A Next Generation Method For PK/PD Modeling

Andreas Sidelmann Christensen

Kongens Lyngby 2007
IMM-M.Sc.-2007-18

Supervisors: Bernd Dammann and Henrik Madsen
External supervisor: Niels Rode Kristensen

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Preface

This thesis was prepared at the institute of Informatics and Mathematical Modelling (IMM), the Technical University of Denmark (DTU) in partial fulfillment of the requirements for acquiring the M.Sc. degree in engineering. The work started on September 1, 2006.

This thesis deals with pharmacokinetic and pharmacodynamic modeling based mixed-effects models using stochastic differential equations. A prototype has been implemented in Fortran 95.

The thesis comprises a report and a population stochastic pharmacokinetic and pharmacodynamic modeling prototype implemented in Fortran 95.

Lyngby, February 28, 2007

Andreas Sidelmann Christensen

Abstract

This thesis describes the development of a software prototype implemented in Fortran 95 for population pharmacokinetic/pharmacodynamic (PK/PK) modeling based on non-linear mixed-effects models using stochastic differential equations (SDEs). An advantage of using SDEs is that it allows residual errors to be separated into two fundamentally different types of noise, namely (1) correlated system noise attributed to unmodelled dynamics of the system, and (2) uncorrelated observation noise.

A maximum likelihood method for estimating the fixed- and random-effects parameters in the model is adopted. The likelihood function is approximated numerically using a First-Order Conditional Estimate (FOCE) method and Kalman filtering. The prototype handles linear time-invariant and linear time-varying models.

The choice of Fortran 95 as programming language is motivated by high computational speed, availability of scientific software packages and support of OpenMP shared-library multiprocessing API for parallel computing. With the intent of aiding future model extensions and modifications, the thesis attempts to provide extensive documentation of the program interface and, at the same time, raise awareness of known weaknesses in the implementation.

KEYWORDS: stochastic differential equation (SDE); non-linear mixed-effects; FOCE approximation; Kalman filter; maximum likelihood estimation; pharmacokinetic; pharmacodynamic; PK/PD modeling.

Resumé

Dette eksamensprojekt omhandler konstruktionen af en software prototype implementeret i Fortran 95 til populations farmakokinetik/farmakodynamik (PK/PD) modellering for ikke-lineære mixed-effekt modeller baseret på stokastiske differentialligninger (SDEer). En fordel ved at anvende SDEer er, at de muliggør opsplitning af residualer i to fundamentalt forskelling fejltyper, nemlig (1) korreleret systemstøj, der stammer fra modelmangler eller egentlige tilfældige ændringer i systemet, og (2) ukorreleret målestøj.

En maksimaliseringsmetode anvendes til estimering af faste og tilfældige effekter i modellen. Maksimaliseringsfunktionen tilnærmes numerisk ved hjælp af en første-ordens betinget estimeringssmetode og Kalman filtrering. Prototypen er i stand til at håndtere lineære tidsinvariante og lineære tidsvarierende modeller.

Valget af Fortran 95 som programmeringssprog støttes af hurtig beregningshastighed, adgang til videnskabelige software-pakker, samt understøttelse af OpenMP fælles-bibliotek multiprocessor API til parallelisering. Med henblik på at støtte fremtidige udvidelser og modifikationer af prototype forsøger opgaven at yde udførlig dokumentation for programinterfacet, samt at bringe fokus på kendte svagheder ved implementeringen.

STIKORD: stokastiske differentialligninger (SDEer); ikke-lineær mixed-effekter; første-ordens betinget estimeringsmetode; Kalman filter; maximum likelihood estimering; farmakokinetik; farmakodynamik; PK/PD modellering.

Acknowledgements

I would like to thank my supervisors Bernd Dammann, Henrik Madsen and Niels Rode Kristensen for your support, your interest in the project and motivating discussions. Without Bernd Dammann's irreplaceable expertise and readiness to assist in moments of model development crises, I would have been indeed ill positioned. Thank you!

Likewise, I would like to thank Stig Mortensen and Søren Klim for your valuable insights on the theory and helpful assistance with the model development. It has been very much appreciated, thank you. I wish to thank Hans Bruun Nielsen for help with the implementation of the parameter optimization package *ucminf*.

