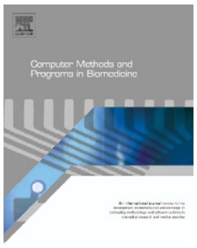
journal homepage: www.intl.elsevierhealth.com/journals/cmpb



Population stochastic modelling (PSM)—An R package for mixed-effects models based on stochastic differential equations

Søren Klim[a,c,*], *Stig Bousgaard Mortensen*[b,c], *Niels Rode Kristensen*[a], *Rune Viig Overgaard*[a], *Henrik Madsen*[c]

[a] *Novo Nordisk A/S, Novo Alle, 2880 Bagsværd, Denmark*
[b] *H. Lundbeck A/S, Ottiliavej 9, 2500 Valby, Denmark*
[c] *Department of Informatics and Mathematical Modelling, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark*

A R T I C L E   I N F O

Article history:
Received 26 August 2008
Received in revised form
30 January 2009
Accepted 1 February 2009

Keywords:
Stochastic differential equations (SDEs)
State-space models
Mixed-effect
Pharmacokinetic
Pharmacodynamic

A B S T R A C T

The extension from ordinary to stochastic differential equations (SDEs) in pharmacokinetic and pharmacodynamic (PK/PD) modelling is an emerging field and has been motivated in a number of articles [N.R. Kristensen, H. Madsen, S.H. Ingwersen, Using stochastic differential equations for PK/PD model development, J. Pharmacokinet. Pharmacodyn. 32 (February(1)) (2005) 109–141; C.W. Tornøe, R.V. Overgaard, H. Agersø, H.A. Nielsen, H. Madsen, E.N. Jonsson, Stochastic differential equations in NONMEM: implementation, application, and comparison with ordinary differential equations, Pharm. Res. 22 (August(8)) (2005) 1247–1258; R.V. Overgaard, N. Jonsson, C.W. Tornøe, H. Madsen, Non-linear mixed-effects models with stochastic differential equations: implementation of an estimation algorithm, J. Pharmacokinet. Pharmacodyn. 32 (February(1)) (2005) 85–107; U. Picchini, S. Ditlevsen, A. De Gaetano, Maximum likelihood estimation of a time-inhomogeneous stochastic differential model of glucose dynamics, Math. Med. Biol. 25 (June(2)) (2008) 141–155].

PK/PD models are traditionally based ordinary differential equations (ODEs) with an observation link that incorporates noise. This state-space formulation only allows for observation noise and not for system noise. Extending to SDEs allows for a Wiener noise component in the system equations. This additional noise component enables handling of autocorrelated residuals originating from natural variation or systematic model error. Autocorrelated residuals are often partly ignored in PK/PD modelling although violating the hypothesis for many standard statistical tests.

This article presents a package for the statistical program R that is able to handle SDEs in a mixed-effects setting. The estimation method implemented is the FOCE[1] approximation to the population likelihood which is generated from the individual likelihoods that are approximated using the Extended Kalman Filter's one-step predictions.

* *Corresponding author at*: DTU Informatics, Technical University of Denmark, Richard Petersens Plads, Building 321, 2800 Kgs. Lyngby, Denmark. Tel.: +45 4525 3351.
E-mail addresses: SKli@novonordisk.com, skl@imm.dtu.dk (S. Klim).

[1] FOCE—First-Order Conditional Estimation.

0169-2607/$ – see front matter © 2009 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.cmpb.2009.02.001

1. Introduction

The use of mixed-effects models based on ordinary differential equations (ODEs) is the standard for pharmacokinetic and pharmacodynamic (PK/PD) modelling. The use of stochastic differential equations (SDEs) is an emerging field and has been introduced and motivated in the papers [1–4]. This paper presents an accessible software package for handling dynamic models based on SDEs in a mixed-effects setting. The program is a package for the statistical program R and thereby easy to install through R's interface and available for a wide range of operating systems.

The package implements the (Extended) Kalman Filter for evaluating the likelihood function in models based on SDEs. The parameter estimation procedure in the package is maximum likelihood based with fixed effects estimation based on the FOCE approximation.

2. Computational methods and theory

The most widely used program to analyze state-space models in PK/PD-modelling is NONMEM [5], which is focused on mixed-effects models based on ordinary differential equations. The use of SDEs in non-linear mixed-effects models is possible in NONMEM as described in [2]. The trick is an implementation of the Extended Kalman Filter in the NONMEM control file with corresponding adjustments to the data file. This is a non-trivial task even for rather simple models and must be repeated for every change in model or data. Single subject data can be modelled with stochastic differential equations in the program *CTSM* [6]. *CTSM* is a stand alone program that works across different platforms.

The matlab framework described in [7] handles SDEs in a mixed-effect setting. The experiences collected in the development of the Matlab framework have now been used to create an extended and more flexible R-package PSM.

The mathematical basis for the PSM package is also described in the articles [3,7,8]. It should be noted that there are notation differences between the articles.

For simplicity this article focuses on the class of linear models but it must be emphasised that the package also handles non-linear models.

2.1. Single subject

The modelling of observations for a single subject is based on a continuous-discrete state-space model. The states represent the internal hidden states of the system. The states reside in a continuous time domain and their dynamics are described by stochastic differential equations. The observations are sampled at discrete time points and hence described by a discrete time relation.

The class of linear models handled by PSM are time-invariant models meaning that system matrices do not change over time. More specifically the linear state-space model can be stated as

$$d\boldsymbol{x}_t = (\boldsymbol{A}(\boldsymbol{\phi}_i)\boldsymbol{x}_t + \boldsymbol{B}(\boldsymbol{\phi}_i)\boldsymbol{u}_t)\,dt + \boldsymbol{\sigma}(\boldsymbol{\phi}_i)d\boldsymbol{\omega}_t \qquad (1)$$

$$\boldsymbol{y}_{ik} = \boldsymbol{C}(\boldsymbol{\phi}_i)\boldsymbol{x}_{ik} + \boldsymbol{D}(\boldsymbol{\phi}_i)\boldsymbol{u}_{ik} + \boldsymbol{e}_{ik} \qquad (2)$$

