NLMEModeling: A Wolfram Mathematica Package for Nonlinear Mixed Effects Modeling of Dynamical Systems

Jacob Leander[1,2,3,✉], Joachim Almquist[3], Anna Johnning[1],

Julia Larsson[1,2], and Mats Jirstrand[1]

[1]Fraunhofer-Chalmers Centre, Chalmers Science Park, 412 88 Gothenburg, Sweden
[2]Department of Mathematical Sciences, Chalmers University of Technology and University of Gothenburg, 412 96 Gothenburg, Sweden
[3]Clinical Pharmacology & Quantitative Pharmacology, Clinical Pharmacology & Safety Sciences, AstraZeneca R&D, Gothenburg, Sweden

✉ jacob.leander@fcc.chalmers.se

Abstract

Nonlinear mixed effects modeling is a powerful tool when analyzing data from several entities in an experiment. In this paper, we present NLMEModeling, a package for mixed effects modeling in Wolfram Mathematica. NLMEModeling supports mixed effects modeling of dynamical systems where the underlying dynamics are described by either ordinary or stochastic differential equations combined with a flexible observation error model. Moreover, NLMEModeling is a user-friendly package with functionality for model validation, visual predictive checks and simulation capabilities. The package is freely available and provides a flexible add-on to Wolfram Mathematica.

Keywords: Nonlinear mixed effects, Dynamical models, Ordinary differential equations, Stochastic differential equations, Wolfram Mathematica, First-order conditional estimation (FOCE), Modeling software

1 Introduction

In several applications, repeated measurements are collected from a number of entities to study a specific system of interest. The nonlinear mixed effects (NLME) model is a popular statistical framework able to quantify variability in response, especially popular in pharmacometrics and drug development. Typically, the underlying system of interest is described by a system of ordinary differential equations in combination with an observation model. In recent years there has been an increasing interest in extending the NLME framework to incorporate stochastic differential equations, leading to a class of models called stochastic differential equations mixed effects models (SDEMEMs) [1, 2, 3, 4, 5, 6]. There are several software options available for parameter estimation in NLME models with ordinary differential equations, including both commercial software such as NONMEM [7, 8, 9], Monolix [10], and Phoenix, and open-source such as nlmixr [11, 12] and Stan [13]. However, the number of options available for SDEMEMs is limited [14, 15, 16]. A programming platform well suited for modeling and scientific computation is Wolfram Mathematica [17] but currently there is no functionality for performing mixed effects modeling in Wolfram Mathematica.

In this paper, we present the software package NLMEModeling used for NLME modeling of dynamical systems in Wolfram Mathematica. The NLMEModeling package provides an easy-to-use, integrated NLME modeling environment in Wolfram Mathematica. The current version supports dynamical models with mixed effects where the dynamical system is described by either ordinary or stochastic differential equations. By utilizing the symbolic calculation capabilities and compact syntax in Wolfram Mathematica, a user-friendly package is provided. Moreover, the user can develop additional functionality to tailor the package towards its own needs. The modeling framework has previously been applied in several applications, including oncology [18, 19, 20], single cell experiments [21], and pharmacokinetic and pharmacodynamic modeling [22, 23, 24, 25], but the package has not previously been publicly available. The purpose of this paper is to present the NLMEModeling package from a user perspective and to

introduce interested users of Wolfram Mathematica to the field of NLME modeling, but not to compare the package with existing software in terms of computational speed and robustness.

The paper is organized in the following way. First, the mathematical theory behind the mixed effects framework based on ordinary and stochastic differential equations is introduced. Second, the functionality of the NLMEModeling package is illustrated with three examples of different complexity. The first example is a pharmacokinetic (PK) model which serves to introduce the model syntax, parameter estimation, and model validation. The more advanced examples consist of pharmacokinetic-pharmacodynamic models with multiple observation variables and random effects, in an ordinary as well as a stochastic differential equation setting. We also introduce the concept of so called phi-parameters, which is a convenient way of introducing the relation between different parameters types in a mixed effects model. Finally, the modeling package is discussed and future extensions are outlined.

2 Methods

The mathematical foundations of NLMEModeling will be provided by first introducing the NLME model. Second, we describe how the estimation of model parameters is done according to the maximum likelihood approach and give a high-level description of the numerical methods used. A more detailed description of the methodology can be found in [26, 27]. Instructions for retrieving and installing the package are also provided.

2.1 The Nonlinear Mixed Effects Modeling Framework

The statistical model

We consider models where the underlying system is described by either a system of ordinary differential equations (ODEs) or stochastic differential equations (SDEs). In the case of ODEs, the dynamical system for individual i is governed by the ordinary differential equation

$$d\boldsymbol{x}_i = \boldsymbol{f}(\boldsymbol{x}_i, t, \boldsymbol{u}_i, \boldsymbol{\theta}, \boldsymbol{\eta}_i)dt, \quad \boldsymbol{x}_i(t_0) = \boldsymbol{x}_0(\boldsymbol{u}_i, \boldsymbol{\theta}, \boldsymbol{\eta}_i), \tag{1}$$

where $\boldsymbol{x}_i$ is a vector of state variables, $\boldsymbol{u}_i$ is a vector of system input or covariates, $\boldsymbol{\theta}$ a vector of fixed effects parameters, and $\boldsymbol{\eta}_i$ a vector of random effects. The random effects $\boldsymbol{\eta}_i$ are assumed to be multivariate normal distributed with mean zero and covariance matrix $\boldsymbol{\Omega}$,

$$\boldsymbol{\eta}_i \sim N(\boldsymbol{0}, \boldsymbol{\Omega}). \tag{2}$$

In the case of stochastic dynamics, the underlying model is described by an SDE on the form

$$d\boldsymbol{x}_i = \boldsymbol{f}(\boldsymbol{x}_i, t, \boldsymbol{u}_i, \boldsymbol{\theta}, \boldsymbol{\eta}_i)dt + \boldsymbol{G}(\boldsymbol{x}_i, t, \boldsymbol{u}_i, \boldsymbol{\theta}, \boldsymbol{\eta}_i)d\boldsymbol{W}_i, \quad \boldsymbol{x}_i(t_0) = \boldsymbol{x}_0(\boldsymbol{u}_i, \boldsymbol{\theta}, \boldsymbol{\eta}_i), \tag{3}$$

where $\boldsymbol{G}$ is a weighting matrix and $\boldsymbol{W}_i$ is a standard Wiener process with increments $d\boldsymbol{W}_i \sim N(\boldsymbol{0}, dt\boldsymbol{I})$ with $\boldsymbol{I}$ being the identity matrix. A model for the observations of the dynamical system is given by

$$\boldsymbol{y}_{ij} = \boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{u}_i, t_{ij}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) + \boldsymbol{e}_{ij}, \tag{4}$$

where the vector $\boldsymbol{y}_{ij}$ denotes the jth observation for the ith individual. We let N denote the total number of individuals and n_i denote the total number of observations for individual i. In the observation model $\boldsymbol{e}_{ij}$ are assumed to be multivariate normal distributed according to $\boldsymbol{e}_{ij} \sim N(\boldsymbol{0}, \boldsymbol{\Sigma}(\boldsymbol{x}_i, t_{ij}, \boldsymbol{u}_i, \boldsymbol{\theta}, \boldsymbol{\eta}_i))$.

Derivation of the likelihood function

Given an NLME model and a set of observations, we are interested in estimating the fixed effects $\boldsymbol{\theta}$, the random effects covariance matrix $\boldsymbol{\Omega}$, and the observation error covariance matrix $\boldsymbol{\Sigma}$. To simplify the notation, we use $\boldsymbol{\theta}$ to denote all parameters of interest, including parameters in $\boldsymbol{\Sigma}$, $\boldsymbol{\Omega}$, and, in the case of stochastic dynamics, also the matrix $\boldsymbol{G}$.

NLMEModeling estimates the model parameters using a method where the likelihood is approximated using a second-order Taylor expansion. The approximation of the likelihood has been derived previously [1, 23, 28] and we will, therefore, only give a brief overview of the derivation.

We let $\mathcal{Y}_{i,n_i} = \{\boldsymbol{v}_{i,1}, \boldsymbol{v}_{i,2}, \ldots, \boldsymbol{v}_{i,n_i}\}$ denote the collection of all observations for individual i and let $\mathcal{Y} = \{\mathcal{Y}_{1,n_1}, \mathcal{Y}_{2,n_2}, \ldots, \mathcal{Y}_{N,n_N}\}$ denote the totality of observations. The residuals $\boldsymbol{\epsilon}_{ij}$ are defined as

$$\boldsymbol{\epsilon}_{ij} = \boldsymbol{v}_{ij} - \hat{\boldsymbol{y}}_{ij} \tag{5}$$

where the expected observation value $\hat{\boldsymbol{y}}_{ij}$ and covariance $\boldsymbol{R}_{ij}$ are given by

$$\hat{\boldsymbol{y}}_{ij} = E[\boldsymbol{y}_{ij}|\mathcal{Y}_{i(j-1)}] \tag{6}$$

$$\boldsymbol{R}_{ij} = Cov[\boldsymbol{y}_{ij}|\mathcal{Y}_{i(j-1)}] \tag{7}$$

For ODE models, $\boldsymbol{R}_{ij}$ is equal to the observation error covariance matrix $\boldsymbol{\Sigma}$ as the dynamical model is deterministic. For SDE models, on the other hand, NLMEModeling utilizes the extended Kalman filter [29] to estimate $\hat{\boldsymbol{y}}_{ij}$ and $\boldsymbol{R}_{ij}$ [23, 27, 30].

Since the random effects are unobserved quantities, we marginalize over the random effects to obtain an expression of the likelihood function that depends only on $\boldsymbol{\theta}$. Assuming independence between individuals, we have

$$L(\boldsymbol{\theta}|\mathcal{Y}) = \prod_{i=1}^{N} \int p(\mathcal{Y}_{i,n_i}|\boldsymbol{\theta}, \boldsymbol{\eta}_i) p(\boldsymbol{\eta}_i) d\boldsymbol{\eta}_i = \prod_{i=1}^{N} \int \exp(l_i) d\boldsymbol{\eta}_i. \tag{8}$$

In the expression above, $l_i = l_i(\boldsymbol{\eta}_i)$ is the *a posteriori* log-likelihood for the random effects of the ith individual given by

$$l_i = -\frac{1}{2} \sum_{j=1}^{n_i} \left(\boldsymbol{\epsilon}_{ij}^T \boldsymbol{R}_{ij}^{-1} \boldsymbol{\epsilon}_{ij} + \log\det\left(2\pi \boldsymbol{R}_{ij}\right) \right) - \frac{1}{2} \boldsymbol{\eta}_i^T \boldsymbol{\Omega}^{-1} \boldsymbol{\eta}_i - \frac{1}{2} \log\det\left(2\pi\boldsymbol{\Omega}\right). \tag{9}$$

In most cases, there is no closed-form expression for the integral in Equation (8), but it can be approximated using the Laplace approximation [28, 31]. The Laplace approximation uses a second-order Taylor expansion of l_i around a point $\boldsymbol{\eta}_i^*$. Here, the point is chosen to be the value of $\boldsymbol{\eta}_i$ which maximizes the *a posteriori* log-likelihood in Equation (9),

$$\boldsymbol{\eta}_i^* = \arg\max_{\boldsymbol{\eta}_i} l_i(\boldsymbol{\eta}_i). \tag{10}$$

Using this $\boldsymbol{\eta}_i^*$, the approximate population likelihood, L_L, becomes

$$L(\boldsymbol{\theta}|\mathcal{Y}) \approx L_L = \prod_{i=1}^{N} \left(\exp(l_i(\boldsymbol{\eta}_i^*)) \det\left[\frac{-\Delta l_i(\boldsymbol{\eta}_i^*)}{2\pi} \right]^{-\frac{1}{2}} \right), \tag{11}$$

where $\Delta l_i(\boldsymbol{\eta}_i^*)$ denotes the Hessian of l_i with respect to $\boldsymbol{\eta}_i$ evaluated at $\boldsymbol{\eta}_i^*$. By taking the logarithm of L_L, the approximate population log-likelihood becomes

$$\log L_L = \sum_{i=1}^{N} \left(l_i(\boldsymbol{\eta}_i^*) - \frac{1}{2} \log\det\left[\frac{-\Delta l_i(\boldsymbol{\eta}_i^*)}{2\pi} \right] \right). \tag{12}$$

Dependent on the number of terms that is kept in the expression of the Hessian of the individual log-likelihood, the first-order conditional estimation (FOCE) and the FOCE with interaction method (FOCEI) are obtained, where the latter is used in NLMEModeling. For more information regarding the methods, we refer the reader to [23, 28].