The project in many aspects opened up a world of challenges, which for the most part were new to me. From high performance computing considerations, relatively large scale model development to the theory of pharmacokinetic and pharmacodynamic modeling and stochastic differential equations. I would therefore like to emphasize that successful accomplishment are accredited the intellectual capacities of my supervisors. Any prone errors in the model implementation or lacks of theoretical understanding are solely ascribable to my own inadequacies.

On a personal note, I would like to thank my parents, my brothers and friends for your love and kind support. Keep it coming. I love you right back.

Symbols and Abbreviations

List of symbols and list of abbreviations are presented in Table 1 and 2 respectively.

Table 1: List of symbols

Symbol	Type	Description
$\boldsymbol{e}_{ij}$	$\mathbb{R}^l$	Measurement noise at time t_{ij}
$\boldsymbol{\eta}_i$	$\mathbb{R}^s$	Individual random-effects
$\boldsymbol{\phi}_i$	$\mathbb{R}^p$	Individual parameters
$\boldsymbol{\theta}$	$\mathbb{R}^q$	Population fixed-effects
$\boldsymbol{\Pi}_{\boldsymbol{w}}$	$\mathbb{R}^{s\times s}$	Magnitude of system noise
$\boldsymbol{\Delta}$	–	Hessian operator
$\boldsymbol{\nabla}$	–	Vector differential operator
$\boldsymbol{\Omega}_i$	$\mathbb{R}^{s\times s}$	Random-effects covariance
$\boldsymbol{\Sigma}_i$	$\mathbb{R}^{l\times l}$	Measurement noise covariance
$\boldsymbol{\epsilon}_{ij}$	$\mathbb{R}^l$	Innovation at time t_{ij}
$\boldsymbol{f}(\cdot)$	$\mathbb{R}^l$	Non-linear function describing the relation between the states and the observations (NL model)

Continued on next page...

Symbol	Type	Description
$\boldsymbol{g}(\cdot)$	$\mathbb{R}^n$	Non-linear function describing the dynamics of the state (NL model)
$\boldsymbol{h}(\cdot)$	$\mathbb{R}^p$	Structural type parameter model describing the dynamics of the individual parameters $\boldsymbol{\phi}_i$
l	$\mathbb{N}\backslash\{0\}$	Dimension of outputs $\boldsymbol{y}_{ij}$
$l_{p,i}$	$\mathbb{R}$	Approximate individual a posteriori log-likelihood $\log_e(L_{p,i})$
m	$\mathbb{N}$	Dimension of inputs $\boldsymbol{u}_{it}$
n	$\mathbb{N}\backslash\{0\}$	Dimension of states $\boldsymbol{x}_{it}$
n_i	$\mathbb{N}\backslash\{0\}$	Number of observations for patient i
p	$\mathbb{N}\backslash\{0\}$	Dimension of individual parameters $\boldsymbol{\phi}_i$
q	$\mathbb{N}\backslash\{0\}$	Dimension of fixed-effects $\boldsymbol{\theta}$
r	$\mathbb{N}$	Dimension of covariates $\boldsymbol{z}_i$
s	$\mathbb{N}$	Dimension of random-effects $\boldsymbol{\eta}_{ij}$
t_{ij}	$\mathbb{R}$	Time of j^{th} observation for individual i
$\boldsymbol{u}_{ij}$	$\mathbb{R}^m$	Inputs at time t_{ij}
$\hat{\boldsymbol{x}}_{i(t\|j)}$	$\mathbb{R}^n$	Updated state given observations at time t_{ij}
$\boldsymbol{y}_{ij}$	$\mathbb{R}^l$	Outputs at time t_{ij}
$\hat{\boldsymbol{y}}_{i(j\|j-1)}$	$\mathbb{R}^l$	Output prediction at time t_{ij} given observations at time $t_{i(j-1)}$
$\boldsymbol{z}_i$	$\mathbb{R}^r$	Covariates for individual i
$\mathcal{Y}_{ij}$	–	$\mathcal{Y}_{ij} = [\boldsymbol{y}_{i1}, \ldots, \boldsymbol{y}_{ij}]$ represents all observations until time t_{ij} for individual i
L	$\mathbb{R}$	Approximate population likelihood
$\boldsymbol{A}(\cdot)$	$\mathbb{R}^{n\times n}$	Coefficients for $\boldsymbol{x}_{it}$ in state equation (LTI/LTV)
$\boldsymbol{B}(\cdot)$	$\mathbb{R}^{n\times m}$	Coefficients for $\boldsymbol{u}_{it}$ in state equation (LTI/LTV)
$\boldsymbol{C}(\cdot)$	$\mathbb{R}^{l\times n}$	Coefficients for $\boldsymbol{x}_{ij}$ in output equation (LTI/LTV)
$\boldsymbol{D}(\cdot)$	$\mathbb{R}^{l\times m}$	Coefficients for $\boldsymbol{u}_{ij}$ in output equation (LTI/LTV)
$\boldsymbol{K}_{ij}$	$\mathbb{R}^{n\times l}$	Kalman gain at time t_{ij}
$L_{p,i}$	$\mathbb{R}$	Approximate individual a posteriori likelihood