where $\boldsymbol{x}_t \in \mathbb{R}^{dimX}$ is the vector of states at time t. The dimension of $\boldsymbol{x}$ is denoted as *dimX* for simplicity. $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ are time-invariant matrices defined as functions of $\boldsymbol{\phi}_i$ with properties $\boldsymbol{A}(\cdot) \in \mathbb{R}^{dimX\times dimX}$, $\boldsymbol{B}(\cdot) \in \mathbb{R}^{dimX\times dimU}$, $\boldsymbol{C}(\cdot) \in \mathbb{R}^{dimY\times dimX}$ and $\boldsymbol{D}(\cdot) \in \mathbb{R}^{dimY\times dimU}$. $\boldsymbol{\phi}_i$ is the parameter vector for the ith individual (see Eq. (9) for further details). The exogenous input $\boldsymbol{u} \in \mathbb{R}^{dimU}$ can be used to include other measured variables which influence the time evolution of the states in the model. The input $\boldsymbol{u}$ is assumed to be constant between observation points which is often refered to as zero-order hold or piece wise constant. The component $\boldsymbol{\sigma}(\boldsymbol{\phi}_i)d\boldsymbol{\omega}_t$ is the system noise consisting of a scaling diffusion term $\boldsymbol{\sigma}(\cdot) \in \mathbb{R}^{dimX\times dimX}$ and $\boldsymbol{\omega}_t$ which is a *dimX*-dimensional Wiener process. The subscript i denotes the ith subject and the subscript k is a short hand notation for t_k. $\boldsymbol{y}_{ik}$ is the observation at time t_k for the ith subject. $\boldsymbol{e}_{ik}$ is the residual for individual i at time t_k and is assumed to normal distributed $N(\boldsymbol{0}, \boldsymbol{S}(\phi_i))$ with $\boldsymbol{S}(\cdot) \in \mathbb{R}^{dimY\times dimY}$ being the covariance matrix for the errors.

2.2. Kalman Filter

The deterministic behaviour of ordinary differential equations can be handled with standard differential equation solvers. The additional component in the SDE systems requires a more advanced solution method. As mentioned in the introduction this package uses the Kalman Filter as solution method for systems of SDEs.

The Kalman Filter is only briefly explained in this article but the mathematics behind the Kalman Filter is well described in the Mathematics guide to CTSM [6] and in the original reference [9]. Several links and additional material can be found on the homepage [10].

The assumptions on system noise being driven by a Wiener process and normally distributed errors will in a linear system under some regularity conditions [6] result in the conditional densities for the observations being fully described by their first- and second-order moments. The Kalman Filter can be used to determine the optimal internal states in the system conditioning on prior observations. The Kalman Filter updates the internal state vector after each observation and during this process the Kalman Filter needs to weigh the probability of the residual being due to system noise or measurement noise. For this purpose the one-step prediction $\hat{\boldsymbol{y}}_{k|k-1}$ and associated covariance $\boldsymbol{R}_{k|k-1}$ are defined below:

$$\hat{\boldsymbol{y}}_{k|k-1} = E[\boldsymbol{y}_k|\mathcal{Y}_{k-1}, \boldsymbol{\phi}_i] \qquad (3)$$

$$\boldsymbol{R}_{k|k-1} = V[\boldsymbol{y}_k|\mathcal{Y}_{k-1}, \boldsymbol{\phi}_i] \qquad (4)$$

where $\mathcal{Y}_{k-1}$ denotes all measurements up to time t_{k-1}.

The description of conditional densities based on first- and second-order moments is only exact for linear models. For nonlinear models the Extended Kalman Filter can be used which is based on continous linerazations of the model however the forming of the conditional densitites will only be approximate.

The structure of the Kalman Filter is thus an iterative process with a prediction/updating scheme. The one-step

Table 1 – The Kalman Filter written in algorithmic form. Copied from [11].

```
Algorithm: The Kalman Filter
Given parameters and initial prediction
$\phi_i$, $\hat{x}_{1|0}$ and $P_{1|0}$
For $k = 1$ to $n_i$ do
    Output Prediction:
    $\hat{y}_{k|k-1} = C\hat{x}_{k|k-1} + Du_k$
    $R_{k|k-1} = CP_{k|k-1}C^T + S$
    State Update:
    $K_k = P_{k|k-1}C^T R_{k|k-1}^{-1}$
    $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - \hat{y}_{k|k-1})$
    $P_{k|k} = P_{k|k-1} - K_k R_{k|k-1} K_k^T$
    State Prediction:
    $d\hat{x}_{t|k}/dt = A\hat{x}_{t|k} + Bu_t$
    $dP_{t|k}/dt = A_t P_{t|k} + P_{t|k}A^T + \sigma\sigma^T$
end for
```

prediction will be based on the deterministic part of the model as the Wiener component has mean zero. The measured observation should be considered as a result of the current states including accumulated system noise since last observation and current observation noise. An updating of the internal states based on the residual is weigthed according to system noise and measurement noise. The iterative structure restarts with a prediction based on the updated states.

The initial conditions for the Kalman Filter are the initial states $\hat{x}_{1|0}$ and initial state covariance matrix $\hat{P}_{1|0}$. Optimally the initial conditions and uncertainties are known a priori but generally none or only the initial state is known. The initial states can be either specified directly or estimated simultaneously with the remaining parameters. As the uncertainty is rarely known the package assumes that the initial state uncertainty is equal to the integral of the Wiener noise over a time interval equal to the interval between the first two observations.

$$P_0 = P_s \int_{t_0}^{t_1} e^{As}\sigma\sigma^T(e^{As})^T ds \tag{5}$$

where P_s is a scaling factor. One solution method for the integral (5) is shown in [6].

In Table 1 the iterative structure of the Kalman Filter is shown in algorithmic form.

The Kalman Filter setup requires a specific model structure as shown in Eqs. (1) and (2). Two requirements that should be noted are the additive noise in the observations and the state independent Wiener component. The Kalman Filter cannot handle a multiplicative or full error model in Eq. (2). Using a log transformation of the observations a log normal error model can however be dealt with. The limitation with state independent Wiener component can only be surpassed by transformation of the system equations or by introducing more sophisticated methods such as higher order filters or Markov Chain Monte Carlo methods.