2.2 Gradient Based Optimization

The maximum likelihood estimate is obtained my maximizing the log-likelihood with respect to the model parameters

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \log L_L, \tag{13}$$

which is achieved using a gradient-based optimization method called the Broyden-Fletcher-Goldfarb-Shanno (BGFS) algorithm [32]. The BFGS is an iterative algorithm for solving unconstrained nonlinear optimization problems. It belongs to the family of quasi-Newton methods, which is a collection of methods that seek a stationary point of a function. It should be noted that the BGFS algorithm is a local optimization method, meaning that global optimum is not guaranteed.

In NLMEModeling the gradient of the objective function is calculated using so called exact gradients. Instead of the commonly used finite difference approach, exact gradients are calculated using the analytical solution to the gradient. The derivation requires symbolic differentiation of the model with respect to the model parameters and this is achieved using the symbolic engine in Wolfram Mathematica. For derivation of the exact gradients in NLMEModeling, we refer the interested reader to [26, 27].

To obtain the uncertainty in the estimated parameters NLMEModeling uses the variance-covariance matrix of the estimated parameters, given by the negative inverse of the Hessian matrix at the optimum.

The point $\boldsymbol{\eta}_i^*$ calculated given the optimal parameters values $\boldsymbol{\theta}^*$ is the most likely parameters for each subject. These values are referred to as Empirical Bayes Estimates (EBEs).

2.3 Differential Equation Solver

To solve the system of ordinary differential equations, NLMEModeling utilizes the Mathematica NDSolve framework. NDSolve is a collection of several differential equation solvers, including adaptive solvers with the ability to automatically detect and switch solver for stiff and non-stiff problems. This is important since the solution to the ordinary differential equations might behave differently for different values of the model parameters.

For the simulation of stochastic differential equations, NLMEModeling uses the Euler-Maruyama approximation [33], which is implemented as part of the package.

2.4 Model Evaluation

Functionality for goodness-of-fit analysis and model validation is provided in NLMEModeling. This includes population predictions versus observations, individual predictions versus observations, individual weighted residuals versus time, and individual weighted residuals versus individual predictions. In addition, several diagnostics of the EBEs are provided as well as functionality for visual predictive checks. The VPC functionality supports both the standard VPC plot and prediction-corrected VPC. For additional information regarding the evaluation of nonlinear mixed effects model we refer the reader to [34]. For details regarding the VPC methods, we refer the reader to [35].

2.5 Installing the Package

The latest release of NLMEModeling can be obtained from `http://www.fcc.chalmers.se/software/other-software/nlmemodeling` and requires an installation of Mathematica 8.0 or later. The package is installed using the installation functionality in Mathematica found under File -> Install. In the Type of Item dropdown menu, choose Application. In the Source dropdown menu, choose From File. Choose OK and the package is installed. To load the package, start a new Mathematica session and run the command

```
In[•]:= Needs["NLMEModeling`"]
```

3 An Introductory Example

This section is meant to serve as a tutorial on the NLMEModeling package. The modeling workflow is illustrated using a simple PK model with the purpose of introduce the syntax, main functions, and the resulting model object. In the Advanced examples section, two additional model examples of increasing complexity are provided.

3.1 The Model

In this hypothetical example we consider a one-compartmental PK model with oral administration of a drug. We use A_1 and A_2 to denote the amount of drug in the absorption and central compartment, respectively. The absorption is modelled by a first-order process with rate k_a (unit h^{-1}), and the elimination is described by the clearance CL (unit L/h) and volume of distribution V (unit L). The system of ordinary differential equations is given by

$$\frac{dA_1}{dt} = -k_a A_1(t),\ A_1(0) = Dose \tag{14}$$

$$\frac{dA_2}{dt} = k_a A_1(t) - \frac{CL_{ind}}{V} A_2(t),\ A_2(0) = 0. \tag{15}$$

The individual clearance CL_{ind} is log-normally distributed between individuals,

$$CL_{ind} = CL \exp(\eta_1) \tag{16}$$

where η_1 is a random effect that is normally distributed with mean 0 and variance ω_1^2. Observations of the concentration in the central compartment,

$$c(t) = \frac{A_2(t)}{V}, \tag{17}$$

are taken at time points 0.25, 0.5, 1, 1.5, 2, 3, 4, 6, 8, 12, 18, and 24 hours after dose, and modelled according to a proportional error model

$$y(t) = c(t) + e_t,\ e_t \sim N(0, (\sigma_{prop} c(t))^2). \tag{18}$$

The population in this example consists of 20 individuals, each administrated a dose of 100 mg. Parameter values and Mathematica code for simulating the dataset are available in Appendix 1.

3.2 Dataset Structure

In NLMEModeling, the dataset for a population of individuals is handled by the List entity in Mathematica. To illustrate the dataset structure, we import the dataset generated in Appendix 1 and look in detail at the data for the first subject in the dataset.

```
In[•]:= data = Import[NotebookDirectory[] <> "Example1-data.m"];
        data[[1]]
        data[[1, 1]]
        data[[1, 2]]

Out[•]= {{{0.25, {0.357399}}, {0.5, {0.793175}}, {1, {1.04842}}, {1.5, {1.61639}}, {2, {2.26188}},
          {3, {1.33062}}, {4, {1.30948}}, {6, {1.09769}}, {8, {0.783039}}, {12, {0.391855}},
          {18, {0.172501}}, {24, {0.0756571}}}, {SubjectID → 1, Dose → 100, Weight → 73}}

Out[•]= {{0.25, {0.357399}}, {0.5, {0.793175}}, {1, {1.04842}},
         {1.5, {1.61639}}, {2, {2.26188}}, {3, {1.33062}}, {4, {1.30948}}, {6, {1.09769}},
         {8, {0.783039}}, {12, {0.391855}}, {18, {0.172501}}, {24, {0.0756571}}}

Out[•]= {SubjectID → 1, Dose → 100, Weight → 73}
```

As shown in the output above, the individual dataset consists of two different lists: the observation list and the rule list. Note that the observation at each time point is vector-valued, which allows for multiple variables to be measured. Moreover, the rule list is flexible and can include individual covariates (e.g. ID, weight, or sex) or more complex rules such as solutions to ordinary differential equations (InterpolatingFunction objects) or other time-dependent expressions (e.g. dosing information).

The population dataset can easily be plotted and explored using built-in functionality in NLMEModeling. It is also possible for the user of NLMEModeling to develop additional code that utilizes the pre-defined data structure. The simulated PK data for this introductory example is depicted in Figure 1.

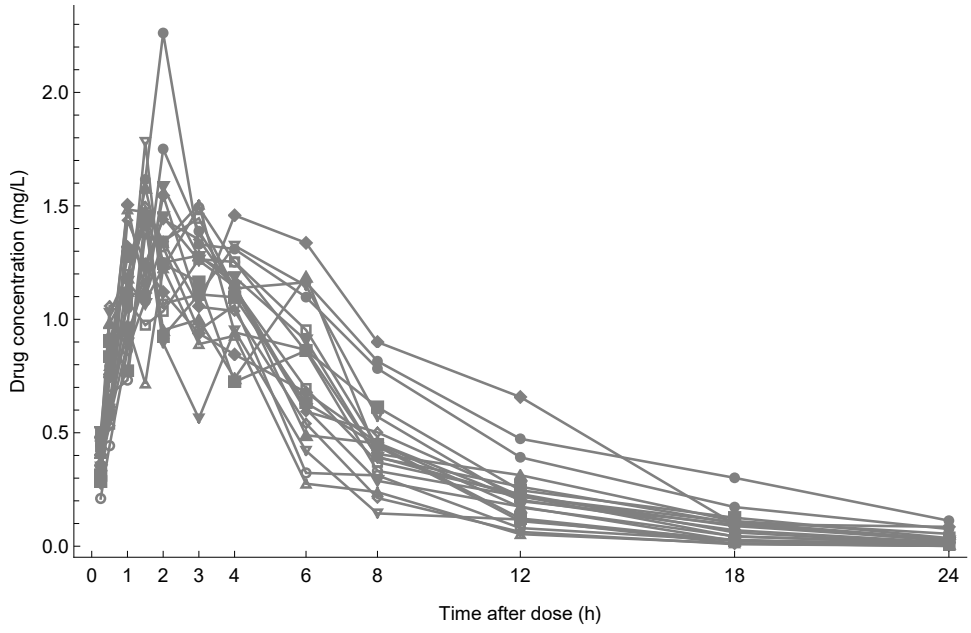


Figure 1: Simulated data for the PK experiment in Example 1.

3.3 Estimating Model Parameters

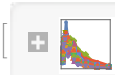
We now turn our attention to estimating model parameters given the observed data. To define the model in NLMEModeling, we need to define the dynamical model and the observation model. We will estimate model parameters using the same model structure that was used to generate the data.

We define the dynamical model as a system of ordinary differential equations using the standard notation. Note the possibility to explicitly state known relations of interest: in this example the definition of the concentration in the central compartment. We also define the observational model, here equal to the concentration in the central compartment.

```
In[•]:= sys = {
          A1'[t] == -ka * A1[t],
          A2'[t] == ka * A1[t] - CL * Exp[η1] / V * A2[t],
          A1[0] == Dose,
          A2[0] == 0,
          c[t] == A2[t] / V};
        obs = {c[t]};
```

The main function in the NLMEModeling package is named **NLMEDynamicalModelFit**, which is used to estimate the parameters in a mixed effects model given a set of observations from a number of individuals. The minimal function call requires the data, the dynamical model, the observation model, start values for the fixed effects parameters, and a list of the random effect parameters. In this example, we are considering a proportional error model and specify this as an option (the default Sigma option is an additive error model). For a full description of the available options of **NLMEDynamicalModelFit**, see Table 1.

```
In[•]:= modelFit = NLMEDynamicalModelFit[data, {sys, obs}, {{ka, 0.9}, {CL, 5}, {V, 30}}, η1,
          Sigma → "Proportional"]

Out[•]= FittedNLMEModel[  State variables: 2
                         Observables: 1 ]
```

Moreover, **NLMEDynamicalModelFit** is extendable to multiple model structures within the same parameter estimation problems. In that case, the arguments is a list of datasets together with a list of system and observational models. The different models may have distinct and/or shared random and fixed effect parameters while the random effects covariance matrix $\mathbf{\Omega}$ is shared between the models. For more information, we refer the reader to the package documentation.

3.4 The Model Object

NLMEDynamicalModelFit returns a FittedNLMEModel object, which contains information about the estimated model, including the estimated parameters, likelihood value, and model equations. See Table 2 for a full list of available properties that can be extracted from the model object.

Use of the property "ModelSummary" is a convenient way to get an overview of the estimated model, including the estimated parameters values, the estimated $\mathbf{\Sigma}$ and $\mathbf{\Omega}$ matrices, and the log-likelihood value. For easy comparison between different models, $-2 * \log L_L$ as well as the Akaike information criterion (AIC) and Bayesian information criterion (BIC) are provided [37, 38]

Table 1: Options to **NLMEDynamicalModelFit**. For details regarding each option, we refer the reader to the package documentation.