Continued on next page...

Symbol	Type	Description
N	$\mathbb{N}\backslash\{0\}$	Number of patients
P_s	$\mathbb{R}$	Scaling factor for initial state covariance $\boldsymbol{P}_0$
$\boldsymbol{P}_{i(t\|j)}$	$\mathbb{R}^{n\times n}$	State covariance given observations at time t_{ij}
$\boldsymbol{R}_{i(j\|j-1)}$	$\mathbb{R}^{l\times l}$	Output prediction covariance at time t_{ij} given observations at $t_{i(j-1)}$

Table 2: List of abbreviations

Abbreviation	Description
APL	Approximate population likelihood
CSV	Comma-separated values
CTSM	Continuous time stochastic modeling
DTU	Technical University of Denmark
FDA	Food & Drug Administration
FOCE	First-order conditional estimation
EKF	Extended Kalman filter
IM	Intramuscular
IMM	Informatics and Mathematical Modelling
IV	Intravenous
KF	Kalman filter
LTI	Linear time-invariant
LTV	Linear time-varying
ML	Maximum likelihood
NL	Non-linear
NLME	Non-linear mixed-effects
ODE	Ordinary differential equation
PD	Pharmacodynamic
PK	Pharmacokinetic
PSM	Population stochastic modeling
SDE	Stochastic differential equation
SSM	State space model
SSSM	Stochastic state space model

Contents

Chapter 1

Introduction

In recent years, population pharmacokinetic/pharmacodynamic (PK/PD) modeling has become an increasingly important tool for analyzing the dose-exposure-effect relationship of drugs in humans and animals.

Generally speaking, the main interest in population PK/PD studies is twofold, namely (1) to identify the overall tendency of the parameters across an entire population, and (2) to describe departures from the population trend among subgroups of individuals [3]. Subgroups may be identified by the factorial levels of a set of covariates defined by, for example, demographics, genetic information, co-medication, environmental aspects and disease states [13]. By recognizing that subgroups may not respond similarly to the same treatment, population PK/PD modeling makes it possible to indicate individualized optimal levels of, for example, dose-regimen for one or several subgroups associated with certain characterics.

Recognizing that different subgroups of patients within a population may require different dosing strategies, the Danish Medicines Agency's[1] "Working group on clinical pharmacy" in 2005 recommended that clinical pharmacists perform pharmacokinetic services. Referencing to several studies, the authors argued that individualized medial treatment based on PK data may improve clinical benefits and savings on hospitalisation [19].

[1] *The Danish Medicines Agency:* Lægemiddelstyrelsen (da.)

Furthermore, the use of PK/PD modeling in pharmaceutical development has lately received increasing regulatory awareness. In 1999, the U.S. Food & Drug Administration (FDA) published the "Population Pharmacokinetics" guidance for the pharmaceutical industry [4], which motivates the integration of PK/PD analysis as a part of clinical trials.

Population PK/PD analysis is often accomplished using non-linear mixed-effects (NLME) models, since it allows modeling of data from several patients by decomposing the intra-individual variability into *inter-individual* and *intra-individual* variability. NLME models are typically combined with ordinary differential equations (ODEs) with uncorrelated residuals. This technique, however, inherits several limitations. For example, correlations between residuals are not uncommon in population PK/PD analysis and, secondly, deficiencies in the structural model are not appreciated [12].

Recently, several publications have given evidence to the benefits of using stochastic differential equations (SDEs) in population PK/PD modeling. SDEs in state space models (SSM) make it possible to separate system noise from measurement noise. Furthermore, correlated system variability is allowed in SDEs.

Applying SDEs to PK/PD analysis based on NLME models allows decomposition of the intra-variability into (1) *system noise* caused by unmodelled dynamics or true random fluctuations in the system and (2) non-correlated *measurement noise*.

