISR as a random walk actually means that no model is given for the ISR, and therefore it is instead estimated entirely based on the data. For a linear system this technique resembles a deconvolution, but it will provide a more smooth estimate compared to an ordinary deconvolution. This is because the EKF separates system noise from measurement noise, where the system noise for this model is assumed to be the ISR of interest. The extent of smoothing is determined by the maximum likelihood estimated σ_{ISR}, the magnitude parameter for the random walk for ISR, which influences the Kalman gain on increments of the random walk. A larger magnitude leads to a more fluctuating random walk with larger increments and vice versa for a smaller magnitude. The resulting estimate of the random walk and thereby the ISR is thus optimal in a likelihood sense, since the EKF as mentioned earlier has been shown to yield the minimum variance state estimate for a linear system.

The deconvolution setup requires three states, namely a central compartment state C_1 modelling the measured C-peptide concentration, a peripheral compartment state C_2, and a state ISR for the random walk. This gives the state vector $\boldsymbol{x} = [\ C_1\ C_2\ ISR\ ]^T$. The C-peptide kinetic parameters k_1, k_2, k_e are set equal to the Van Cauter estimates found in [17].

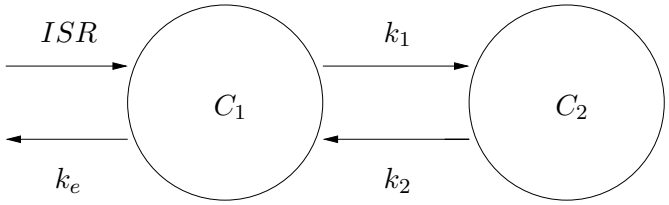

Figure 3: Two-compartment model used for estimation of ISR.

The C-peptide measurement error is assumed to be additive Gaussian white noise with variance Σ. The model states are constrained to steady state at $t = 0$ given an initial individually estimated concentration C_i in C_1, that is $\boldsymbol{x}_0 = \begin{bmatrix} C_i & \frac{k1}{k2}C_i & k_eC_i \end{bmatrix}^T$ and $C_i = C_1^0 \exp\eta, \quad \eta \in N(0, \Omega_{C_1})$. The state equation for the model is shown in Eq. (18)

$$d\boldsymbol{x} = \begin{bmatrix} -(k_1+k_e) & k_2 & 1 \\ k_1 & -k_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{x}\, dt + diag \begin{bmatrix} 0 \\ 0 \\ \sigma_{ISR} \end{bmatrix} d\boldsymbol{\omega} \tag{18}$$

and the measurement equation is simply $y = C_1 + \epsilon$, where $\epsilon \in N(0, \Sigma)$. The ML estimated population parameters are C_1^0, Σ, σ_{ISR} and $\Omega_{C_1^0}$ and based on these an optimal estimate of ISR can be found by using the Kalman smoothing algorithm.

Figure 4 shows the smoothed estimate of ISR for the first two individuals together with a ± 1 standard deviation band. The assumption of steady state in the beginning defines the initial level of ISR based on C_1^0 and this appears appropriate.

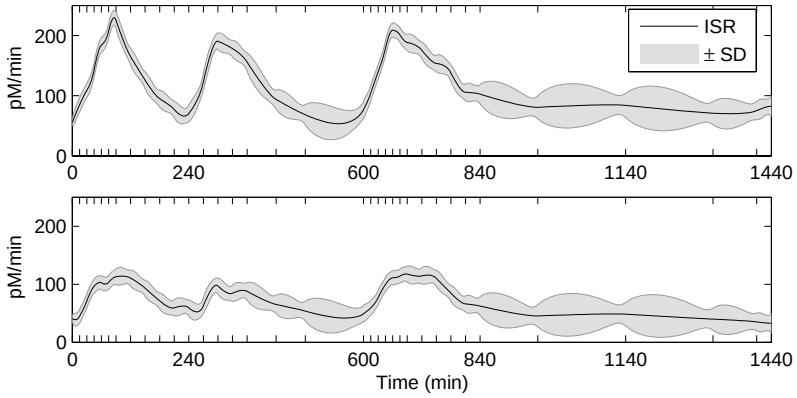

Figure 4: Smoothed estimate of ISR for individual 1 and 2.