Option name	**Description**
Sigma	Structure of the observation error covariance matrix $\mathbf{\Sigma}$. The default option is a diagonal matrix.
Omega	Structure of the random effects covariance matrix $\mathbf{\Omega}$. The default option is a diagonal matrix.
PostiveParameters	List of strictly positive parameters, for which the parameter estimation is done on log-scale to improve numerical stability.
PerformParameterEstimation	Whether or not to perform parameter estimation
PerformCovarianceStep	Whether or not to perform covariance calculation
PrintToFile	Whether to print the optimization history to a file
OuterFile	File name of the history file to be saved.
OuterPrintOn	Specifies how often printout of optimization to the FrontEnd occurs.
Parallel	Whether to perform the parameter estimation in parallel
Precondition	Precondition of the covariance step. For details regarding the method, see paper by Aoki *et al.* [36]
PrecisionGoalOuterEstimation	Number of absolute digits in the parameter estimation.
RelativeDigitsOuterEstimation	Number of relative digits in parameter estimation.
RelativeDigitsOuterCovariance	Number of relative digits in the covariance step.
PrecisionGoalInnerEstimation	Number of absolute digits in the inner optimization.
RelativeDigitsInnerEstimation	Number of relative digits in the inner optimization.
PrecisionGoalInnerCovariance	Number of absolute digits in the inner optimization during the covariance step.
RelativeDigitsInnerCovariance	Number of relative digits in the inner optimization during the covariance step.

In[•]:= **modelFit["ModelSummary"]**

Out[•]=

Model summary			
Estimation successful	True		
Covariance step successful	True		
Number of subjects	20		
Number of observations	240		
LogLikelihood (LL)	189.014		
Objective Function (−2*LL)	−378.027		
AIC	−368.027		
BIC	−363.048		
Condition number	6.38193		
Fixed effects	**Estimate**	**Standard error**	**RSE [%]**
ka	0.936901	0.0420829	4.49172
CL	10.4122	0.625062	6.00319
V	49.4082	1.2268	2.483
Ω matrix	**Ω standard error**	**Ω RSE (%)**	
(0.0667926)	(0.021578)	(32.306)	
Σ matrix			
(0.0476537 c[t]^2)			

3.5 Model Validation

The model object can be used by several other functions with the purpose of model validation and goodness-of-fit evaluation.

Goodness-of-fit plots

To obtain a collection of basic goodness-of-fit plots, as shown in Figure 2, we make use of the function call

In[•]:= **GoodnessOfFitAnalysis[modelFit]**

The obtained figure consists of four different panels: 1) population predictions versus observations (where the random effects are set to zero), 2) individual predictions versus observations, 3) individual weighted residuals versus time, and 4) individual weighted residuals versus individual predictions.

Table 2: FittedNLMEModel object properties

Property	**Description**
"ModelSummary"	A summary table of the estimated model.
"NumberOfSubjects"	The total number of subjects in the datasets.
"NumberOfObservations"	The total number of observations in the datasets.
"ObjectiveFunctionValue"	The value of the objective function ($-2 * L_L$) at the optimum.
"AIC"	The Akaike information criterion.
"BIC"	The Bayesian information criterion.
"LogLikelihoodValue"	The value of the log-likelihood (L_L) at the optimum.
"ConditionNumber"	The condition number of the parameter correlation matrix.
"FixedEffects"	The estimated fixed effects.
"OmegaMatrix"	The estimated Omega matrix.
"SigmaMatrix"	The estimated Sigma matrix.
"EBEs"	The empirical Bayes estimates (EBEs) for all individuals.
"SuccessFlagEstimation"	True if parameter estimation succeeded, False otherwise.
"SuccessFlagCovariance"	True if the covariance step succeeded, False otherwise.
"EstimationTime"	The time in seconds for the estimation problem.
"CovarianceTime"	The time in seconds for the covariance step.
"Data"	The datasets.
"Models"	The model equations.
"IndependentVariable"	The independent variable in each model.
"BestFit"	All estimated parameters.
"ParameterCovarianceMatrix"	The covariance matrix of the parameter estimates at the optimum.
"ParameterCorrelationMatrix"	The correlation matrix of the parameter estimates at the optimum.
"RandomEffectsParameters"	The symbolic random effects parameters.
"PhiParameters"	The symbolic phi parameters.
"StateVariables"	The state variables in each model.
"WhiteNoiseVariables"	The variables corresponding to Gaussian white noise variables (empty list for ODE models).
"StartValues"	The start values for all parameters in the estimation problem.

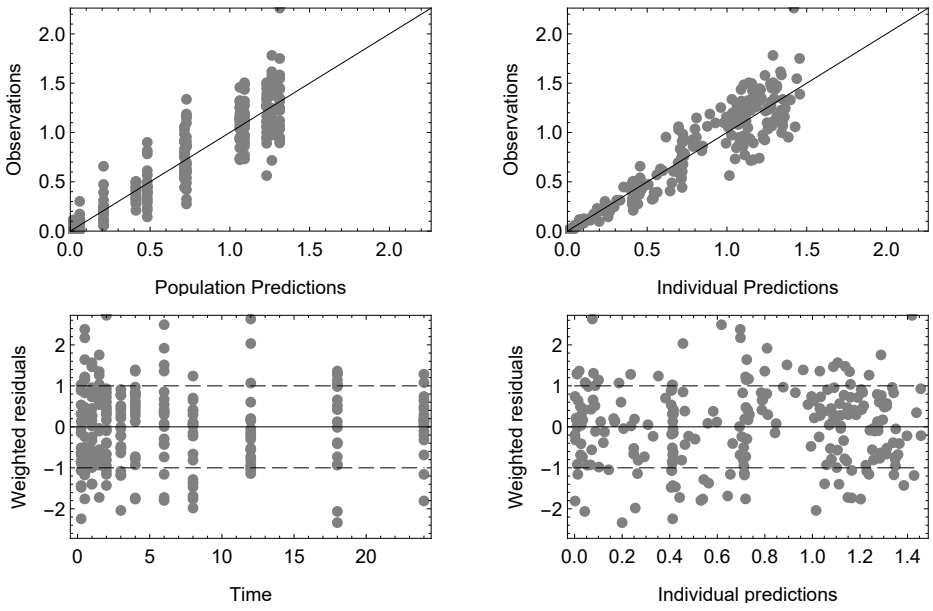


Figure 2: Basic goodness-of-fit plots for the model in Example 1.

Visual predictive checks

To generate a VPC based on 200 simulated datasets showing the 10th percentile, median, and 90th percentile with 90% confidence intervals, we call

```
In[•]:= VisualPredictiveCheck[modelFit, 200, Quantiles → {0.1, 0.5, 0.9}, ConfidenceInterval → 90];
```

yielding the VPC plot shown in Figure 3 (prediction-correction is set to False by default). The black solid lines show the calculated quantiles from observations, the grey solid lines show corresponding quantiles from the model simulation (with shaded confidence intervals), and the black dots show the observed data. As seen in VPC plot, we have good agreement between the observations and model predictions which is to be expected in this simulated example.

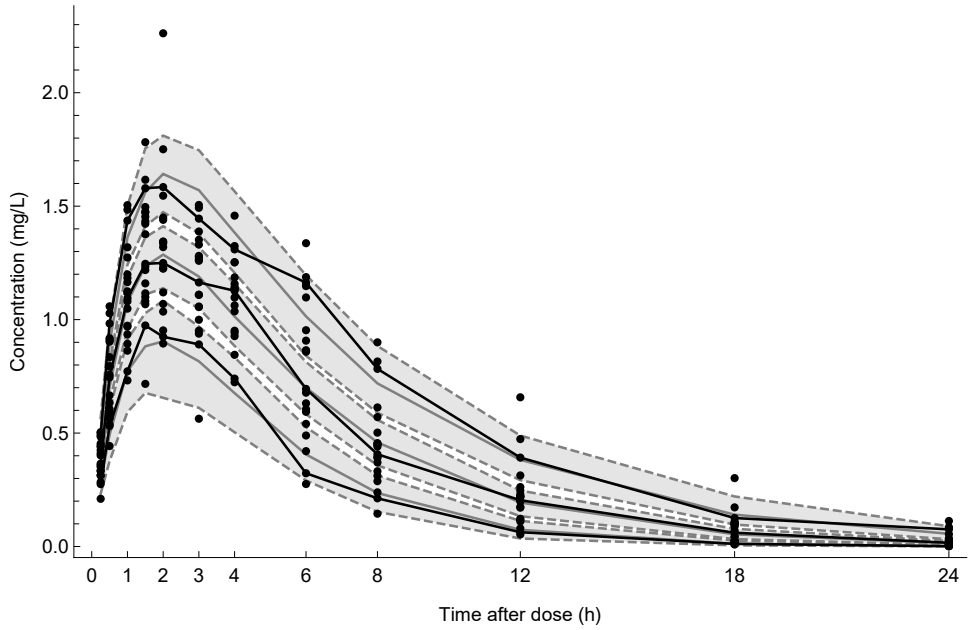


Figure 3: A VPC plot for the observations for the estimation model in Example 1.

4 Advanced Examples

In this section, we extend the introductory example to allow for multiple observations as well as stochastic dynamics. We will also consider the use of so called phi-parametrization of model parameters. In contrast to the introductory example, where the combination of clearance and random effect was written in the ordinary differential equations, composite phi-parameters containing both fixed and random effects can be defined. By utilizing the composite structure of fixed and random effects, the calculation of likelihood and gradient can be further improved in terms of speed. For the exact derivation of the phi-sensitivities we refer the reader to Appendix 2.

4.1 Example 2: Multiple Observation Variables

Here, we give an example of a model with two observed variables. This often occurs in pharmacokinetic-pharmacodynamic (PKPD) applications where both the drug concentration and the drug effect are measured during an experiment. We use the same PK model as in the introductory example, but now with an additional variable describing a drug response dependent on the drug concentration. The system

of differential equations is given by

$$\frac{dA_1}{dt} = -k_a A_1(t),\ A_1(0) = Dose \tag{19}$$

$$\frac{dA_2}{dt} = k_a A_1(t) - \frac{CL_{ind}}{V} A_2(t),\ A_2(0) = 0 \tag{20}$$

$$\frac{dR}{dt} = k_{out}\left(E0_{ind}\left(1 - \frac{c(t)}{EC50 + c(t)}\right) - R(t)\right),\ R(0) = E0_{ind}. \tag{21}$$

In the equations above the third differential equation represents an indirect response where the drug concentration $c(t) = A_2(t)/V$ inhibits the production term. The random effects $\boldsymbol{\eta} = (\eta_1, \eta_2) \sim N(0, \boldsymbol{\Omega})$ are assumed to be uncorrelated with covariance matrix

$$\boldsymbol{\Omega} = \begin{pmatrix} \omega_1^2 & 0 \\ 0 & \omega_2^2 \end{pmatrix} \tag{22}$$

yielding log-normally distributed clearance and baseline response

$$CL_{ind} = CL\exp(\eta_1) \tag{23}$$

$$E0_{ind} = E0\exp(\eta_2) \tag{24}$$

Compared to the introductory example we now have two types of observation variables and a slightly more complex error model: a combined error model for the PK observations and an additive error model for the PD observations. Hence, observations are assumed to be taken according to

$$\boldsymbol{y}(t) = (c(t), R(t)) + \boldsymbol{e}_t,\ \boldsymbol{e}_t \sim N(0, \boldsymbol{\Sigma}) \tag{25}$$

where the observation error covariance matrix $\boldsymbol{\Sigma}$ is given by

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{add1}^2 + (\sigma_{prop} c(t))^2 & 0 \\ 0 & \sigma_{add2}^2 \end{pmatrix}. \tag{26}$$

Observations of the concentration in the central compartment (unit mg/L) and of the response R(t) are taken at 0.25, 0.5, 1, 1.5, 2, 3, 4, 6, 8, 12, 18, and 24 hours after dose. In this example, we consider multiple doses (100 mg, 300 mg, and 1000 mg) with 15 subjects in each dose group. Mathematica code for simulating the dataset is available in Appendix 1. A plot of the simulated data is shown in Figure 4.