In 2006, Mortensen and Klim [10] presented a novel software prototype named "PSM"[2] for population PK/PD analysis based on NLME models using SDEs. The PSM prototype was implemented in Matlab® using the maximum likelihood method for parameter estimation suggested by Overgaard [12].

This thesis pursuits the development of a new PSM prototype in a high-level scientific programming language with particular attention to optimal computational efficiency and speed.

Emanating from the successful Matlab® implementation and the extensive experiences with stochastic state-space modeling (SSSM) at the Department of Informatics & Mathematical Modelling at DTU, the vision is to provide the first steps of a new generation of software for population PK/PD analysis that may contribute to improved medical treatment of patients and better tools for pharmaceutical drug development.

[2]PSM: acronym of "Population Stochastic Modelling"

1.1 Objectives

The objectives of this thesis are formally expressed here:

1. To develop a prototype for population PK/PD modeling based NLME models with SDEs.
2. To choose a functional programming language that ensures:
   - High computational speed
   - Availability of efficient procedures for numerical manipulations
   - Parallelization (future work)
3. To formulate a conceptual design that provides a flexible, transparent and generic program interface. Importantly, the model construction should facilitate the needs of future model modifications and development.

In agreement with the broadly formulated goals above, a set of specifications for the proposed prototype were explicitly defined in Table 4.1 in Chapter 4. The specifications were imposed with the intent of ensuring that the project was completed in due course.

The final section of this chapter briefly outlines the rest of this thesis.

1.2 Outlines

This report consist of the following parts:

Chapter 2 explains the fundamental terminology of population PK/PD modeling and describes existing software for PK/PD modeling with reference to the current program development.

The theory for NLME models based on SDEs is expressed mathematically in **Chapter 3**. The two-stage hierarchical model and the Kalman filtering algorithm are formulated mathematically. Finally, the population likelihood method for estimating the parameters in the population models is formally introduced.

Chapter 4 presents the design and structure of the proposed prototype for population PK/PD analysis. The major program units, procedure interfaces and program flowcharts are documented.

Details of the model implementation are emphasized in **Chapter 5** . Validation results are provided for the individual-level modeling part of the population PK/PD prototype.

Chapter 6 specifies the input and output interface of the proposed prototype. The chapter accounts for all steps involved in declaring, building and running a model.

Chapter 7 discusses the results of the model implementation, which gives rise to a set of recommendations for future work.

Chapter 8 states the main points in the discussion and concludes on the thesis objectives.

Chapter 2

PK/PD Modeling

This chapter presents an overview of the characteristics of pharmacokinetic and pharmacodynamic (PK/PK) analysis and introduces the general terminology.

Pharmacokinetics constitute the part of pharmacology concerned with the movement of pharmaceutical drug entities in the body, whereas pharmacodynamics relate to the effects of pharmaceuticals and the mechanisms of their actions and elimination. General terms used in PK/PD analysis are described shortly in Section 2.1.

Section 2.2 concludes the introduction to PK/PD modeling by giving a short summary of existing software based on non-linear mixed-effects (NLME) models for PK/PD analysis .

2.1 Terms in PK/PD modeling

Pharmacokinetics

Pharmacokinetics (PK) describes the relationship between the drug availability in the body, especially at the sites where the drug is active, and drug admin-

istration. The availability of drugs is typically expressed in terms of the drug concentration and is concerned with three types of processes, namely absorption, distribution and elimination[1].

Essentially, PK analysis seeks to identify the factors that influence the dose-concentration relationship. The information gathered from PK studies is used to identify appropriate dose-regimen in clinical practise.

Pharmacodynamics

Pharmacodynamics (PD) defines the study of the relationships between concentration and the magnitude of the biological or physiological effect of a drug [7].

The inter-individual variations in drug concentrations due to individual PK properties only partly explains why individuals experience different responses to medical treatment. In other words, reponses vary across a population of individuals who are exposed the same drug concentrations. PD analysis studies the exposure-effect relationship within individuals and its variability among individuals in a population of interest.

Figure 2.1 illustrates the biological processes affecting the PK/PD drug properties.

Population PK/PD modeling

The purpose of population PK/PD modeling is to make inference on the mechanisms governing individual profiles of repeated measurements of the response and how the individual profiles vary across a population.

Attention is devoted to identification of subgroups of patients within a population that, based characteristic covariate patterns, show identical response.