The residual used in the likelihood function is defined as

$$\epsilon_k = y_k - \hat{y}_{k|k-1} \tag{6}$$

The likelihood of the parameters ϕ_i based on data can be calculated using the assumption of normality combined with the conditional covariance which is calculated in the Kalman Filter.

$$L(\phi_i; \mathcal{Y}) = \left(\prod_{k=1}^{n_i} \frac{\exp(-(1/2)\epsilon_k^T R_{k|k-1}^{-1}\epsilon_k)}{\sqrt{\det(R_{k|k-1})}(\sqrt{2\pi})^{dimY}} \right) p(y_0|\phi_i) \tag{7}$$

The negative log likelihood can be derived from Eq. (7) by conditioning on the initial condition y_0. The negative log-likelihood is the objective function used in the parameter estimation in the Kalman Filter.

$$-\ln(L(\phi_i; \mathcal{Y}|y_0)) = \frac{1}{2}\sum_{k=1}^{n_i}(\ln(\det(R_{k|k-1})) + \epsilon_k^T R_{k|k-1}^{-1}\epsilon_k) + \frac{1}{2}\left(\sum_{k=1}^{n_i} dimY\right)\ln(2\pi) \tag{8}$$

2.3. *Mixed-effects*

The use of non-linear mixed effects models in PK/PD modelling has long been the standard and has been supported by the functionality in NONMEM. The mixed-effects approach to model variation in pharmacokinetics was first introduced by Sheiner in [12]. Mixed-effects modelling is a hierarchical division of the variation, where the fixed effects describe the population mean and the random effects describe the inter-individual variation. This is often described in two stages. The first stage model describes the intra-individual variability and the second stage describes the inter-individual variation.

The first stage model is described in Eqs. (1) and (2) which are based on the individual parameters. The inter-individual variation in parameters is covered in the creation of the individual parameters in the function $h(\cdot)$ described below:

$$\phi_i = h(\theta, \eta_i, Z_i) \tag{9}$$

where θ are the fixed effects—also sometimes referred to as the population parameters. Z_i denotes subject covariates and $\eta_i \in N(0, \Omega)$ are the random effects. The individual parameters can be modelled as either normally or log-normally distributed by combining the population parameters and the random effects in either an additive ($\phi_i = \theta + \eta_i$) or an exponential transform ($\phi_i = \theta \exp(\eta_i)$).

The likelihood function for the fixed effects can be stated as below:

$$L(\theta) = \prod_{i=1}^{N} \int p_1(\mathcal{Y}_i|\theta, \eta_i) p_2(\eta_i|\Omega)\, d\eta_i = \prod_{i=1}^{N} \int \exp(l_i)\, d\eta_i \tag{10}$$

where N is the number of subjects. $p_1(\mathcal{Y}_i|\theta, \eta_i)$ is the probability for the first stage model which is proportional to Eq. (7). $p_2(\eta_i|\Omega)$ is the probability related to the second stage model that relates the random effects to the inter-individual variation. l_i is the *a posteriori* log-likelihood function for the ith individual. $\mathcal{Y}_i$ is the complete sequence of observations for individual i. The population likelihood function in Eq. (10) rarely has a closed form solution and therefore l_i is approximated by a second-order

Taylor expansion, where the expansion is made around the value $\hat{\eta}_i$ that maximizes l_i. At this optimum the first derivative $\nabla l_i|_{\hat{\eta}_i} = 0$ and the population likelihood function will under some assumptions reduce to

$$L(\theta) \approx \prod_{i=1}^{N} \left| \frac{-\Delta l_i}{2\pi} \right|^{-(1/2)} \exp(l_i)|_{\hat{\eta}_i} \tag{11}$$

The approximation of the 2nd derivative Δl_i is done using the First-Order Conditional Estimation (FOCE) method, which is defined as

$$\Delta l_i^* \approx -\sum_{j=1}^{n_i} (\nabla \epsilon_{ij}^T R_{i(j|j-1)}^{-1} \nabla \epsilon_{ij}) - \Omega^{-1}, \quad \text{where} \quad \nabla \epsilon_{ij} = \frac{\partial}{\partial \eta_i} \epsilon_{ij}|_{\eta_i^*} \tag{12}$$

When the random effects have a non-linear influence on the likelihood through the first stage model, the combined model is called a non-linear mixed effects model.

The conditional residual covariance $R_{i(j|j-1)}^{-1}$ is calculated in the (Extended) Kalman Filter and the gradient in the residual with relation to the random effects $\nabla \epsilon_{ij}$ is typically found by numerical methods.

2.4. *Maximum likelihood estimation*

The population likelihood function in (11) is used in maximum likelihood estimation of the population parameters. This optimization becomes a nested optimization as the FOCE approximation is based on the optimal random effects (η_i^*). Each calculation of the population likelihood thus requires N optimizations of the random effects. This nested optimization makes the computational effort substantial and highly dependent on the number of subjects, number of observations and the number of fixed and random effects. The optimization in the PSM package is performed with the default optimizer (optim) which is a quasi-Newton based optimizer.

The optimization can be constrained using boundaries on the population parameters using a logit-transform. It is assumed that the optimizer works on a unconstrained parameter space so the logit transform maps the bounded parameters into an unbounded space. In order to avoid flat gradients in population parameters in the outer parts of the logit transform a penalty function has been added. The penalty function is defined as below:

$$P(\lambda, \theta, \theta^{\min}, \theta^{\max}) = \lambda \left(\sum_{j=1}^{p} \frac{|\theta_j^{\min}|}{\theta_j - \theta_j^{\min}} + \sum_{j=1}^{p} \frac{|\theta_j^{\max}|}{\theta_j^{\max} - \theta_j} \right) \tag{13}$$

where p is the number of parameters and $\theta_j^{\min}$ and $\theta_j^{\max}$ denotes the lower and upper limit for the jth parameter.

The computational effort in the parameter estimation can as already mentioned be substantial and it is advised to find good initial estimates for the parameters in advance.

3. Program description

The framework for handling mixed-effects models based on SDEs has previously been implemented in Matlab [7]. The R package PSM presented here is a ported and extended version. The switch in platform was motivated by R being an open source program and its availability on different platforms. The PSM package was extended from the Matlab framework by extending the flexibility, improving performance by low level implementations and enabling capabilities for bolus doses. The dosing capability is crucial for modelling in drug development.

The package is able to handle multivariate observations, which are useful when performing simultaneous fits of multivariate data such as insulin and glucose. Another feature is that it is possible to have input into the model and include subject covariates. Finally, the package handles missing observations.

The package is mainly implemented in the R-language which is closely related to the S-language. Core components of the code have been implemented in FORTRAN for faster computation.