State-Estimation

The second example of application goes to illustrate how the model setup may be used for state-estimation in non-linear systems. The method is also sometimes referred to as parameter tracking, when the state represents a parameter, which is suspected of having some time-varying behavior. Although non-linear state-estimation is fundamentally different from deconvolution, which only applies to linear systems, it can be performed with SDEs in basically the same way as the approach for deconvolution presented in the first example of application.

The aim is to estimate the dynamic liver extraction rate, which represents the fraction of insulin that is absorbed by the liver. This fraction is often modelled as a constant to simplify statistical models, although it is known to be time-varying. As previously done the insulin secretion rate is estimated based on the information in the C-peptide measurements and then used as input into a one-compartment insulin model. The state I models the measured insulin concentration in the compartment and the insulin elimination is set to $k_{e,I} = 0.355\ min^{-1}$. This value has

been reported for a similar study, also on type II diabetic patients [18]. By having a fixed elimination rate and ISR given from the C-peptide part of the data, the information in the insulin measurement can be used to estimate the liver extraction. The fraction which passes through the liver is modelled by a state F, and the input into the insulin compartment is thus $F \cdot ISR$ making the model non-linear in the states. The final layout of the model is shown in Figure 5. The layout is identical to the layout first proposed in [19], where it is shown that by assuming a constant liver extraction rate it is possible to estimate the kinetic parameters and a piecewise constant ISR.

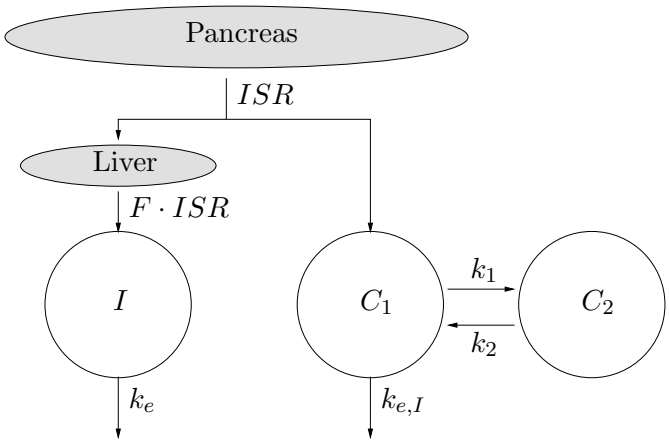

Figure 5: Dynamics of the combined model for estimation of insulin secretion rate and liver extraction rate.

In an initial model F was modelled directly as a random walk in the same way as ISR. The estimation of the model returned a very low estimate of the insulin measurement standard deviation at only 0.01pmol/L. This is an unrealistically small value and indicates a problem with separation of noise components, since virtually all the variation in the insulin measurements thereby is assumed to originate from the fluctuations of the liver extraction.

This problem can be solved by imposing further smoothing to the state-estimation by choosing to model *the derivative* of F as a random walk instead of directly F as before. This is achieved by introducing a new state named X as shown in Eq. (19) and (20)

$$\frac{dF}{dt} = X \tag{19}$$

$$dX = \sigma_X d\omega \tag{20}$$

where ω is a Wiener process. The change in the model for F causes the increments of the derivative of F to be penalized by the Wiener noise gain σ instead of the increments of F directly. The result is a less flexible model for F where fluctuations are further constrained, and

the modification is easily implemented using the flexibility made available by the stochastic state space approach. In total the model contains six states, namely $\boldsymbol{x} = [C_1\ C_2\ I\ ISR\ F\ X]^T$, which are all estimated simultaneously by the Extended Kalman Filter using the two-dimensional measurements with C-peptide and insulin. The system equations are shown in Eq. (21).

$$d\boldsymbol{x} = \begin{bmatrix} -(k_1 + k_e)C_1 + k_2 C_2 + ISR \\ k_1 C_1 - k_2 C_2 \\ -k_{e,I} I + F \cdot ISR \\ 0 \\ X \\ 0 \end{bmatrix} dt + diag \begin{pmatrix} 0 \\ 0 \\ 0 \\ \sigma_{ISR} \\ 0 \\ \sigma_X \end{pmatrix} d\boldsymbol{\omega} \tag{21}$$

The estimation of the population parameters in the new model with a constrained model for F results in a better separation of noise. The standard deviation for the insulin secretion rate is estimated at a satisfactory level of 19.8pmol/L.