Population PK/PD analysis are commonly carried out by means of NLME models, which allow for simultaneous estimation of inter- and intra-individual variability (random-effects). An important advantage of NLME models is that it enables analysis of PK/PD data obtained from both scattered observations and unbalanced study designs [16].

[1] *Elimination:* may be achieved by degradation, chemical alteration as a natural part of the metabolism or through excretion.

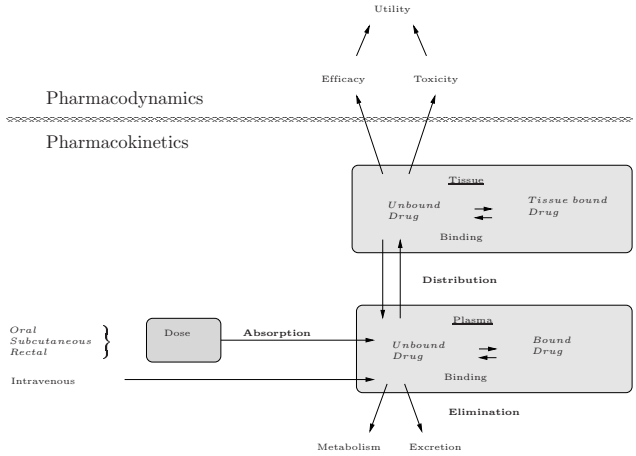

Figure 2.1: Illustration of the biological processes governing PK/PD properties [6].

Alternatively, the standard two-stage method may be used for population modeling, but this approach is known to yield overestimated inter-individual variability. Onward, this thesis only considers PK/PD analysis based on NLME models.

Figure 2.2 grahically shows an example of PK data from a study, where 12 patients received identical oral doses (mg/kg) of a drug (Theophylline). It illustrates that the individual concentration profiles are similarly-shaped, while maxima, gradients and durations of decay vary across the population. This variability may be attributed to inter-individual variations in the PK processes and, consequently, gives support to the need of population PK/PD modeling.

The following section focuses on the software available for PK/PD modeling.

2.2 Software for PK/PD analysis

Several commercial and non-commercial software packages exist for PK/PD modeling. Their intended use generally depends on the type of model and kind of information available in each case. This section briefly introduces two existing software packages available for PK/PD analysis, namely NONMEM® and CTSM.

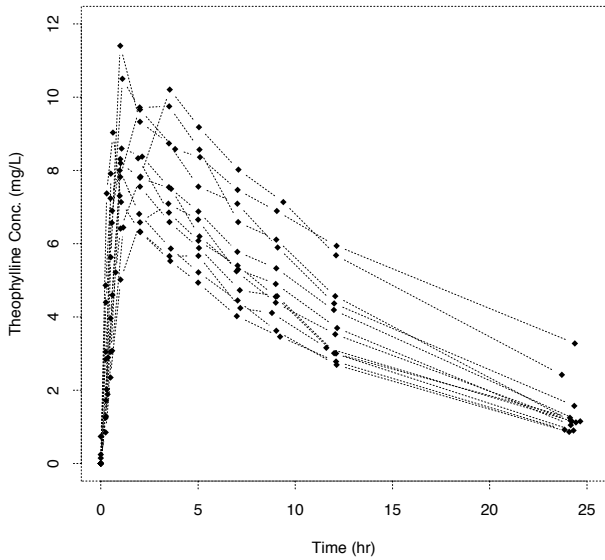

Figure 2.2: Illustration of concentration profiles for 12 subjects in a Theophylline study receiving an identical oral dose (mg/kg) [2].

NONMEM®

NONMEM is developed by the NONMEM Project at the University of California. Today, NONMEM is the de facto standard software package for population PK/PD modeling. It is written and distributed in ANSI FORTRAN 77. The "NONMEM" name is the acronym of "NON-linear Mixed-Effects Models".

NONMEM conducts non-linear mixed-effects modeling of population PK/PD data based on ordinary differential equations, but Tornøe has previously demonstrated how to formulate a population PK/PD models based on SDEs in NONMEM (version VI Beta) [17]. However, since NONMEM is a commercial software based on proprietary source code, it naturally imposes several restrictions to the flexibility of incorporating SDEs.