The current PSM version is 0.8-3. The package has dependencies for three other R packages. MASS is used in the simulation part to sample from the multivariate normal distribution. MASS is an integrated part of the R installation. The package odesolve is used in the non-linear models to solve systems of differential equations. The package numDeriv is used to calculate the Hessian which is used in the calculations of the confidence intervals for the estimated parameters.

The PSM package is available as a standard R package. Installation can be performed using the R interface or by executing the command.

```
> install.packages("PSM")
```

The package comes with complete documentation and "get-started" guide. The documentation can be found by executing the command help("PSM"). A more thorough guide to the package and its usage can be obtained with the command

```
> vignette("PSM")
```

The package is divided into three parts according to functionality. The three parts are

- Simulation
- Estimation
- Smoothing

Simulation is the creation of observations based on a given model and model inputs. The Estimation part performs a maximum likelihood estimation of the population parameters based on the one-step predictions in the Kalman Filter. The smoothing functionality creates the optimal state estimates based on the entire data series and a set of parameters.

All three functions operate on a model object and a data object. The following sections introduce the model and data objects before going into detail with the three functions.

Table 2 – Model specification.

Functions	Output
`Matrices`	List with system matrices see Eq. (1)
`X0`	Matrix with initial state condition(s)
`SIG`	Matrix with diffusion scaling term $\boldsymbol{\sigma}$
`S`	Matrix with residual covariance
`h`	Vector with individualized parameters $\boldsymbol{\phi}_i$
`ModelPar`	List with fixed effects $\boldsymbol{\theta}$ and inter individual variation $\boldsymbol{\Omega}$

Table 3 – Data specification.

`PSM`notation	Description
`Time`	Vector with dose and observation times
`Y`	Matrix with observations. Multivariate observations in columns
`U`	Matrix with input
`Dose`	List with `Time`, `State`and `Amount`
`covar`	Subject covariates

3.1. *Model specification*

The model specification is divided into components corresponding to the mathematical parts of the state-space model. For the linear case the state-space model can be stated in matrix form as seen in Eq. (1), but variance components and initial conditions are also needed. Table 2 shows the components in the model. The components are collected in a list to have a single object that contains the model. The individual components are all functions that return either a matrix or a list, if multiple outputs are needed.

Fig. 1 shows the model specification in a diagram with mathematical references displayed. The sequence for these components needs some elaboration. The `ModelPar` function is used to split the vector of parameters to be optimized $\boldsymbol{\Theta}$ into the fixed effects vector $\boldsymbol{\theta}$ and the random effects covariance matrix $\boldsymbol{\Omega}$. The individual parameters are created with the *h* function that uses the fixed effects, the random effects and the subject covariates to create the $\boldsymbol{\phi}_i$ vector. The remaining components in the dynamical system can be evaluated using $\boldsymbol{\phi}_i$ and the system input $\boldsymbol{u}$.

3.2. *Data specification*

The data specification in the simulation procedure is different from the specification in the parameter estimation and smoothing. In the simulation part the observations are simulated based on the model. Time points for the observations and potentially system input, doses or subject covariates still need to be provided. The time points, system input, doses or covariates are specified per subject in a list. Names in the list need to be according to the `PSM` specifications as the referencing in the package is done with names. The naming of the components can be seen in Table 3. The lists for all subjects are finally collected in a list which makes the overall data object a list of lists.

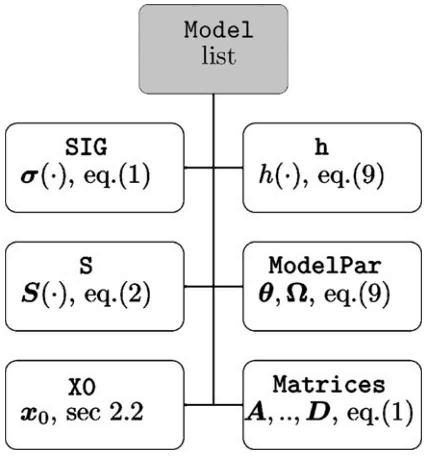

Model list
SIG
$\boldsymbol{\sigma}(\cdot)$, eq.(1)
h
$h(\cdot)$, eq.(9)
S
$\boldsymbol{S}(\cdot)$, eq.(2)
ModelPar
$\boldsymbol{\theta}, \boldsymbol{\Omega}$, eq.(9)
X0
$\boldsymbol{x}_0$, sec 2.2
Matrices
$\boldsymbol{A}, .., \boldsymbol{D}$, eq.(1)

Fig. 1 – Model components.

The observations are specified in the element `Y` which is a matrix with dimensions [*dim*Y, *dimT*]. *dimT* is the length of the `Time` vector. As can be seen from the dimensions of $\boldsymbol{Y}$, multivariate observations are specified in columns. Missing observations are indicated using the `NA` identifier. `Y` can be omitted if the data object is used in a simulation.

The `Dose` component contains the bolus doses information. The elements used to describe a bolus dose are the time of dosing, the amount dosed and the state in the model into which the bolus is given. The `Time` vector in `Dose` contains the times to which doses are given. It is important that the time points in the `Dose` component is a subset of the overall `Time` vector otherwise the dose will not be given. The dose is given post-observation and prediction. This means that predictions to observations at times where a bolus dose is also given are formed prior to the “injection” of the dose. The elements `State` and `Amount` specifies in which compartment/state the dose should be given and what amount is given. Multiple doses are allowed at the same time point. Infusions can be specified using the input element $\boldsymbol{u}$. The `covar` element contains the subject covariates (Z_i in Eq. (9)) and can be an array or list however the choice should be consistent with the referencing in the `h`function in Table 2.

3.3. *Package functionality*

Each of the three previously mentioned functionality parts is enclosed in a single function. The three functionality parts with corresponding functions are described in detail in the following sections.

3.3.1. *Simulation*

The function `PSM.simulate` performs the simulation of the system using an Euler based scheme. The simulation also includes inter-individual variation if the $\boldsymbol{\Omega}$ matrix is specified. The stochastic noise term in the system equation is included by perturbing the states after each Euler step. The size of the pertubation is found by randomly sampling from a multivariate normal distribution with covariance proportional to the time step scaled with the diffusion scaling term. The default time step in the Euler scheme is 0.1 unless specified differently. It is upon the user to ensure that the observation times are a multiple of the time step.

The function arguments to `PSM.simulate` are as follows:

Table 4 – Simulation output.