As could be expected, the model finds an ISR which is almost identical to the one found using just a C-peptide deconvolution model, since the information in the added insulin measurements is used to estimate the liver extraction. The smoothed estimate of the fraction of insulin passing the liver F is shown in Figure 6 for individual 1 and 2. The plots illustrate that the proportion sent through the liver, F, is below one for the entire time interval as it naturally should be. This also holds for 8 out of the remaining 10 individuals. For the two last individuals F varies between 0.5 and 1.8. This is however not of great concern, because F and $k_{e,I}$ are correlated and it is thus probably just indicating that $k_{e,I}$ for this particular individual is set too high.

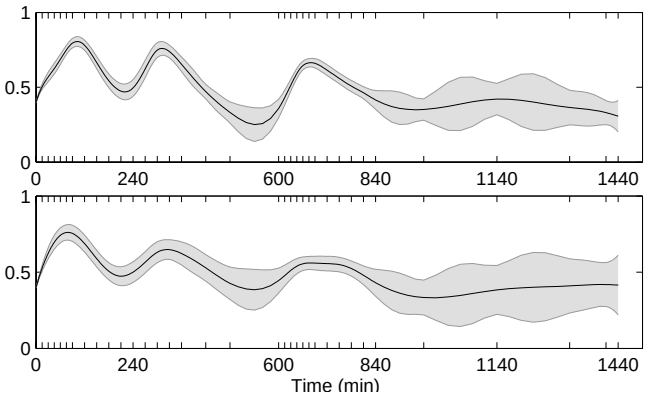

Figure 6: Smoothed fraction of insulin passing the liver ±1SD for individual 1 and 2.

DISCUSSION

By the presented software implementation PSM (Population Stochastic Modelling), we have shown that it is possible to develop a general purpose PK/PD population modelling tool that is able to handle the extra functionality made available by using SDEs in NLME models. The implementation opens up for the possibility to easily make further experiments with the model setup to allow for accumulation of more knowledge about the modelling approach.

It is important to emphasize that the software implementation is to be considered a prototype, which should only be used on research level. A necessary step to make it more widely usable is to move to another programming language. This implementation has been done in Matlab, which is ideal for numerical implementations, but it lacks in speed and parallel computing options. The standard within scientific programming today is Fortran, and this is also an obvious choice here due to its efficient handling of numerical computations and linear algebra calculations. Another advantage of Fortran is the accessibility of modules already available, such as algorithms for numerical optimization and ODE solvers.

The optimal platform for a future implementation is a shared memory system. Shared memory parallelism can be implemented easily in Fortran using the OpenMP[3] application program interface. OpenMP supports multi-platform shared-memory parallel programming in Fortran on all architectures, including Unix and Windows platforms. OpenMP is a scalable model that gives a flexible interface for developing a parallel application for platforms ranging from the desktop to the supercomputer and it supports parallelism through meta tags that will make portability to single CPU, multi-core CPU, and shared-memory multiprocessor (SMP) units simple. Some compilers are also able to create parallel calculations by automatically analyzing the code, but the largest improvements are achieved using manual parallelization.

The present paper has illustrated how parallelization introduced at the individual minimizations of the population likelihood function has a strong potential of reducing the estimation time for a future final software program when dealing with data containing a large number of individuals. It can also be argued to introduce parallelism at an even higher level in the gradient calculation of the population likelihood function. This would generally be advantageous for models where the number of population parameters exceeds the number of individuals in data.