CTSM

CTSM, short name for Continuous Time Stochastic Modelling, is a program for modeling of semi-physical dynamic systems [8]. CTSM is developed at the Department of Informatics and Mathematical Modelling (IMM), DTU, and has

served various applications such as modeling of PK/PD systems. It is written in FORTRAN 77.

CTSM permits modeling of dynamic systems based on SDEs. The program handles both linear and non-linear models. Furthermore, CTSM offers two methods for parameter estimation, namely the *maximum likelihood* or the *maximum a posteriori* methods. In PK/PK analysis, CTSM is limited to single subject modeling.

In 2006, Mortensen and Klim [10] accomplished the implementation of a prototype for population PK/PD modeling based on NLME models using SDEs. The prototype was written in Matlab® and provided evidence that the implementation of SDEs in a population PK/PD modeling framework was technically possible.

This project represents the next step by, founded on the same theoretical principles, undertaking the implementation of a prototype in a programming language that contrasted with Matlab offers higher computational speed and parallel computing features.

The task of developing a new prototype is encouraged by the possibility of creating a more efficient model design for population PK/PD analysis based on SDEs. Both NONMEM and CTSM share the common trait of having been developed over several decades. Therefore, advances in mathematical and statistical theory related to PK/PD modeling as well as developments within computational capacities might not have been fully appreciated in the design of these software packages. These consideration supported the choice of formulating the prototype from scratch.

Allowing the program development to start from scratch makes it possible to define an appropriate structure and interface, where dependence on redundant features of earlier versions is avoided. Starting from scratch also makes it possible to exploit the object-oriented structure of modern high-level programming languages, such as C and Fortran 95, which was not available in FORTRAN 77.

As a further matter, the development of the prototype aims to take into account the data workflow in clinical trials. This, for example, has led to a slight modification of the data input interface compared to NONMEM.

On a final note, this work is motivated by the possibility providing improved software tools for population PK/PD modeling that, so is the hope, ultimately will lead to improved understanding about pharmaceutical compounds and treatment of disease.

Chapter 3

Population Modeling Theory

This chapter mathematically defines the non-linear mixed-effects (NLME) models based on stochastic differential equations (SDEs) for population pharmacokinetic and pharmacodynamic (PK/PD) modeling. The formulas presented in the following sections define the mathematical framework for the proposed prototype, which will be discussed in the remaining chapters.

The theory presented here originates from Overgaard [12], Mortensen and Klim [10], Tornøe [16] and Kristensen et al. [9]. The presentation pursues a rather concise mathematical formulation of the model, as extensive descriptions of the theoretical background and mathematical derivations are available in the mentioned publications.

The chapter consists of six sections. First, Section 3.1 defines the notation and terminology. Section 3.2 introduces the principles of NLME modeling and explains how this gives rise to a two-stage hierarchical model structure.

The first-stage in the hierarchy, which models the intra-individual variability, is formulated in Section 3.3 by means of stochastic state-space models (SSSMs). The first-stage modeling is also denoted *individual-level modeling* part of the analysis. Section 3.4 defines the recursive Kalman filter for estimating the states in a stochastic state-space model.

The second-stage model, which represents the *population-level modeling* part of the analysis, is defined in Section 3.5.

Finally, Section 3.6 concludes the chapter with the definition of the likelihood method suggested by Overgaard [12] for estimating the *fixed-* and *random-effects* parameters in NLME model.

3.1 Notation and terminology

The following notation and terms apply

Individual index $i = \{1, \ldots, N\}$ identifies the i^{th} individual in the model, which contains N individuals.

Time index $j = \{1, \ldots, n_i\}$ identifies the j^{th} discrete time point t_{ij} for the i^{th} individual, where n_i is the total number of observations for individual i.

Scalar expressions are written in *italic* font, e.g. $l_{p,i}$.

Vectors are written in ***bold italic*** font using small letters, e.g. $\boldsymbol{y}_{ij}$. All vectors are defined as column vectors.

Matrices are written in ***bold italic*** font using capital letters, e.g. $\boldsymbol{A}$.

For simplicity, an abbreviated nomenclature for the time-dependency of variables is, for example, defined by $\boldsymbol{y}_i(t_{ij}) = \boldsymbol{y}_{ij}$, which identifies the j^{th} observation obtained at time t_{ij} for individual i.