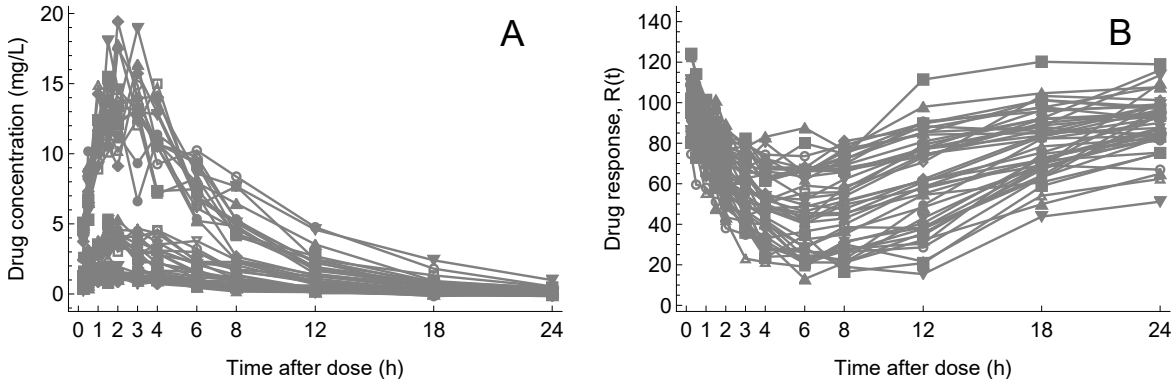


Figure 4: Simulated data from the PKPD experiment in Example 2. Panel A shows the drug concentration and panel B shows the drug response.

To estimate the model parameters we express the system model using an additional differential equation for the turnover model compared to the previous example. Moreover, we use direct coding of an auxiliary expression for the concentration relationship and use phi-parametrization for the inter-individual variability in clearance and baseline response. Since we measure both the concentration of the drug and the drug response, the observation is a vector consisting of two entities.

```
sys = {
      A1'[t] == -ka * A1[t],
      A2'[t] == ka * A1[t] - phi1/V * A2[t],
      R'[t] == kout * (phi2 * (1 - 1 * c[t] / (EC50 + c[t])) - R[t]),
      A1[0] == Dose,
      A2[0] == 0,
      R[0] == phi2,
      phi1 == CL * Exp[η1],
      phi2 == E0 * Exp[η2],
      c[t] == A2[t] / V
     };
obs = {c[t], R[t]};
```

In addition to the system definition, the user may also provide a user-defined structure of the random effects covariance matrix, $\mathbf{\Omega}$, and the observation error covariance matrix, $\mathbf{\Sigma}$. In this example, we consider a $\mathbf{\Sigma}$ matrix that describes a combined error model for the PK observations and an additive error model for the PD observations, parametrized by three parameters as follows.

```
SigmaMatrix = ( add1^2 + (prop1 * c[t])^2    0
                0                            add2^2 );
```

To estimate the model parameters, we need the model definition, a list of the fixed effects parameters (k_a, CL, V, k_{out}, $E0$, and $EC50$) together with their corresponding initial values, and a list of the random effects parameters (η_1 and η_2). Additionally, in this example, we make use of the advanced option for the $\mathbf{\Sigma}$ matrix and provide the user-defined symbolic $\mathbf{\Sigma}$ matrix together with the related initial values. For the covariance matrix of the random effects, $\mathbf{\Omega}$, we here choose to estimate the full covariance matrix, specified by the option value "Full".

```
modelFit = NLMEDynamicalModelFit[data, {sys, obs},
      {{ka, 0.8}, {CL, 20}, {V, 40}, {kout, 0.2}, {E0, 90}, {EC50, 5}}, {η1, η2},
      Sigma → {"Advanced", {{SigmaMatrix}, {{add1, 0.1}, {prop1, 0.1}, {add2, 6}}}},
      Omega → "Full"]
```

```
FittedNLMEModel[  State variables: 3
                                                                   Observables: 2 ]
```

The summary for the estimated PKPD model is obtained using the "ModelSummary" property.

In[•]:= `modelFit["ModelSummary"]`

Out[•]=

Model summary			
Estimation successful	True		
Covariance step successful	True		
Number of subjects	45		
Number of observations	1080		
LogLikelihood (LL)	−2021.38		
Objective Function (−2*LL)	4042.75		
AIC	4066.75		
BIC	4088.43		
Condition number	10.724		
Fixed effects	**Estimate**	**Standard error**	**RSE [%]**
ka	0.980047	0.0328382	3.35067
CL	9.96922	0.356708	3.5781
V	49.0419	0.949498	1.9361
kout	0.404746	0.00789907	1.95161
E0	98.8206	1.67517	1.69516
EC50	2.00386	0.0546019	2.72483
Ω matrix	**Ω standard error**	**Ω RSE (%)**	
$\begin{pmatrix} 0.0517205 & -0.00349516 \\ -0.00349516 & 0.0119313 \end{pmatrix}$	$\begin{pmatrix} 0.0118723 & 0.00393226 \\ 0.00393226 & 0.00260447 \end{pmatrix}$	$\begin{pmatrix} 22.9547 & 112.506 \\ 112.506 & 21.8288 \end{pmatrix}$	
Σ matrix			
$\begin{pmatrix} 0.0101352 + 0.0347157\, c[t]^2 & 0 \\ 0 & 24.0272 \end{pmatrix}$			

The goodness-of-fit functionality extends seamlessly to multiple observation variables. When calling the GoodnessOfFitAnalysis function, a plot is created for each observation variable. The GOF plots for the PK and PD observations are shown in Figure 5 and 6.

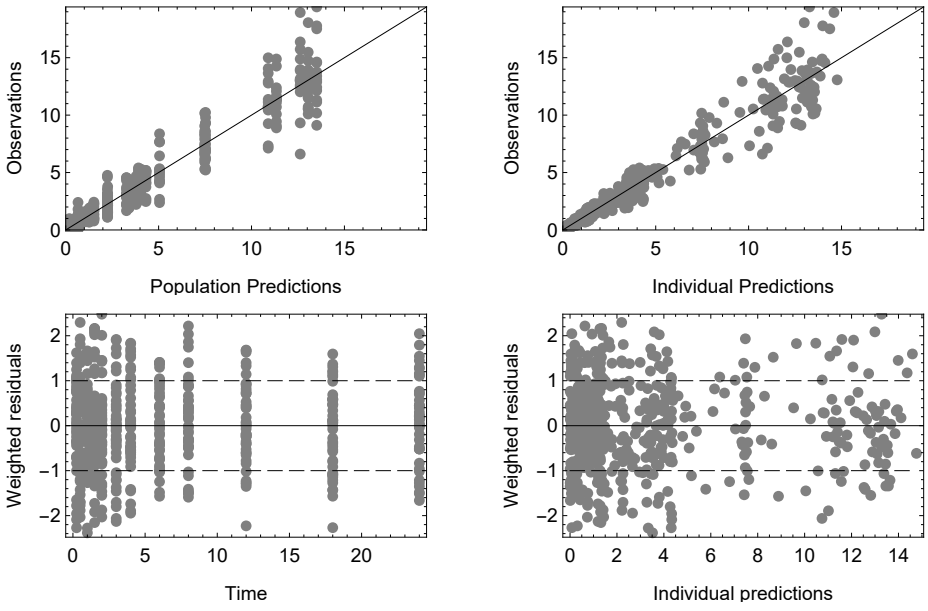


Figure 5: Standard goodness-of-fit plots for the PK observations in Example 2.

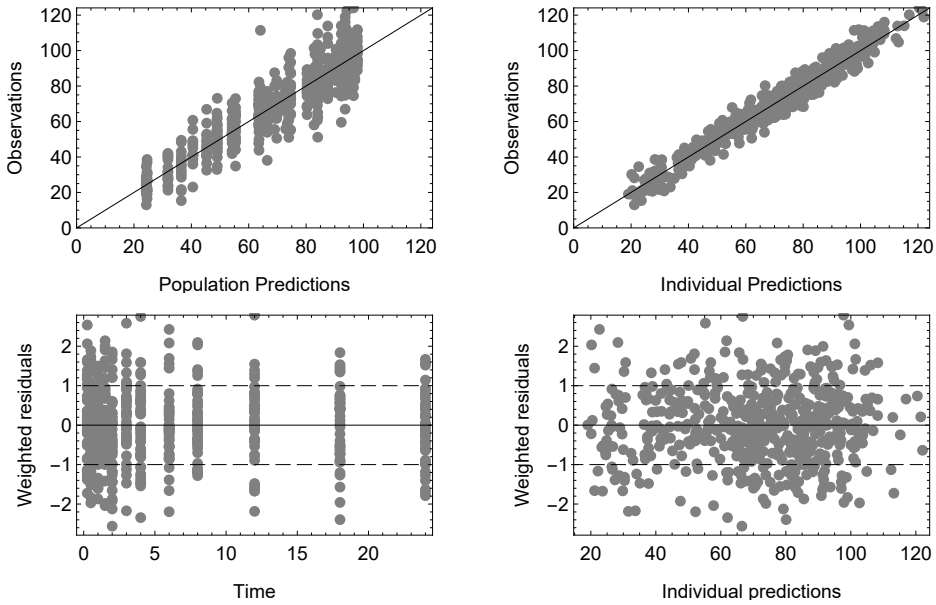
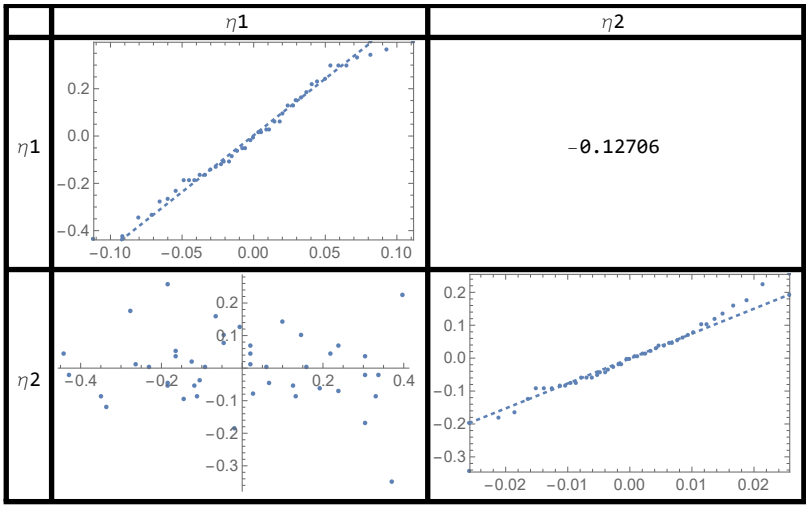


Figure 6: Standard goodness-of-fit plots for the PD observations in Example 2.