Element	Type
`Time`	Vector
`X`	Matrix with the simulated states
`Y`	Matrix with the simulated observations
`U`	Matrix with the input used in the simulation
`Dose`	Dose list used in the simulation
`eta`	Matrix with the random effects used in the simulation
`longX`	Matrix with states at all sub-sampled timepoints
`longY`	Matrix with observations at all sub-sampled timepoints

```
PSM.simulate(Model, Data, THETA, deltaTime,
longX)
```

where `Model` and `Data` are lists as described in previous sections. Any observations in `Data` are disregarded as the simulation returns a set of observations `Y`. `THETA` is a vector with the population parameters to be used. `deltaTime` is the time step in the Euler scheme and the `longX` option is used to indicate whether the output should include all generated data at all sub-sampled time points or only return observations at prespecified time points.

The output from the simulation function is a list of lists where each underlying list contains the data for one subject. The list contains the elements shown in Table 4.

3.3.2. *Estimation*

The function `PSM.estimate` performs maximum likelihood estimation for the population parameters in the model. The objective function for the optimization is the negative log likelihood as defined in Eq. (11). The function calculates the numerical gradients and determines the optimal random effects needed in optimization of the likelihood function.

Functionality has been included to allow for constrained optimization using the logit transformation. A logit parameter transformation is used to convert the bounded parameters to unbounded parameters. In order to stabilize the optimization with boundaries a penalty function has also been included. The penalty increases as the parameter estimate approaches the boundary. The penalty function is introduced to ensure that the optimization will not get trapped in the flat regions of the logit transformation. For very large values in the unbounded parameter domain the transformed parameter will be close to the upper boundary. This also means that changing an extreme value in the unbounded parameter domain will hardly change the bounded parameter estimate. The optimizer stops when a change in the unbounded parameters does not change the likelihood function. The penalty function stabilizes this problem.

The currently used optimizer does not allow for `NaN` to be returned from the likelihood function. The search path for the optimization can lead to parameter values that generate `NaN` resulting in the search failing. This problem can be solved by using tighter boundaries and restarting the optimization with new boundaries in the recently found parameter values.

The parameter estimation based on the likelihood function consists of nested optimizations which makes the likelihood function highly nonlinear in parameters. The optimizer does not guaranty that the found minimum is the global minimum so the user should be aware of local minima and the importance of initial parameter values in the optimization. The user should preferably start the minimization in different initial parameter values to eliminate the risk of using parameter estimates from a local minimum in the modelling onwards.

To evaluate the quality of the parameter estimates the related uncertainties can be calculated. The uncertainties are based on the Hessian of the likelihood function. The parameter confidence bands are returned from the estimation procedure by setting the argument `CI=TRUE`. The Hessian is calculated using the `numDeriv` package.

The argument list for the estimation can be seen below:

```
PSM.estimate(Model, Data, Par, CI, trace, con-
trol, fast)
```

where the `Model` and `Data` are as previously described. `Par` is a list containing the initial parameter value in `Init` and optionally the upper and lower boundaries in `UB` and `LB`. `CI` specifies if the confidence intervals for the parameters should be calculated. `trace` is an integer controlling the amount of output from the optimization. The `control` argument is passed directly on to the optimiser—for further details see the help files for `optim`. The `fast` argument specifies whether the FORTRAN code should be used when possible. This can be useful for debugging purposes.

The Kalman Filter has been implemented in FORTRAN for linear models with non-singular system matrix. Non-linear models and singular linear models are implemented in R-code. The matrix exponential used for solving linear systems is also implemented in FORTRAN. Hence linear models with full matrices are estimated faster than other models. For initial modelling purposes it can often be extremely helpful to convert a singular model into a non-singular by adding a small rate constant to the diagonal.

The output from the estimation function can be seen in Table 5.

3.3.3. *Smoothing*

The estimation procedure relies on one-step predictions of observations based on previous data but in order to determine the overall most likely profile based on all data smoothing can be employed. The inclusion of all data allows noise effects occurring later in the time series to influence the profile earlier

Table 5 – Estimation output.

Element	Description
`NegLogL`	Negative log likelihood in the found optimum
`THETA`	Vector of parameters in optimum
`CI`	Confidence intervals based on the Hessian
`SD`	Standard error for the optimal parameters
`COR`	Correlation matrix for the optimal parameters
`sec`	Optimization time in seconds
`opt`	Messages from the optimizer

Table 6 – Smooth output.

Element	Description
`Time`	Sub-sampled time
`Xs`, `Ps`	Smoothed states and uncertainty
`Ys`	Predictions based on smoothed states
`Xf`, `Pf`	Filtered state and uncertainty
`Xp`, `Pp`	One-step state predictions and uncertainty
`Yp`, `R`	One-step observation predictions and uncertainty
`eta`	Estimated random effects
`NegLogL`	Negative log likelihood

on. Smoothing constructs the optimal state vector to all time points given both prior and future observations as opposed to filtered states that only depend on prior observations. The smoothed estimate is commonly used in post-processing of data as it represents the best fit based on all available data.

The function `PSM.smooth` performs smoothing of states using a Bryson Frazier algorithm [13].

The smoothing function argument list is shown below:

```
PSM.smooth(Model, Data, THETA, subsample, trace,
etaList)
```

where `Model` and `Data` are as described ealier. `THETA` is a vector with the population parameters for the evaluation, i.e. the returned parameter vector from the estimation. `subsample` is the number of sub-samples in between observations. Sub-sampling can be used to display the system behaviour in between observations. `trace` is an integer controlling the amount of text output. `etaList` is a matrix, where each column is the random effects for a subject. If `etaList` is set to `NULL` the random effects will be determined.

The complete listing of output from the smoother is shown in Table 6.

Several internal functions are used in simulation, estimation and smoothing functions and they can all be found in the documentation. For general use the three described functions will form a good base.

In the next section an application of the package is described.

4. Application: insulin secretion rates

In the article by Mortensen [7] the insulin secretion rate is determined by stochastic deconvolution using a Matlab framework. The insulin secretion rates are modelled as random walks and the Kalman Filter is used to determine the trajectory that most likely resulted in the observations. This section describes an extension implementing an intervention type model as known from Time Series Analysis in order to better characterize the insulin secretion.

The challenge in describing the insulin secretion is that the kinetic system for insulin is potentially non-linear and partly unknown. This makes insulin itself a poor descriptor for insulin secretion. During the production of insulin a by-product called connecting peptide (C-peptide) is produced in equimolar amounts. Insulin and C-peptide are split just as insulin is secreted into systemic circulation.