The first example of application in this paper demonstrated how the NLME model can be used for deconvolution of ISR by introducing SDEs. Although the estimation of ISR using stochastic differential equations is loosely denoted deconvolution, it is in fact not strictly speaking deconvolution but instead a probabilistic description of an unknown input, which is modelled as the realization of a stochastic process. Pure deterministic deconvolution using ODEs for the model shown in Figure 3 will estimate ISR at each measurement to be equal to the rate giving the 'missing' amount in the central C-peptide compartment C_1. With the SDE approach the measurement noise on C-peptide is taken into account by the Kalman filter, which yields a

[3]Further details may be found at www.openmp.org.

minimal variance estimate of the states resulting in a more smooth estimate of ISR where the effect of noise is reduced.

Deconvolution based on noisy data is generally an ill-posed problem, meaning that even small perturbations in data lead to significant changes in the estimated solution [20]. The problem has been addressed by existing software by applying various kinds of regularization techniques to constrain the solution. An example is WinNonlin [21], which is a standard PK/PD software solution that can also be used for deconvolution. The program addresses the problem of deconvolution by introducing a smoothness factor and as a consequence it is simply left up to personal choice and preference of the user to specify the level of smoothing. An improved solution can be found using WinStoDec presented by [22], which is based on stochastic deconvolution and can be used for linear time-invariant systems [23]. It has been shown by [24] that the stochastic deconvolution approach is equivalent to the SDE approach presented here, which is furthermore by nature also able to handle non-linear time-varying systems, as has been demonstrated with the state-estimation approach in the second example of application presented here.

In conclusion, a fully functional prototype tool named PSM (Population Stochastic Modelling) for estimation of non-linear mixed-effects models based on SDEs has been implemented in Matlab and validated. The use of parallelization in the implementation has demonstrated a strong potential of reducing computation times in future implementations in a faster programming language. Finally two examples of application concerning insulin modelling demonstrated the possibility for deconvolution and non-linear parameter tracking facilitated by the extension of the NLME model to use SDEs.

ACKNOWLEDGMENTS

The Authors wish to thank Ole Schmitz MD and his co-workers at The Institute of Pharmacology, University of Aarhus, for conducting the present study enabling us to perform the presented analysis.

—— o ——

A Approximation of population likelihood function

The population likelihood function for the NLME model with SDE's is defined in Eq. (13) as

$$L(\boldsymbol{\theta}, \boldsymbol{\Sigma}, \boldsymbol{\sigma}_\omega, \boldsymbol{\Omega}) \;=\; \prod_{i=1}^{N} \int \exp(l_i) d\boldsymbol{\eta}_i \tag{22}$$

where l_i is the individual *a posteriori* log-likelihood function. In most cases the integral cannot

be evaluated analytically. For a general evaluation the individual *a posteriori* likelihood function can be approximated by a second order Taylor series expansion of l_i around the value $\hat{\boldsymbol{\eta}}_i$ of the individual random effects parameter which maximizes l_i. It follows that

$$\begin{aligned} l_i &\approx l_i + \boldsymbol{\nabla}^T l_i(\boldsymbol{\eta}_i - \hat{\boldsymbol{\eta}}_i) + \frac{1}{2}(\boldsymbol{\eta}_i - \hat{\boldsymbol{\eta}}_i)^T \boldsymbol{\Delta} l_i (\boldsymbol{\eta}_i - \hat{\boldsymbol{\eta}}_i) && (23) \\ &\approx l_i + \frac{1}{2}(\boldsymbol{\eta}_i - \hat{\boldsymbol{\eta}}_i)^T \boldsymbol{\Delta} l_i (\boldsymbol{\eta}_i - \hat{\boldsymbol{\eta}}_i) && (24) \end{aligned}$$

since $\boldsymbol{\nabla} l_i = 0$ at $\hat{\boldsymbol{\eta}}_i$. Based on the approximation in Eq. (24) the integral in Eq. (22) can now be evaluated by moving constants such that the integral is over a multi-variate Gaussian density with mean $\hat{\boldsymbol{\eta}}_i$ and variance $(-\boldsymbol{\Delta} l_i)^{-1}$. This integral is equal to one and the result is