In addition to the goodness-of-fit plots, we now turn the attention to assessment of the model assumption regarding the distribution of the random effects parameters. To validate the assumption of normality, we can plot the distribution and correlation of the Empirical Bayes Estimates (EBEs). In NLMEModeling this is achieved using the function call

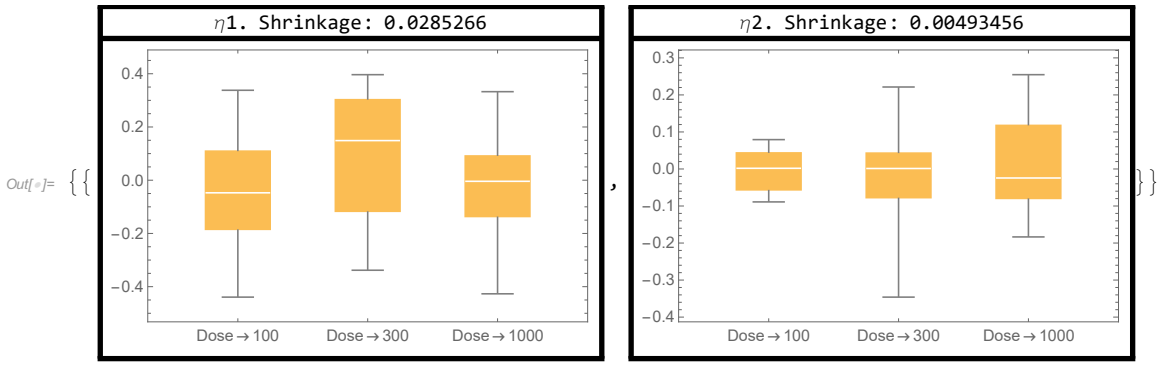
```
In[•]:= EBECorrelationAnalysis[modelFit]
```

Out[•]=



where the diagonal shows the quantile-quantile plot for each random effect, the sub-diagonal plots show the pairs-plot, and the above-diagonal plots show the Pearson correlation coefficient between the EBEs. To check the distribution of the EBEs versus a specific discrete covariate of interest, we use the function EBEBoxWhiskerChart which takes the model object as argument and an option for potential stratification. The function will also report the shrinkage of each random effect parameter.

```
In[•]:= EBEBoxWhiskerChart[modelFit, Stratify → "Dose"]
```



The VPC analysis extends seamlessly to multiple observation variables, with a VPC for each observation variable. Here, we create a prediction-corrected VPC plot for both the PK and PD observations (using 10th, 50th and 90th percentiles with 90% confidence interval) based on 200 simulated datasets. The resulting VPC plots are shown in Figure 7.

```
In[•]:= VisualPredictiveCheck[modelFit, 200,
          Quantiles → {0.1, 0.5, 0.9},
          ConfidenceInterval → 90,
          PredictionCorrection → True
        ];
```

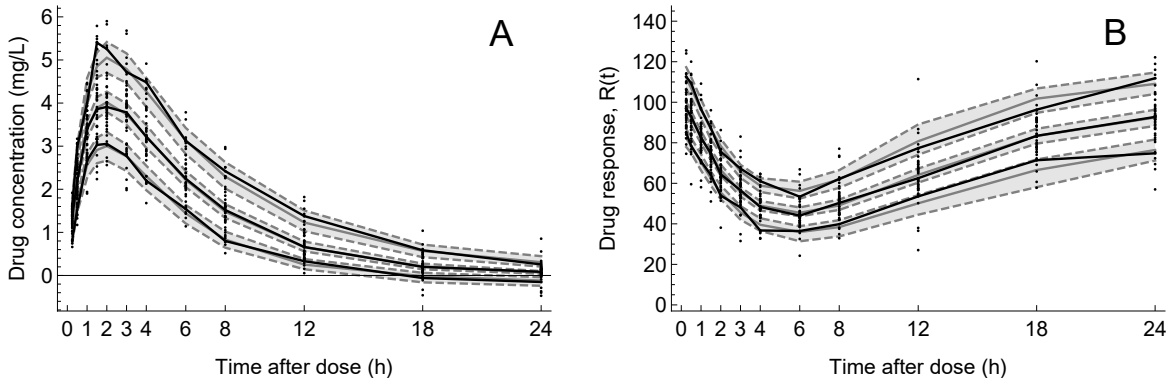


Figure 7: Prediction-corrected visual predictive checks for the PKPD model in Example 2, stratified on observation variable. Panel A shows the PK observations and panel B shows the PD observations.

4.2 Example 3: Extension to Stochastic Dynamics

In this example, we illustrate how the deterministic ODE model can easily be extended to incorporate stochastic differential equations. We consider the same model as in Example 2, but replace the differential equation for the drug effect $R(t)$ with a stochastic differential equation,

$$\frac{dA_1}{dt} = -k_a A_1(t),\ A_1(0) = Dose \tag{27}$$

$$\frac{dA_2}{dt} = k_a A_1(t) - \frac{CL_{ind}}{V} A_2(t),\ A_2(0) = 0 \tag{28}$$

$$dR = k_{out}(E0_{ind}(1 - \frac{c(t)}{EC50 + c(t)}) - R(t))dt + g \cdot dW(t),\ R(0) = E0_{ind} \tag{29}$$

where the SDE is written on differential form with $dW(t)$ being the increment of a standard Wiener process. The parameter g is of special interest in this example since it quantifies the influence of the stochastic term on the dynamics. The random effects covariance matrix $\mathbf{\Omega}$, the observation error covariance matrix $\mathbf{\Sigma}$, and experimental design is kept the same as in Example 2. Mathematica code for simulating the dataset is available in Appendix 1. A plot of the simulated data is shown in Figure 8. Note the increased variability in the PD observations compared to Example 2, due to the stochastic nature of the underlying model.

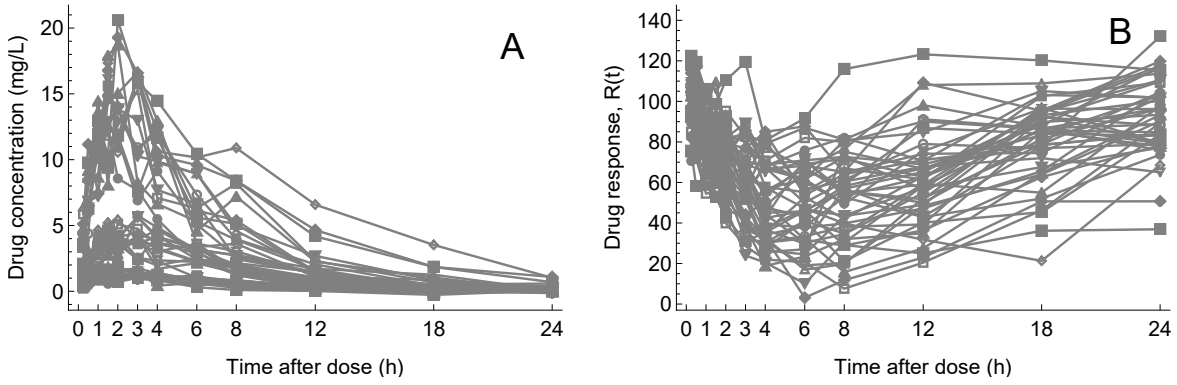


Figure 8: Simulated data from a PKPD experiment with a stochastic PD model in Example 3. Panel A shows the drug concentration and panel B shows the drug response.

To define the SDE model in NLMEModeling, the only thing we need to add is the stochastic component to the model definition. To align with the ODE notation, we here write the stochastic component as the generalized derivative of the standard Wiener process, which we will refer to as Gaussian white noise. The user can use any symbol for the Gaussian white noise (here denoted $w(t)$) and it can scale with any parameter, here multiplied with the parameter g.

```
In[•]:= sys = {
          A1'[t] == -ka * A1[t],
          A2'[t] == ka * A1[t] - phi1/V * A2[t],
          R'[t] == kout * (phi2 * (1 - 1 * c[t] / (EC50 + c[t])) - R[t]) + g * w[t],
          A1[0] == Dose,
          A2[0] == 0,
          R[0] == phi2,
          phi1 == CL * Exp[η1],
          phi2 == E0 * Exp[η2],
          c[t] == A2[t] / V
         };
        obs = {c[t], R[t]};
```

To estimate the model parameters we make a similar call as for the ODE case but with an additional start value for the parameter g, as well as a list of the symbols used to represent the white noise variables. In this example, we use a diagonal random effects covariance matrix $\mathbf{\Omega}$.

```
In[•]:= modelFit = NLMEDynamicalModelFit[data, {sys, obs},
          {{ka, 0.8}, {CL, 20}, {V, 40}, {kout, 0.2}, {E0, 90}, {EC50, 5}, {g, 5}}, {η1, η2}, w,
          Sigma → {"Advanced", {{SigmaMatrix}, {{add1, 0.1}, {prop1, 0.1}, {add2, 6}}}},
          Omega → "Diagonal"]

Out[•]= FittedNLMEModel[ State variables: 3
                         Observables: 2 ]
```

The model summary for the estimated model is obtained as before, using the "ModelSummary" property.

Notice the additional fixed effect parameter g in the table, which is estimated with good precision (the value used for simulation was $g = 10$, see Appendix 1).

```
In[•]:= modelFit["ModelSummary"]
```

Out[•]=

Model summary			
Estimation successful	True		
Covariance step successful	True		
Number of subjects	45		
Number of observations	1080		
LogLikelihood (LL)	−2413.61		
Objective Function (−2*LL)	4827.23		
AIC	4851.23		
BIC	4872.91		
Condition number	9.47035		
Fixed effects	**Estimate**	**Standard error**	**RSE [%]**
ka	1.04653	0.0410174	3.91936
CL	9.67091	0.433053	4.47789
V	50.4931	1.11361	2.20546
kout	0.373084	0.0182335	4.88723
E0	101.314	1.85596	1.83189
EC50	1.98379	0.119739	6.03585
g	8.34384	0.67933	8.1417
Ω matrix	**Ω standard error**	**Ω RSE (%)**	
$\begin{pmatrix} 0.0823331 & 0 \\ 0 & 0.0115488 \end{pmatrix}$	$\begin{pmatrix} 0.0187053 & 0. \\ 0. & 0.00297464 \end{pmatrix}$	$\begin{pmatrix} 22.719 & 0. \\ 0. & 25.7571 \end{pmatrix}$	
Σ matrix			
$\begin{pmatrix} 0.0110889 + 0.0462735\, c[t]^2 & 0 \\ 0 & 36.3744 \end{pmatrix}$			

The model object and functionality for stochastic models are designed to be the same as for the ODE models. As an example, we here show a visual predictive check for the stochastic mixed effects model. For each dose group, we create a VPC using the 50th percentile with 90% confidence intervals (no prediction-correction). The resulting VPCs are shown in Figure 9.

```
In[•]:= Table[VisualPredictiveCheck[modelFit, 200,
          Quantiles → {0.5},
          ConfidenceInterval → 90,
          PredictionCorrection → False,
          Stratification → {True, {Dose → dosei}}
         ], {dosei, {100, 300, 1000}}];
```