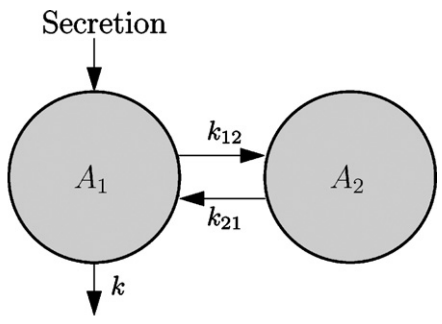


Fig. 2 – C-peptide PK model.

The pharmacokinetic system for C-peptide has been described in a population model by Van Cauter [14] with parameters based on subject covariates. The model structure is a linear two compartment model with elimination from the central compartment. The well known kinetics and longer half-life of C-peptide makes it a better descriptor of the actual insulin secretion even though the determined secretion rates are C-peptide secretion rates and not insulin secretion rates.

The graphical representation of the C-peptide model can be seen in Fig. 2. The exchange rate parameters and the elimination rate are all first order and the mathematical equations are given in (14)–(16):

$$\frac{dA_1}{dt} = SR - (k + k_{12})A_1 + k_{21}A_2 \tag{14}$$

$$\frac{dA_2}{dt} = k_{12}A_1 - k_{21}A_2 \tag{15}$$

$$y = \frac{A_1}{V} + e \tag{16}$$

where A_1 and A_2 are amounts in compartment 1 and 2. SR is the secretion rate measured as [amount/ min]. k, k_{12} and k_{21} are rate constants [min^{-1}]. V is the distribution volume [L].

The data originates from a meal tolerance test where the test subject is served three standardized meals over a period of 24 h. The insulin, C-peptide and glucose concentrations are measured throughout the 24 h and more frequently during meals. The C-peptide profiles can be seen in Fig. 3. The subjects are newly diagnosed type II diabetes patients with relatively intact insulin secretion. One complication with type II diabetes is decreasing beta cell mass, i.e. decreasing insulin secretion. To determine if the insulin secretion is intact, stochastic deconvolution [15] can be used.

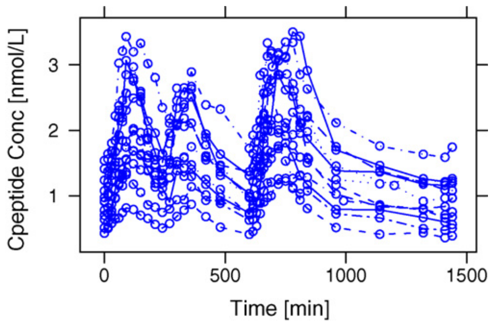


Fig. 3 – C-peptide profiles.

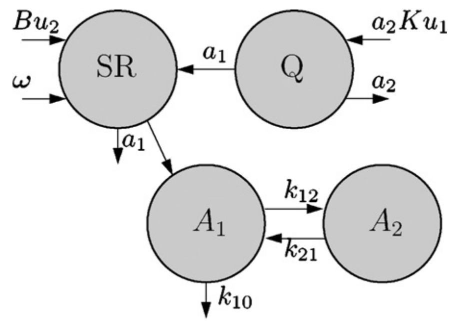


Fig. 4 – Intervention C-peptide model.

In the article [7] the insulin secretion is assumed to be a random walk and the most plausible trajectory is determined. The smoothed estimate is a compromise between the expected variation in the system and the observed variation in the observations. The scaling diffusion term $\boldsymbol{\sigma}$ is overestimated due to the physiological structure of the insulin secretion, where the secretion can only assume positive values. To remedy this assumption an intervention model is added to aid the structure of the secretion. The intervention model scheme is implemented by a step function located at meal time. A two compartment structure is assumed to allow for some flexibility in the form of the secretion rate profile. The secretion is furthermore assumed to have an underlying basal secretion rate. The basal rate, the rate parameters and the amplification are now estimated in this underlying structure for the secretion and the random walk is used to determine the deviations from this structure (Fig. 4).

The states of the system for further use in the equations are defined as below:

$$\boldsymbol{x} = [A_1, A_2, SR, Q]^T \tag{17}$$

The mathematical equations describing the system can now be written as

$$d\boldsymbol{x} = \left(\begin{bmatrix} -(k_{12}+k_{10}) & k_{21} & 1 & 0 \\ k_{12} & -k_{21} & 0 & 0 \\ 0 & 0 & -a_1 & a_1 \\ 0 & 0 & 0 & -a_2 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & B\exp(\eta_2) \\ a_2K\exp(\eta_1) & 0 \end{bmatrix} \boldsymbol{u} \right) dt + \text{diag} \begin{bmatrix} 0 \\ 0 \\ \sigma_{SR} \\ 0 \end{bmatrix} d\boldsymbol{\omega} \tag{18}$$

where k_x and a_x are rate constants. The input variable $\boldsymbol{u}_t = [u_1, u_2]'$ is used to model the baseline level of secretion (u_1) and the impulse effect from a meal (u_2). u_1 is equal to one for the entire time series whereas u_2 is only equal to one just after meals (i.e. from meal start and 30 min on). The B parameter specifies the baseline level in the secretion compartment and K specifies the amplitude of the meal impulse. Both B and K are modelled with an individual random effect (η_1, η_2). σ_{SR} is the scaling diffusion term for remaining description of the secretion rate.

The observation equation linking the model states to the C-peptide observations is shown below:

$$y = \left[\frac{1}{V}, 0, 0, 0\right] \boldsymbol{x} + e = \frac{A_1}{V} + e, \qquad \text{where} \quad e \in N(0, S) \tag{19}$$

This model is a simplification as the secretion responses to the meals are assumed equal over all three meals. An extension to make an individual secretion response per meal can be made by extending the input and the parameter list accordingly. This will however increase the number of parameters and thereby the estimation time.

Individual random effects have been added to the amplification of the response and the basal level so that each individual can have different secretion responses.

The parameters in the model to be estimated are the secretion parameters a_1, a_2, K, B and the variance parameter σ_{SR}. The inter-individual variance in $\boldsymbol{\Omega}$ is assumed to be 0.1. The residual variance is assumed to be $(50\,\text{pmol/L})^2$ which was derived from the plot of the profiles.