$$\begin{aligned} \int L_i d\boldsymbol{\eta}_i &\approx \int L_i \cdot \exp\left(-\frac{1}{2}(\boldsymbol{\eta}_i - \hat{\boldsymbol{\eta}}_i)^T (-\boldsymbol{\Delta} l_i)(\boldsymbol{\eta}_i - \hat{\boldsymbol{\eta}}_i)\right) d\boldsymbol{\eta}_i && (25) \\ &\approx L_i \left|\frac{2\pi}{-\boldsymbol{\Delta} l_i}\right|^{\frac{1}{2}} \int \left|\frac{2\pi}{-\boldsymbol{\Delta} l_i}\right|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{\eta}_i - \hat{\boldsymbol{\eta}}_i)^T (-\boldsymbol{\Delta} l_i)(\boldsymbol{\eta}_i - \hat{\boldsymbol{\eta}}_i)\right) d\boldsymbol{\eta}_i && (26) \\ &\approx L_i \left|\frac{2\pi}{-\boldsymbol{\Delta} l_i}\right|^{\frac{1}{2}} \cdot 1 && (27) \\ &\approx L_i \left|\frac{-\boldsymbol{\Delta} l_i}{2\pi}\right|^{-\frac{1}{2}} && (28) \end{aligned}$$

where $L_i = \exp(l_i)$. The step in Eq. (28) is done to avoid a matrix inversion of the Hessian. By combining Eq. (22) and Eq. (28) the population log-likelihood function can now be approximated by

$$L(\boldsymbol{\theta}, \boldsymbol{\Sigma}, \boldsymbol{\sigma}_\omega, \boldsymbol{\Omega}) \approx \prod_{i=1}^{N} \left|\frac{-\boldsymbol{\Delta} l_i}{2\pi}\right|^{-\frac{1}{2}} \exp(l_i)\Big|_{\hat{\boldsymbol{\eta}}_i} . \quad (29)$$

References

[1] L. Aarons. Editorial - pharmacokinetic and pharmacodynamic modelling in drug development. *Stat. Methods Med. Res.*, 8(3):181–182, 1999.

[2] M. O. Karlsson, E. N. Jonsson, C. G. Wiltse, and J. R. Wade. Assumption testing in population pharmacokinetic models: Illustrated with an analysis of moxonidine data from congestive heart failure patients. *J. Pharmacokinet. Biopharm.*, 26(2):207–246, 1998.

[3] M. O. Karlsson, S. L. Beal, and L. B. Sheiner. Three new residual error models for population pk/pd analyses. *J. Pharmacokinet. Pharmacodyn.*, 23(6):651–672, 1995.

[4] N. R. Kristensen, H. Madsen, and S. H. Ingwersen. Using stochastic differential equations for pk/pd model development. *J. Pharmacokinet. Pharmacodyn.*, 32:109–41, Feb 2005.

[5] N. R. Kristensen, H. Madsen, and S. B. Jørgensen. Parameter estimation in stochastic grey-box models. *Automatica*, 40:225–237, 2004.

[6] Rune V. Overgaard, Niclas Jonsson, Christoffer W. Tornøe, and Henrik Madsen. Non-linear mixed-effects models with stochastic differential equations: implementation of an estimation algorithm. *J. of Pharmacokinet. Pharmacodyn.*, 32(1):85–107, 2005.

[7] C. W. Tornøe, R. V. Overgaard, H. Agersø, H. A. Nielsen, H. Madsen, and E. N. Johnson. Stochastic differential equations in nonmem®: Implementation, application, and comparison with ordinary differential equations. *Pharm. Res.*, 22(8):1247–1258, 2005.

[8] S. L. Beal and L. B. Sheiner. *NONMEM® Users Guide*. NONMEM Project Group, University of California, San Francisco, 2004.

[9] R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *Trans. ASME Ser. D. J. Basic Engrg.*, 83:95–108, 1961.