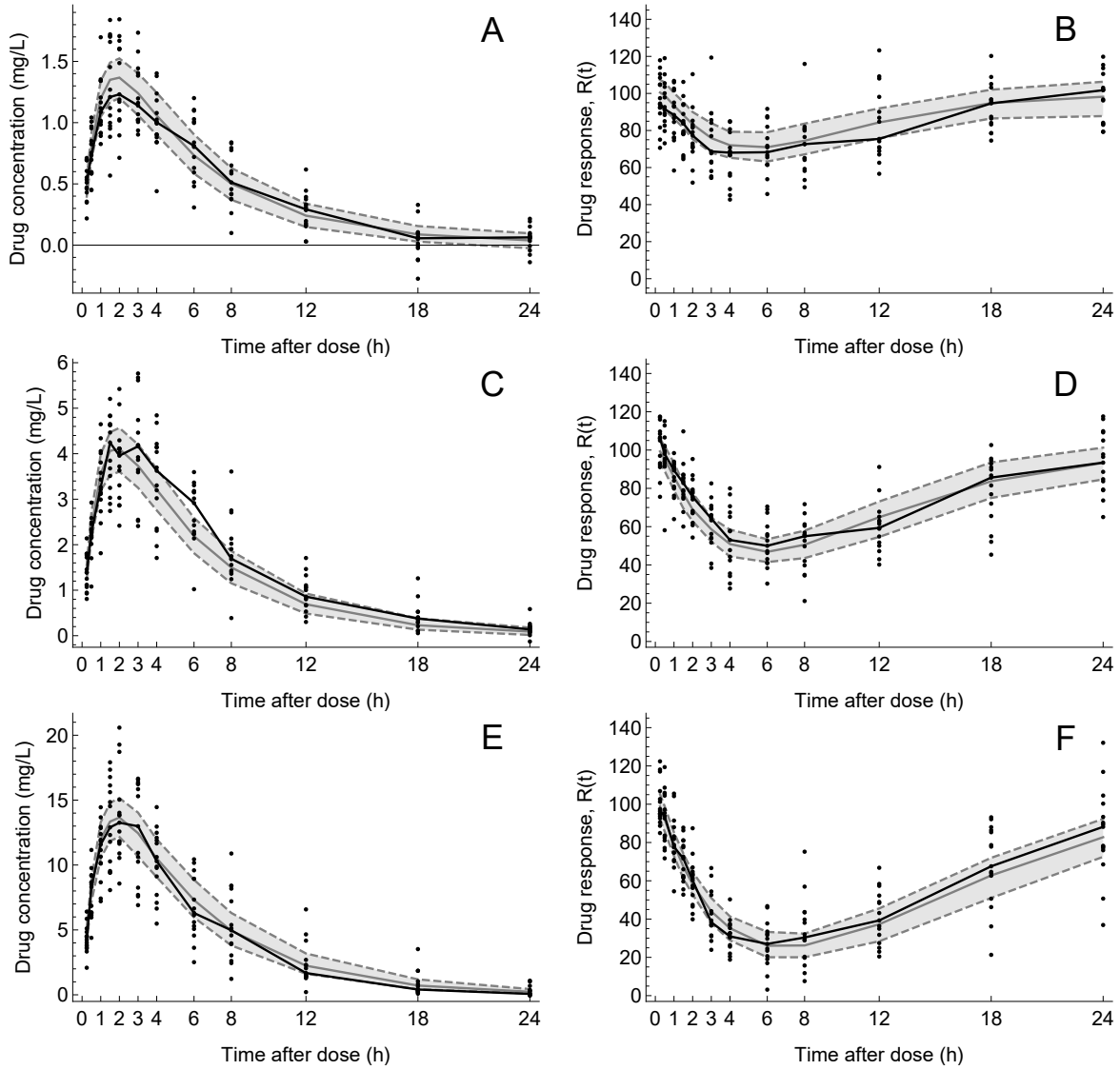


Figure 9: Visual predictive checks (VPCs) for the stochastic PKPD model in Example 3. Panel A: Dose 300 mg, PK observations. Panel B: Dose 300 mg, PD observations. Panel C: Dose 600 mg, PK observations. Panel D: Dose 600 mg, PD observations. Panel E: Dose 1000 mg, PK observations. Panel F: Dose 1000 mg, PD observations.

5 Discussion

In this paper, we have presented the Wolfram Mathematica package NLMEModeling for performing nonlinear mixed effects modeling of dynamical systems. NLMEModeling offers a user-friendly environment for both simulation and estimation of nonlinear mixed effects models, and the package also comes with functionality for performing visual predictive checks and standard goodness-of-fit plots. Furthermore, the model object returned by the estimation procedure in NLMEModeling contains all the necessary information for doing additional analysis.

NLMEModeling provides an easy-to-use modeling framework integrated into Wolfram Mathematica as an add-on package. Wolfram Mathematica has a large library of statistical models, simulation, and visualization capabilities. By providing a mixed-effects modeling framework that is tightly linked to the Wolfram Language, the user is free to integrate other types of analyses around the mixed-effects framework presented in this paper.

In addition to the commonly used ordinary differential equations, NLMEModeling also supports stochastic differential equations mixed effects models (SDEMEMs). As discussed in the recent review paper

by Irurzun-Arana *et al.* [39], incorporating stochastic behavior in mixed-effects models is a promising way of capturing three sources of variability: inter-individual, intra-individual, and system stochasticity. Moreover, as the syntax and functionality for ODE and SDE models are closely linked in the package, the extension to stochastic models is seamless. To the authors best knowledge this is also the first work that provides functionality for visual predictive checks for SDEMEMs. In the current version of NLMEModeling predicted-corrected VPCs are not supported for SDEMEMs, but it is an interesting extension to the approach presented in this work.

In terms of parameter estimation methods, NLMEModeling use the first-order conditional estimation method for approximation of the likelihood. In the case of stochastic models, the extended Kalman filter (EKF) is used for estimating the underlying state of the system conditional on the observations. The EKF has previously successfully been combined with both the FOCE method [1, 15] and the stochastic approximation expectation-maximization method [5]. In addition to earlier work, NLMEModeling utilizes the exact gradient method where the gradient of the objective function is calculated using the sensitivity equations of the underlying system, leading to a faster and more precise optimization routine, [26, 27].

In terms of limitations, the current release of NLMEModeling estimates the model parameters using an approximate likelihood approach. It might be of interest to consider a Bayesian setting, to enable inclusion of prior information and assessment of the posterior distribution. Other types of sampling-based methods have been proposed and might be a valuable addition to the current functionality [40]. In addition to the limitations discussed above, the dataset structure in the current release of NLMEModeling is rather simple. An extension to a more hierarchical structure (e.g. the Dataset object in Wolfram Mathematica) is currently under consideration as well as automatic import and conversion of NONMEM and Monolix datasets.

The different examples presented in this paper are meant to serve as an overview of the model-building tools that NLMEModeling provides and the types of models it can process. For additional details regarding function options, methods, and additional examples, we refer the reader to the package documentation in Wolfram Mathematica. One interesting application of SDEMEMs not covered in this paper is to model stochastic behavior in the model parameters, for example the absorption rate in a PK model as presented in the paper by Matzuka *et al.* [6]. In NLMEModeling, this can easily be achieved by introducing a stochastic differential equation to describe the evolution of a specific model parameter of interest.

To conclude, NLMEModeling provides a user-friendly package for performing mixed effects modeling for dynamical systems in Wolfram Mathematica. The development of NLMEModeling is currently active and the package is available on request from `http://www.fcc.chalmers.se/software/other-software/nlmemodeling`.

Acknowledgements The authors would like to acknowledge the contribution to the development of the NLMEModeling package that has been done at Fraunhofer-Chalmers Centre for Industrial Mathematics. The authors would especially like to thank Helga Kristín Ólafsdóttir, Simon Berglund Watanabe, and Felix Held. This work has been partly founded by the Swedish Foundation for Strategic Research by the project Hierarchical Mixed Effects Modeling of Dynamical Systems (Grant no. AM13-0046).

References

[1] Overgaard, R. V., Jonsson, N., Tornøe, C. W. & Madsen, H. Non-linear mixed-effects models with stochastic differential equations: Implementation of an estimation algorithm. *Journal of Pharmacokinetics and Pharmacodynamics* **32**, 85–107 (2005).

[2] Mortensen, S. B. *et al.* A matlab framework for estimation of NLME models using stochastic differential equations: Applications for estimation of insulin secretion rates. *Journal of Pharmacokinetics and Pharmacodynamics* **34**, 623–642 (2007).

[3] Picchini, U., Gaetano, A. D. & Ditlevsen, S. Stochastic differential mixed-effects models. *Scandinavian Journal of Statistics* (2010).

[4] Picchini, U. & Ditlevsen, S. Practical estimation of high dimensional stochastic differential mixed-effects models. *Computational Statistics and Data Analysis* **55**, 1426–1444 (2011).

[5] Delattre, M. & Lavielle, M. Coupling the SAEM algorithm and the extended Kalman filter for maximum likelihood estimation in mixed-effects diffusion models. *Statistics and its Interface* **6**, 519–532 (2013).

[6] Matzuka, B., Chittenden, J., Monteleone, J. & Tran, H. Stochastic nonlinear mixed effects: a metformin case study. *Journal of Pharmacokinetics and Pharmacodynamics* **43**, 85–98 (2016).

[7] Bauer, R. J. NONMEM Tutorial Part I: Description of Commands and Options, With Simple Examples of Population Analysis. *CPT: Pharmacometrics and Systems Pharmacology* **8**, 525–537 (2019).

[8] Bauer, R. J. NONMEM Tutorial Part II: Estimation Methods and Advanced Examples. *CPT: Pharmacometrics and Systems Pharmacology* **8**, 538–556 (2019).

[9] Beal, S. L., Sheiner, L. B., Boeckmann, A. & Bauer, R. NONMEM 7.4 User's Guides (2017).

[10] Lixoft SAS. Monolix version 2019R2 (2019).

[11] Fidler, M. *et al.* Nonlinear Mixed-Effects Model Development and Simulation Using nlmixr and Related R Open-Source Packages. *CPT: Pharmacometrics and Systems Pharmacology* **8**, 621–633 (2019).

[12] Schoemaker, R. *et al.* Performance of the SAEM and FOCEI Algorithms in the Open-Source, Nonlinear Mixed Effect Modeling Tool nlmixr. *CPT: Pharmacometrics and Systems Pharmacology* **8**, 923–930 (2019).

[13] Carpenter, B. *et al.* Stan: A probabilistic programming language. *Journal of Statistical Software* **76** (2017).

[14] Klim, S., Mortensen, S. B., Kristensen, N. R., Overgaard, R. V. & Madsen, H. Population stochastic modelling (PSM)-An R package for mixed-effects models based on stochastic differential equations. *Computer Methods and Programs in Biomedicine* (2009).

[15] Tornøe, C. W. *et al.* Stochastic differential equations in NONMEM®: Implementation, application, and comparison with ordinary differential equations. *Pharmaceutical Research* **22**, 1247–1258 (2005).

[16] Dion, C., Hermann, S. & Samson, A. Mixedsde: A package to fit mixed stochastic differential equations. *R Journal* **11**, 1–22 (2019).

[17] Wolfram Research, Inc. Mathematica 12.1 (2020).

[18] Cardilin, T. *et al.* Tumor Static Concentration Curves in Combination Therapy. *AAPS Journal* **19**, 456–467 (2017).

[19] Cardilin, T. *et al.* Model-Based Evaluation of Radiation and Radiosensitizing Agents in Oncology. *CPT: Pharmacometrics and Systems Pharmacology* **7**, 51–58 (2018).

[20] Cardilin, T. *et al.* Modeling long-term tumor growth and kill after combinations of radiation and radiosensitizing agents. *Cancer Chemotherapy and Pharmacology* **83**, 1159–1173 (2019).

[21] Almquist, J. *et al.* A nonlinear mixed effects approach for modeling the cell-to-cell variability of Mig1 dynamics in yeast. *PLoS ONE* **10**, 1–32 (2015).

[22] Tapani, S. *et al.* Joint feedback analysis modeling of nonesterified fatty acids in obese zucker rats and normal sprague-dawley rats after different routes of administration of nicotinic acid. *Journal of Pharmaceutical Sciences* **103**, 2571–2584 (2014).

[23] Leander, J., Almquist, J., Ahlström, C., Gabrielsson, J. & Jirstrand, M. Mixed Effects Modeling Using Stochastic Differential Equations: Illustrated by Pharmacokinetic Data of Nicotinic Acid in Obese Zucker Rats. *AAPS Journal* **17**, 586–596 (2015).

[24] Andersson, R. *et al.* Dose-response-time modelling: Second-generation turnover model with integral feedback control. *European Journal of Pharmaceutical Sciences* **81**, 189–200 (2016).

[25] Andersson, R. *et al.* Modeling of free fatty acid dynamics: insulin and nicotinic acid resistance under acute and chronic treatments. *Journal of Pharmacokinetics and Pharmacodynamics* **44**, 203–222 (2017).

[26] Almquist, J., Leander, J. & Jirstrand, M. Using sensitivity equations for computing gradients of the FOCE and FOCEI approximations to the population likelihood. *Journal of Pharmacokinetics and Pharmacodynamics* **42**, 191–209 (2015).

[27] Olafsdottir, H. K., Leander, J., Almquist, J. & Jirstrand, M. Exact Gradients Improve Parameter Estimation in Nonlinear Mixed Effects Models with Stochastic Dynamics. *AAPS Journal* **20**, 1–13 (2018).

[28] Wang, Y. Derivation of various NONMEM estimation methods. *Journal of Pharmacokinetics and Pharmacodynamics* **34**, 575–593 (2007).