4.1. *Model specification*

The model specification in PSM is described element by element in the next section.

```
> Cpep.Model <- list()
> Cpep.Model$h = function(eta,theta,covar) {
    theta$K12 <- covar[1]
    theta$K21 <- covar[2]
    theta$K10 <- covar[3]
    theta$V   <- covar[4]
    theta$K   <- theta$K*exp(eta[1])
    theta$B   <- theta$B*exp(eta[2])
    phi       <- theta }   }
```

Initially the model is setup by creating an empty list. The h function that translates the population parameters into individual parameters is specified. Function arguments that can be used in the creation of phi are population parameters, random effects and covariates. It can be seen that four individual parameters are extracted from the covariates and the two population parameters K and B are expanded with random effects.

```
> Cpep.Model$Matrices = function(phi) {
    K21  <- phi$K21  ;  K10 <- phi$K10
    K12  <- phi$K12  ;  a1  <- phi$a1
    a2   <- phi$a2   ;  V   <- phi$V
    matA <- matrix(
            c(-(K10+K12),K21,1,0,
        K12,-K21,0,0,
        0,0,-a1,a1,
        0,0,0,-a2),nrow=4,byrow=T)
matB <- matrix(
          c(0,0,0,a2*phi$K,
          0,0,phi$B,0), nrow=4)
matC <- matrix(c(1/V,0,0,0),nrow=1)
matD <- matrix(rep(0,2),nrow=1)
list(matA=matA,matB=matB,
     matC=matC,matD=matD)}
```

The first element created here is the time invariant matrices. In this example all four matrices need to be specified. First the individual parameters are extracted from `phi` and matrices are set up in a list named `Matrices`.

```
Cpep.Model$X0 = function(Time=Na,phi,U=Na) {
    K21  <- phi$K21  ;  K10 <- phi$K10
    K12  <- phi$K12
    B    <- phi$B    ;  a1  <- phi$a1
  matrix(c(B/(a1*K10),
           (B*K12)/(a1*K10*K21),
           B/a1,
           0),nrow=4) }
```

The initial conditions for the states are added to the list as an element named `X0`. The initial conditions used here are steady state conditions calculated using the basal secretion and kinetic parameters. The initial conditions are specified as a function with arguments `Time`, `phi` and `U`. The `Time` argument is the first time point specified and can be useful if the subjects start at different time points. The `U` can contain exogenous input to the system which enters into the initial conditions.

```
      Cpep.Model$SIG = function(phi) {
  SIG       <- matrix(rep(0,16),nrow=4)
  SIG[3,3]  <- phi$SIG33
  SIG }
Cpep.Model$S = function(phi) { matrix(phi$S)}
```

Table 7 – Estimated parameters.

Parameter	MLE	95% CI
`K`	1911.2	[1511.5; 2310.9]
`B`	7.019	[5.0; 9.037]
`a1`	0.029	[0.022; 0.036]
`a2`	0.011	[0.0089; 0.014]
`SIG33`	30.3	[26.05; 34.55]

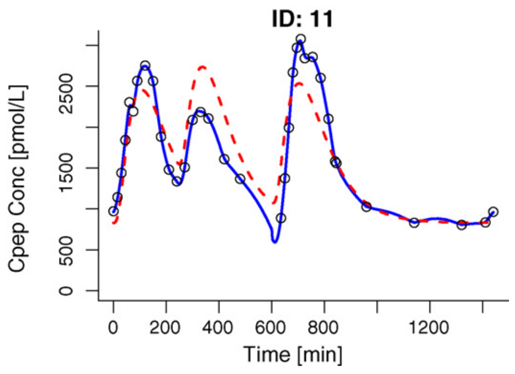


Fig. 5 – Fitted C-peptide concentrations for subject 11.

The diffusion scaling term and the residual variance are specified using the elements `SIG` and `S`. Both are specified as functions of `phi`. It can be seen that `SIG` is a 4×4 matrix with only an element at position [3, 3]. `S` is specified as a matrix even though it is one-dimensional.

```
> Cpep.Model$ModelPar = function(THETA){
  list(theta=list(K=THETA["K"],B=THETA["B"],
  a1=THETA["a1"],a2=THETA["a2"],S=50^2,
  SIG33=THETA["SIG33"]),
  OMEGA=diag(c(.1,.1))) }
```

The final element in the model is the function that splits the parameter vector containing parameters to be optimized into population parameters and inter individual covariance matrix $\boldsymbol{\Omega}$.

The list `Cpep.Model` now contains all the elements required in the model specification and the model can now be used to estimate parameters and create the smoothed profiles.

4.2. *Results*

The parameters in the C-peptide intervention model are estimated using `PSM.estimate` with constraints on the parameters. The optimal parameters and confidence bands are shown in the Table 7.

Two model fits for C-peptide can be seen in Figs. 5 and 6. The plots show the observations as circles and the intervention

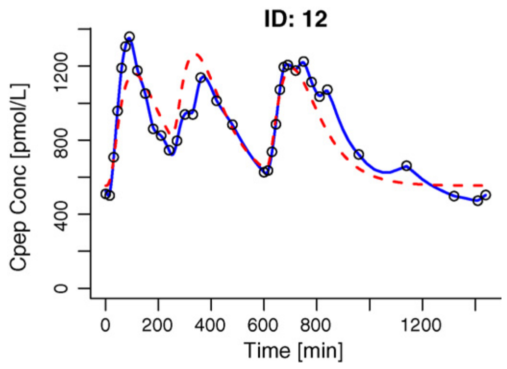


Fig. 6 – Fitted C-peptide concentrations for subject 12.

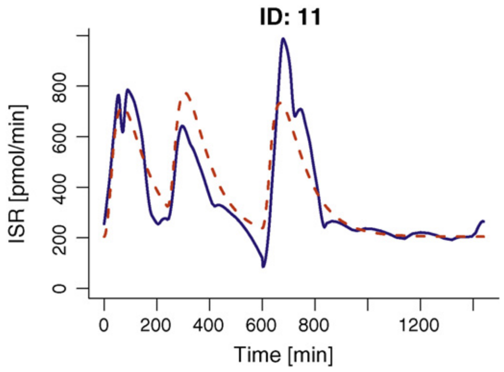


Fig. 7 – Secretion for subject 11.

model fit is shown in blue. The dashed line represents the deterministic model fit where the diffusion scale term has been fixed to zero in the intervention model in order to mimic an ODE model like in NONMEM. The dashed line is thus equal to a simulation with the intervention model.