[10] J. Holst, E. Lindström, H. Madsen, and H. A. Nielsen. Model validation in non-linear continuous-discrete grey-box models. *Proceedings of the 13th IFAC Symposium on System Identification*, Rotterdam, The Netherlands, 2000.

[11] N. R. Kristensen and H. Madsen. Continuous time stochastic modelling - ctsm 2.3 mathematics guide. Technical report, Technical University of Denmark, December 2003.

[12] Arthur Gelb, Jr. Joseph F. Kasper, Jr. Raymond A. Nash, Charles F. Price, and Jr. Arthur A. Sutherland. *Applied Optimal Estimation*. The MIT Press, seventh edition, 1982.

[13] H. B. Nielsen. Ucminf - an algorithm for unconstrained, nonlinear optimization. *Technical report*, IMM, DTU, 2000.

[14] Jr. J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice Hall Series in Computational Mathematics. Prentice Hall Inc., Englewood Cliffs, NJ, 1983.

[15] Kristine B. Degn, Claus B. Juhl, Jeppe Sturis, Grethe Jakobsen, Birgitte Brock, Visvanathan Chandramouli, Joergen Rungby, Bernard R. Landau, and Ole Schmitz. One Week's Treatment With the Long-Acting Glucagon-Like Peptide 1 Derivative Liraglutide (NN2211) Markedly Improves 24-h Glycemia and alpha- and beta-Cell Function and Reduces Endogenous Glucose Release in Patients with Type 2 Diabetes. *Diabetes*, 53(5):1187–1194, 2004.

[16] J. Gabrielsson and D. Weiner. *Pharmacokinetic and Pharmacodynamic Data Analysis: Concepts and Applications*. Kristianstads Boktryckeri, second edition, 1997.

[17] E. Van Cauter, F. Mestrez, J. Sturis, and K. S. Polonsky. Estimation of insulin secretion rates from c-peptide levels. comparison of individual and standard kinetic parameters for c-peptide clearance. *Diabetes*, 41(3):368–77, Mar 1992.

[18] L. L. Kjems, A. Vølund, and S. Madsbad. Quantification of beta-cell function during ivgtt in type ii and non-diabetic subjects: assessment of insulin secretion by mathematical methods. *Diabetologia*, 44:1339–1348, 2001.

[19] R. M. Watanabe, G. M. Steil, and R. N. Bergman. Critical evaluation of the combined model approach for estimation of prehepatic insulin secretion. *American Journal of Physiology*, 274(1):172–183, 1998.

[20] J. Hadamard. *Lectures on the Cauchy Problem in Linear Partial Differential Equations*. Yale University Press, New Haven, 1923.

[21] Pharsight. *WinNonlin User Manual version 3.1*. Pharsight, deconvolution edition, 2004.

[22] G. Sparacino, G. Pillonetto, M. Capello, G. De Nicolao, and C. Cobelli. Winstodec: A stochastic deconvolution interactive program for physiological and pharmacokinetic systems. *Comput. Methods. Programs. Biomed.*, 67:67–77, 2002.

[23] G. de Nicolao, G. Sparacino, and C. Cobelli. Nonparametric input estimation in physiological systems: Problems, methods, and case studies. *Automatica*, 33(5):851–870, 1997.

[24] N. R. Kristensen, H. Madsen, and S. H. Ingwersen. A deconvolution method for linear and nonlinear systems based on stochastic differential equations. poster presented at: Population Approach Group in Europe (PAGE), 13th meeting, June 2004.

CPUs	Time (sec)	Reduced to (%)	CPU-time per individual (sec)	Overhead per CPU (%)
1 *serial*	241.8	100.0	12.1	-
1	242.4	100.2	12.1	0.2
2	128.3	53.0	12.8	5.8
3	101.7	42.0	15.3	20.7
4	72.0	29.7	14.4	16.0
5	66.0	27.3	16.5	26.7
10	50.0	20.6	25.0	51.6

Table I: Computation times using parallel computing.