[29] Jazwinsky, A. H. *Stochastic Processes and Filtering Theory* (Academic Press, 1970).

[30] Leander, J., Lundh, T. & Jirstrand, M. Stochastic differential equations as a tool to regularize the parameter estimation problem for continuous time dynamical systems given discrete time measurements. *Mathematical Biosciences* **251**, 54–62 (2014).

[31] Vonesh, E. F. A note on the use of Laplace's approximation for nonlinear mixed-effects models. *Biometrika* (1996).

[32] Nocedal, J. & Wright, S. *Numerical Optimization* (Springer-Verlag, New York, 2006).

[33] Kloeden, P. E. & Platen, E. *Numerical Solution of Stochastic Differential Equations* (Springer, 1992).

[34] Nguyen, T. H. *et al.* Model evaluation of continuous data pharmacometric models: Metrics and graphics. *CPT: Pharmacometrics and Systems Pharmacology* **6**, 87–109 (2017).

[35] Bergstrand, M., Hooker, A. C., Wallin, J. E. & Karlsson, M. O. Prediction-corrected visual predictive checks for diagnosing nonlinear mixed-effects models. *AAPS Journal* **13**, 143–151 (2011).

[36] Aoki, Y., Nordgren, R. & Hooker, A. C. Preconditioning of Nonlinear Mixed Effects Models for Stabilisation of Variance-Covariance Matrix Computations. *AAPS Journal* **18**, 505–518 (2016).

[37] Akaike, H. A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control* **19**, 716–723 (1974).

[38] Schwarz, G. Estimating the Dimension of a Model. *The Annals of Statistics* **6**, 461–464 (1978).

[39] Irurzun-Arana, I., Rackauckas, C., McDonald, T. O. & Trocóniz, I. F. Beyond Deterministic Models in Drug Discovery and Development. *Trends in Pharmacological Sciences* 1–14 (2020).

[40] Donnet, S. & Samson, A. Using PMCMC in EM algorithm for stochastic mixed models: theoretical and practical issues. *Journal de la Société Française de Statistique* **155**, 49–72–72 (2014).

Appendix 1: Simulation Code

Example 1

```
In[•]:= (*Define ODE and observation model*)
sys = {
    A1'[t] == -ka * A1[t],
    A2'[t] == ka * A1[t] - CL * Exp[η1] / V * A2[t],
    A1[0] == Dose,
    A2[0] == 0,
    c[t] == A2[t] / V
   };
obs = {c[t]};

(*Define Σ and Ω matrices*)
realΣ = {{(0.2 * c[t]) ^2}};
realΩ = {{0.3^2}};

(*Define number of subjects, sampling times and covariates*)
SeedRandom[31];
nInd = 20;
timePoints = {0.25, 0.5, 1, 1.5, 2, 3, 4, 6, 8, 12, 18, 24};
dataStructure = Table[{timePoints, {SubjectID → ToString[i], Dose → 100,
      Weight → Round[RandomVariate[NormalDistribution[70, 5]]]}}, {i, 1, nInd}];

(*Generate data*)
SeedRandom[42];
{randomParamValuesData, data} = NLMEDynamicalModelSimulate[dataStructure, {sys, obs},
   {{ka, 1}, {CL, 10}, {V, 50}}, {η1}, Sigma → realΣ, Omega → realΩ];

(*Export dataset*)
Export[NotebookDirectory[] <> "Example1-data.m", data];
```

Example 2

```
In[•]:= (*Define ODE and observation model*)
sys = {
    A1'[t] == -ka * A1[t],
    A2'[t] == ka * A1[t] - phi1/V * A2[t],
    R'[t] == kout * (phi2 * (1 - 1 * c[t] / (EC50 + c[t])) - R[t]),
    A1[0] == Dose,
    A2[0] == 0,
    R[0] == phi2,
    phi1 == CL * Exp[η1],
    phi2 == E0 * Exp[η2],
    c[t] == A2[t] / V

  };
obs = {c[t], R[t]};

(*Define Σ and Ω matrices*)
realΣ = {{0.1^2 + (0.2 * c[t])^2, 0}, {0, 5^2}} ;
realΩ = {{0.3^2, 0}, {0, 0.1^2}};

(* Define and set parameters *)
randomParams = {η1, η2};
params = {ka, CL, V, kout, E0, EC50};

(*Define number of subjects, sampling times and covariates*)
SeedRandom[100];
nGroup = 15;
timePoints = {0.25, 0.5, 1, 1.5, 2, 3, 4, 6, 8, 12, 18, 24};
doseGroup1 = Table[{timePoints, {SubjectID → ToString[i], Dose → 100,
      Weight → Round[RandomVariate[NormalDistribution[70, 5]]]}}, {i, 1, nGroup}];
doseGroup2 = Table[{timePoints, {SubjectID → ToString[i], Dose → 300,
      Weight → Round[RandomVariate[NormalDistribution[70, 5]]]}}, {i, nGroup + 1, 2 * nGroup}];
doseGroup3 = Table[{timePoints, {SubjectID → ToString[i], Dose → 1000,
      Weight → Round[RandomVariate[NormalDistribution[70, 5]]]}}, {i, 2 * nGroup + 1, 3 * nGroup}];
dataStructure = Join[doseGroup1, doseGroup2, doseGroup3];

(*Generate data*)
SeedRandom[43];
{randomParamValuesData, data} = NLMEDynamicalModelSimulate[dataStructure, {sys, obs},
   {{ka, 1}, {CL, 10}, {V, 50}, {kout, 0.4}, {E0, 100}, {EC50, 2}},
   {η1, η2}, Sigma → realΣ, Omega → realΩ];

(*Export dataset*)
Export[NotebookDirectory[] <> "Example2-data.m", data];
```

Example 3

```
(*Define SDE and observation model*)
sys = {
    A1'[t] == -ka * A1[t],
    A2'[t] == ka * A1[t] - phi1/V * A2[t],
    R'[t] == kout * (phi2 * (1 - 1 * c[t] / (EC50 + c[t])) - R[t]) + g * w[t],
    A1[0] == Dose,
    A2[0] == 0,
    R[0] == phi2,
    phi1 == CL * Exp[η1],
    phi2 == E0 * Exp[η2],
    c[t] == A2[t] / V
   };
obs = {c[t], R[t]};

(*Define Σ and Ω matrices*)
realΣ = {{0.1^2 + (0.2 * c[t])^2, 0}, {0, 5^2}} ;
realΩ = {{0.3^2, 0}, {0, 0.1^2}};

(*Define number of subjects, sampling times and covariates*)
SeedRandom[101];
nGroup = 15;
timePoints = {0.25, 0.5, 1, 1.5, 2, 3, 4, 6, 8, 12, 18, 24};
doseGroup1 = Table[{timePoints, {SubjectID → ToString[i], Dose → 100,
      Weight → Round[RandomVariate[NormalDistribution[70, 5]]]}}, {i, 1, nGroup}];
doseGroup2 = Table[{timePoints, {SubjectID → ToString[i], Dose → 300,
      Weight → Round[RandomVariate[NormalDistribution[70, 5]]]}}, {i, nGroup + 1, 2 * nGroup}];
doseGroup3 = Table[{timePoints, {SubjectID → ToString[i], Dose → 1000,
      Weight → Round[RandomVariate[NormalDistribution[70, 5]]]}}, {i, 2 * nGroup + 1, 3 * nGroup}];
dataStructure = Join[doseGroup1, doseGroup2, doseGroup3];

(*Generate data*)
SeedRandom[44];
{randomParamValuesData, data} = NLMEDynamicalModelSimulate[dataStructure, {sys, obs},
   {{ka, 1}, {CL, 10}, {V, 50}, {kout, 0.4}, {E0, 100}, {EC50, 2}, {g, 10}},
   {η1, η2}, w, Sigma → realΣ, Omega → realΩ];

(*Export dataset*)
Export[NotebookDirectory[] <> "Example3-data.m", data]
```

Appendix 2: Derivation of Phi-sensitivites

The derivation presented here is an extension of the exact gradient method for ordinary differential equations models presented by Almquist et al [26], which is recommended as background material for a full appreciation of the derivation presented in this work.

In the general case, NLME models depends on a vector of fixed effect parameters, $\boldsymbol{\theta}$, and a vector of random effect parameters $\boldsymbol{\eta}$. As shown in Example 2 & 3, it might be more convenient to define a set of composite parameters, $\boldsymbol{\varphi}$, that can depend on both fixed and random effects, $\boldsymbol{\varphi} = \boldsymbol{\varphi}(\boldsymbol{\theta}, \boldsymbol{\eta})$. This φ-parameterization facilitates the definition of for instance normal or log-normal distributed parameters, but also allow for more complex definitions. For maximal flexibility, the NLMEModeling package allows model definitions using $\boldsymbol{\theta}$, $\boldsymbol{\eta}$, and $\boldsymbol{\varphi}$ simultaneously.

Not only does φ-parameterization allow a more intuitive parameterization of the models, but it also results in a significant simplification of the sensitivity equations that are required for solving the parameter estimation problem. For example, for a normally distributed parameter $\varphi = \theta + \eta$, it is easily seen that the sensitivities of model state variables with respect to φ, θ, and η are identical. Because they are identical, the sensitivity equations for θ and η can be replaced by the sensitivity equations for φ. In other words, it is possible to compute sensitivities for the original two parameters by only solving the sensitivity equations for the one new parameter. The approach is similar to the MU referencing used by NONMEM for running the FAST version of FOCE [9]. Below, we generalize the idea beyond the additive definition of φ-parameters.

Definitions

The full vector of fixed effects is defined as the concatenation of the fixed effects used for any φ-parameter, $\boldsymbol{\theta}_{\boldsymbol{\varphi}}$, and their complement, $\boldsymbol{\theta}_{\boldsymbol{c}}$, which only appear on their own,

$$\boldsymbol{\theta} = \boldsymbol{\theta}_{\boldsymbol{\varphi}} \frown \boldsymbol{\theta}_{\boldsymbol{c}}.$$

Similarly for the random effect parameters, we define

$$\boldsymbol{\eta} = \boldsymbol{\eta}_{\boldsymbol{\varphi}} \frown \boldsymbol{\eta}_{\boldsymbol{c}}.$$

We also define the parameter vectors

$$\boldsymbol{p_1} = \boldsymbol{\varphi}(\boldsymbol{\theta}_{\boldsymbol{\varphi}}, \boldsymbol{\eta}_{\boldsymbol{\varphi}}) \frown \boldsymbol{\eta}_{\boldsymbol{c}}$$

and

$$\boldsymbol{p_2} = \boldsymbol{\theta}_{\boldsymbol{c}}.$$

Derivation of relations between sensitivities in the different parameter types

We let $\boldsymbol{x}$ denote the model state variables. The following relations can be derived by applying the chain rule,

$$\begin{aligned}
\frac{d\boldsymbol{x}}{d\boldsymbol{\eta}} &= \frac{d\boldsymbol{x}}{d\boldsymbol{p_1}} \frac{d\boldsymbol{p_1}}{d\boldsymbol{\eta}}, \\
\frac{d\boldsymbol{x}}{d\boldsymbol{\theta}} &= \frac{d\boldsymbol{x}}{d\boldsymbol{p_1}} \frac{d\boldsymbol{p_1}}{d\boldsymbol{\theta}} + \frac{d\boldsymbol{x}}{d\boldsymbol{p_2}} \frac{d\boldsymbol{p_2}}{d\boldsymbol{\theta}}, \\
\frac{d^2\boldsymbol{x}}{d\boldsymbol{\eta}^2} &= \left(\frac{d^2\boldsymbol{x}}{d\boldsymbol{p_1}^2} \frac{d\boldsymbol{p_1}}{d\boldsymbol{\eta}} \right)^{T_{132}} \frac{d\boldsymbol{p_1}}{d\boldsymbol{\eta}} + \frac{d\boldsymbol{x}}{d\boldsymbol{p_1}} \frac{d^2\boldsymbol{p_1}}{d\boldsymbol{\eta}^2}, \\
\frac{d^2\boldsymbol{x}}{d\boldsymbol{\eta} d\boldsymbol{\theta}} &= \left(\left(\frac{d^2\boldsymbol{x}}{d\boldsymbol{p_1}^2} \frac{d\boldsymbol{p_1}}{d\boldsymbol{\theta}} + \frac{d^2\boldsymbol{x}}{d\boldsymbol{p_1} d\boldsymbol{p_2}} \frac{d\boldsymbol{p_2}}{d\boldsymbol{\theta}} \right)^{T_{132}} \frac{d\boldsymbol{p_1}}{d\boldsymbol{\eta}} \right)^{T_{132}} + \frac{d\boldsymbol{x}}{d\boldsymbol{p_1}} \frac{d^2\boldsymbol{p_1}}{d\boldsymbol{\eta} d\boldsymbol{\theta}},
\end{aligned}$$

where the tensor and matrix products of the two last equations are performed so that the proper dimensions are matched. This is done using the tensor transpose operator T_{132}, which transposes the second and third dimension of a tensor. Note that no "back transpose" is needed in the third equation due to symmetry.