The failure of the assumption of equal responses to all meals is obvious in the fit between the dashed red line and the observations. Large peaks are underestimated and small peaks overestimated. This is clear as the model describes the average response after a meal. The solution could be to extend the model so that every meal has it's own amplification.

As the model is used for deconvolution purposes the actual fit to the observations is of less importance, but more interesting is the secretion rate profile and the split between the model and the Wiener component.

The two corresponding secretion rate profiles to the C-peptide fits are found in Figs. 7 and 8. The full line is the optimal secretion rate determined by the Kalman Filter and the dashed line represents the deterministic part of the model. The secretion model captures the overall trends but the compromise with the equal response assumption is clear.

The deconvoluted insulin secretion rates have some jumps which seem unphysiological. The solution could be to use an integrated random walk as driver for insulin secretion. This would make the deconvoluted insulin secretion rate less erratic.

This section has shown a simple application of the PSM package for modelling purposes. Classic log likelihood ratio testing can also be applied in the model development as well as visual predictive checks of the model fits.

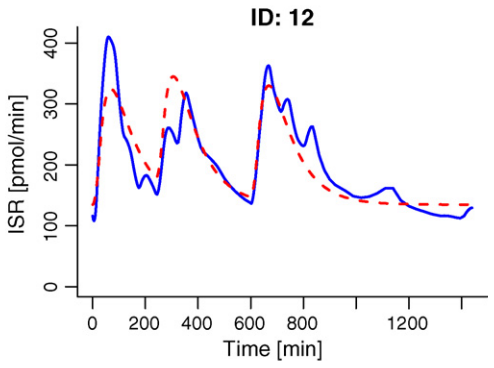


Fig. 8 – Secretion for subject 12.

5. Discussion and conclusion

The PSM package provides a new general framework for handling dynamic models based on stochastic differential equations in a population setting. The package is available in R to ease availability, ease installation and enable a single working environment for data handling, modelling and visualization.

The package is an extension to the Matlab framework of the same name. The package functionality is a combination of the functionality available in CTSM and NONMEM. CTSM and NONMEM have also been used to validate the package as described further on the homepage [16].

The package enables the feature of dosing capabilities which makes the package useful in PK/PD model development. This is further supported by the ability to handle multidimensional observations which aids in modelling work with observations from both PK and PD.

The package includes functionality for modelling tasks such as simulation, parameter estimation and smoothing. The stochastic deconvolution example with an underlying secretion model in this article showed an application of the package.

The computational effort in working with larger models is substantial and the use of parallelization could decrease the time considerably in the modelling. It is currently being investigated how to implement parallelization in a general manner in R. Parallelization is an obvious solution due to the fact that future computer systems will be massive multicore systems.

The package is a promising tool to get started with stochastic differential equations or the inclusion of mixed-effects in continuous-discrete state-space models.

More information can be found on the webpage http://www.imm.dtu.dk/psm.

Acknowledgement

This work was supported by the European Union through the Network of Excellence BioSim, contract no. LSHB-CT-2004-005137.

REFERENCES

[1] N.R. Kristensen, H. Madsen, S.H. Ingwersen, Using stochastic differential equations for PK/PD model development, J. Pharmacokinet. Pharmacodyn. 32 (February(1)) (2005) 109–141.
[2] C.W. Tornøe, R.V. Overgaard, H. Agersø, H.A. Nielsen, H. Madsen, E.N. Jonsson, Stochastic differential equations in NONMEM: implementation, application, and comparison with ordinary differential equations, Pharm. Res. 22 (August(8)) (2005) 1247–1258.
[3] R.V. Overgaard, N. Jonsson, C.W. Tornøe, H. Madsen, Non-linear mixed-effects models with stochastic differential equations: implementation of an estimation algorithm, J. Pharmacokinet. Pharmacodyn. 32 (February(1)) (2005) 85–107.
[4] U. Picchini, S. Ditlevsen, A. De Gaetano, Maximum likelihood estimation of a time-inhomogeneous stochastic differential

model of glucose dynamics, Math. Med. Biol. 25 (June(2)) (2008) 141–155.

[5] L.B. Sheiner, S.L. Beal, NONMEM User's Guide, NONMEM Project Group, University of California, San Francisco, 1994.

[6] N.R. Kristensen, CTSM Mathematics Guide, Technical Report, Technical University of Denmark, December 2003.

[7] S.B. Mortensen, S. Klim, B. Dammann, N.R. Kristensen, H. Madsen, R.V. Overgaard, A Matlab framework for estimation of NLME models using stochastic differential equations: applications for estimation of insulin secretion rates, J. Pharmacokinet. Pharmacodyn. 34 (October(5)) (2007) 623–642.

[8] C.W. Tornøe, H. Agersø, E. Niclas Jonsson, H. Madsen, H.A. Nielsen, Non-linear mixed-effects pharmacokinetic/pharmacodynamic modelling in NLME using differential equations, Comput. Meth. Programs Biomed. 76 (October(1)) (2004) 31–40.

[9] R.E. Kalman, New approach to linear filtering and prediction problems, Am. Soc. Mech. Eng. – Trans. – J. Basic Eng. Ser. D 82 (1) (1960) 35–45.

[10] G. Welch, G. Bishop, http://www.cs.unc.edu/~welch/kalman/.

[11] R.V. Overgaard, Pharmacokinetic/Pharmacodynamic Modeling with a Stochastic Perspective. Insulin Secretion and Interleukin-21 Development as Case Studies, PhD Thesis, Technical University of Denmark, Informatics and Mathematical Modelling, 2006.

[12] L.B. Sheiner, S.L. Beal, Evaluation of methods for estimating population pharmacokinetics parameters. I. Michaelis–Menten model: routine clinical pharmacokinetic data, J. Pharmacokinet. Biopharm. 8 (December(6)) (1980) 553–571.

[13] T. Kailath, A.H. Sayed, B. Hassibi, Linear Estimation, Prentice-Hall, 2000.

[14] E. Van Cauter, F. Mestrez, J. Sturis, 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 (March(3)) (1992) 368–377.

[15] N.R. Kristensen, A deconvolution method for linear and nonlinear systems based on stochastic differential equations, Population Approach Group Europe, 2004.

[16] S. Klim, S.B. Mortensen, PSM homepage. Internet: http://www.imm.dtu.dk/psm, June 2008.