A modified estimation algorithm

The existing algorithm can be modified in the following way. The idea is to compute the original first and second order sensitivities of the model state variables, $\boldsymbol{x}$, which are required for the likelihood function, but to do this using sensitivities with respect to $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$. This is achieved by numerically solving the first-order sensitivity equations

$$\begin{aligned}
\frac{d}{dt}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_1}}\right) &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p_1}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_1}}\right) \\
\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_1}}\right)(t_0) &= \frac{\partial \boldsymbol{x}_0}{\partial \boldsymbol{p_1}}, \\
\frac{d}{dt}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_2}}\right) &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p_2}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_2}}\right) \\
\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_2}}\right)(t_0) &= \frac{\partial \boldsymbol{x}_0}{\partial \boldsymbol{p_2}},
\end{aligned}$$

and the second order sensitivity equations

$$\begin{aligned}
\frac{d}{dt}\left(\frac{d^2\boldsymbol{x}}{d\boldsymbol{p_1^2}}\right) &= \frac{\partial^2 \boldsymbol{f}}{\partial^2 \boldsymbol{p_1}} + \left(2\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}\partial \boldsymbol{p_1}} + \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}^2}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_1}}\right)\right)^{T_{132}}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_1}}\right) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\left(\frac{d^2\boldsymbol{x}}{d\boldsymbol{p_1^2}}\right) \\
\left(\frac{d^2\boldsymbol{x}}{d\boldsymbol{p_1^2}}\right)(t_0) &= \frac{\partial^2 \boldsymbol{x}_0}{\partial \boldsymbol{p_1^2}}, \\
\frac{d}{dt}\left(\frac{d^2\boldsymbol{x}}{d\boldsymbol{p_1}d\boldsymbol{p_2}}\right) &= \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{p_1}\partial \boldsymbol{p_2}} + \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{p_1}\partial \boldsymbol{x}}\left(\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_1}}\right) + \left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_2}}\right)\right) \\
&\quad + \left(\left(\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}^2}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_2}}\right)\right)^{T_{132}}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_1}}\right)\right)^{T_{132}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\left(\frac{d^2\boldsymbol{x}}{d\boldsymbol{p_1}d\boldsymbol{p_2}}\right) \\
\left(\frac{d^2\boldsymbol{x}}{d\boldsymbol{p_1}d\boldsymbol{p_2}}\right)(t_0) &= \frac{\partial^2 \boldsymbol{x}_0}{\partial \boldsymbol{p_1}\partial \boldsymbol{p_2}}.
\end{aligned}$$

Note that $d^2\boldsymbol{x}/d\boldsymbol{p_1^2}$ is a symmetric matrix, which means that either the upper or lower triangular part contains redundant sensitivities. The above sensitivities are subsequently transformed back to the original ones according to the relations in the previous section. The derivatives of $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ with respect to the fixed and random effect parameters can be computed symbolically in advance, and the transformation is not expected to be computationally costly. The original sensitivities are then used to compute the population likelihood just like before.

Impact of φ-parameterization on the number of sensitivity equations

The φ-parameterization introduced above leads to fewer sensitivity equations that have to be solved numerically. The impact in terms of computational time is difficult to predict, mainly since Mathematica's ODE solver to some extent can exploit the fact that many symbolic expressions of the ODE right hand sides are identical, which will be the case for models where a φ-parameterization is natural. Nevertheless, we can derive expressions for the number of sensitivity equations that are required in the φ-parameterization and in the original parameterization.

If we let non-bold symbols denote the number of elements in the corresponding bold symbol vectors, the total number of ODEs that have to be solved using the original parameterization is

$$N_{\text{in}}^{\text{old}} = (1+\eta)\,x,$$

for the inner problem, and

$$N_{\text{out}}^{\text{old}} = (1+\eta+\theta+\frac{\eta+\eta^2}{2}+\eta\,\theta)\,x$$

for the outer problem. For the φ-, or new, parameterization, the numbers are

$$N_{\text{in}}^{\text{new}} = (1+\varphi+\eta_c)\,x,$$

for the inner problem and

$$N_{\text{out}}^{\text{new}} = (1+\varphi+\eta_c+\theta_c+\frac{(\varphi+\eta_c)+(\varphi+\eta_c)^2}{2}+(\varphi+\eta_c)\,\theta_c)\,x$$

for the outer problem.

A favorable situation for φ-parameterization is when all all random and fixed effects parameters appear in pairs that can be assigned as new φ-parameters, e.g., $\varphi_1 = \theta_1 + \eta_1$. In this case $\eta = \theta = \varphi$ and $\eta_c = \theta_c = 0$. There are no gains in the inner problem, but the relative number of sensitivity equations in the outer problem is

$$\frac{N_{\text{out}}^{\text{new}}}{N_{\text{out}}^{\text{old}}} = \frac{(1 + \varphi + \frac{\varphi+\varphi^2}{2})\, x}{(1 + \varphi + 3\frac{\varphi+\varphi^2}{2})\, x},$$

which approaches 1/3 in the limit of large φ.

A strategy for choosing a good parameterization

The most efficient choice of parameterization in terms of minimizing the number of sensitivity equations is not always trivial. The following three situations are however helpful in order to understand the effects of introducing φ-parameters.

If a new φ-parameter can be used to reparameterize the appearance of one random effect and one fixed effect, e.g., $\varphi_1 = \theta_1 + \eta_1$, then $N_{\text{in}}^{\text{new}}$ is unchanged but $N_{\text{out}}^{\text{new}}$ is decreased by $(1 + \phi + \eta_c)\, x$. This kind of reparameterization is therefore desirable as the number of sensitivity equations always decrease. If a new φ-parameter is used to reparameterize a fixed effect only, then $N_{\text{in}}^{\text{new}}$ increases by x, and $N_{\text{out}}^{\text{new}}$ increases by $\theta_c\, x$. If, on the other hand, a new φ-parameter is used to reparameterize a random effect only, the number of equations remain constant for both the inner and outer problem.

There may be other more complex situations that require careful consideration as to which parameterization that is better. As a general recommendation, one or a combination of several φ-parameters should only be used to replace a larger number of fixed and random effects.

Extension to additional parameter types

The idea presented above could also be extended to include inter-occasion random effects, $\boldsymbol{\kappa}$. In this scenario, $\boldsymbol{\varphi}$ may also depend on $\boldsymbol{\kappa}$, $\boldsymbol{\varphi} = \boldsymbol{\varphi}(\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\kappa})$. With $\boldsymbol{\kappa} = \boldsymbol{\kappa}_{\boldsymbol{\varphi}}^{\frown}\boldsymbol{\kappa}_{\boldsymbol{c}}$, we redefine $\boldsymbol{p_1} = \boldsymbol{\varphi}(\boldsymbol{\theta}_{\boldsymbol{\varphi}}, \boldsymbol{\eta}_{\boldsymbol{\varphi}}, \boldsymbol{\kappa}_{\boldsymbol{\varphi}})^{\frown}\boldsymbol{\eta}_{\boldsymbol{c}}^{\frown}\boldsymbol{\kappa}_{\boldsymbol{c}}$. The sensitivities of the model state variables with respect to the inter-occasion random effects can now be obtained in a similar way as outlined above for the standard random effects, e.g.,

$$\frac{d\boldsymbol{x}}{d\boldsymbol{\kappa}} = \frac{d\boldsymbol{x}}{d\boldsymbol{p_1}} \frac{d\boldsymbol{p_1}}{d\boldsymbol{\kappa}}.$$

If $\boldsymbol{\kappa_c}$ is empty, this requires no new sensitivity equations at all.

Removing trivial sensitivity equations

The estimation algorithm can be improved by removing trivial sensitivity equations whose solution is time independent, i.e., constant. In the φ-parameterization framework, this works in the following way.

First, a concatenated vector of the original state variables and the sensitivity state variables is formed for the inner problem,

$$\boldsymbol{s_1} = \boldsymbol{x}^{\frown} \operatorname{vec}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_1}}\right),$$

where vec() denotes the flattening of a matrix to a vector. Similarly, we have for the outer problem

$$\boldsymbol{s_2} = \boldsymbol{x}^{\frown} \operatorname{vec}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_1}}\right)^{\frown} \operatorname{vec}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p_2}}\right)^{\frown} \operatorname{vec}\left(\frac{d^2\boldsymbol{x}}{d\boldsymbol{p_1}^2}\right)^{\frown} \operatorname{vec}\left(\frac{d^2\boldsymbol{x}}{d\boldsymbol{p_1} d\boldsymbol{p_2}}\right).$$

The complete systems of ODEs for the inner and outer problem can be then be compactly written

$$\begin{aligned} \frac{d\boldsymbol{s_1}}{dt} &= \mathbf{g_1}(\boldsymbol{s_1}, t, \boldsymbol{p_1}) \\ \boldsymbol{s_1}(t_0) &= \boldsymbol{s_{10}}, \end{aligned}$$

and

$$\begin{aligned} \frac{d\boldsymbol{s_2}}{dt} &= \mathbf{g_2}(\boldsymbol{s_2}, t, \boldsymbol{p_1}, \boldsymbol{p_2}) \\ \boldsymbol{s_2}(t_0) &= \boldsymbol{s_{20}}. \end{aligned}$$

Second, for both the inner and the outer problem, a so called connectivity matrix, $\mathbf{M}$, is defined. For the inner problem, an element $\mathbf{M}_{ij}$ of the connectivity matrix is equal to 1 if the jth element of $\boldsymbol{s_1}$ appears in the ith element of $\mathbf{g_1}$, otherwise it is 0. In addition, all elements on the diagonal of $\mathbf{M}$ are set to 1. The connectivity matrix is defined in the corresponding way for the outer problem. The connectivity matrix is then iteratively applied to a vector $\boldsymbol{z}$

$$\boldsymbol{z}_{n+1} = \mathrm{sgn}(\mathbf{M}\,\boldsymbol{z}_n),$$

until $\boldsymbol{z}_{n+1} = \boldsymbol{z}_n$. The vector $\boldsymbol{z}$ is initialized by setting the ith element to 0 if the ith element of $\mathbf{g_1}$ (after replacing in variables with their initial conditions) is identically equal to 0 at t_0, or setting it to 1 otherwise. The point of the above idea is to identify which (sensitivity) variables that initially are changing, and then see how these changes propagate through the inter-connected system of ODEs (if a variable initially is changing, it may continue to change without influence of other state variables, and therefore we must set $\mathbf{M}_{ii} = 1$). Any zeros in the fixed point iterate of $\boldsymbol{z}$ shows that these variables have a trivial constant solution.

Third, when the trivial equations have been identified, the appearance of the corresponding variables in the other equations must be replaced by their constant solutions.