Predictions are indicated by hat-notation "ˆ", e.g. $\hat{\boldsymbol{x}}_{i(j|j-1)}$, where subscript $i(j|j-1)$ refers to the j^{th} prediction based on all $j-1$ preceeding observations for individual i. $\mathcal{Y}_{ij} = [\boldsymbol{y}_{i1}, \ldots, \boldsymbol{y}_{ij}]$ represents all observations until time t_{ij} for individual i.

The symbol $p(X)$ denotes the density of X and $p(X|Y)$ the conditional density of X given Y.

Synonyms for the *output* variables (or *outputs*), which are used interchangeably, include *response* variables (or *responses*) and *measurement* variables (or *measurements*).

3.2 Non-linear mixed-effects models

Population PK/PD analysis is typically conducted using NLME modeling. NLME models for repeated observations defines a hierarchical model structure, which makes it possible to separate the variability in the reponses into inter- and intra-individual components.

A two-stage[1] hierarchical model is adopted. It allows simultaneous estimation of (1) the random-effects associated with inter- and intra-individual variability and (2) the fixed-effects parameters influenced by measured concomitant effects or covariates.

The two-stage hierarchical structure of NLME models for estimating the parameters in population PK/PK analysis may in short be conveyed as follows

First-stage: *Individual-level* model for determining the intra-individual variability, e.g. attributed to individual processes in the body (absorption, distribution and elimination).

Second-stage: *Population-level* model for identification of the inter-individual variation, e.g. due to systematic variation in covariates or unexplained variation across a population (represented by random-effects parameters).

The underlying assumption of the first-stage model is that the biological processes acting on each individual are based on the same mechanisms [13]. As illustrated in Figure 2.2, the assumption implies that individual profiles are similarly-shaped, whereas individual variations in, for example, peak levels and steepness of gradients are due to individual properties only.

In other words, biological processes such as absorption, distribution and elimination that govern pharmacokinetic profiles assumably follow the same mechanisms.

The second-stage model is constructed upon the assumption that the individual parameters responsible for the varying individual profiles can be regarded as realizations from a second-stage probability function.

The next section briefly highlights the advantages of using NLME models based on SDEs rather than ordinary differential equantions (ODEs).

[1] A Bayesian analysis requires the addition of a third-stage to account for the prior distributions on the population parameters, see Sheiner and Wakefield [13].

First-stage model based on SDEs

When using ODEs to formulate the structural model of a system, the intra-individual variability is entirely modeled as residual noise. Applying SDEs to NLME models extends the first-stage model by decomposing the intra-individual variation into two fundamentally different types of noise, namely

1. *System noise* representing model deficiencies, e.g. if the model does not capture the dynamics of the states or if true random fluctuations are present in the system.
2. *Measurement noise* accounting for uncorrelated error, e.g. due to assay error or if the observed concentration not correctly portraits the true contration at the site of interest.

Based on the preceding introductory description of NLME models, the subsequent sections mathematically defines the two-stage NLME models based on SDEs. The first-stage setup is stated in Section 3.3 with the stochastic state-space filtering scheme given in Section 3.4. Section 3.5 expresses the second-stage model.

3.3 Individual-level modeling

The NLME model postulated in this chapter handles outputs structured as

$$\boldsymbol{y}_{ij}, \quad i = 1, \ldots, N; \quad j = 1, \ldots, n_i \tag{3.1}$$

where $\boldsymbol{y}_{ij} \in \mathcal{Y} \subset \mathbb{R}^l$ is a vector of output variables at time t_{ij} for individual i in the population of interest; N is the number of individuals in the population; and n_i the number of observations obtained for the i^{th} individual.

The first-stage model is defined by a stochastic state-space model consisting of SDEs for the states $\boldsymbol{x}_{it}$ evolving in continuous time and a set of outputs $\boldsymbol{y}_{ij}$ sampled a discrete time points, i.e.

$$d\boldsymbol{x}_{it} = \boldsymbol{g}(\boldsymbol{x}_{it}, \boldsymbol{u}_{it}, t, \boldsymbol{\phi}_i)dt + \boldsymbol{\Pi}_{\boldsymbol{w}}(\boldsymbol{u}_{it}, t, \boldsymbol{\phi}_i)d\boldsymbol{w}_{it} \tag{3.2}$$

$$\boldsymbol{y}_{ij} = \boldsymbol{f}(\boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}, t_{ij}, \boldsymbol{\phi}_i) + \boldsymbol{e}_{ij} \tag{3.3}$$

where for the i^{th} individual, $\boldsymbol{x}_{it} \in \mathcal{X} \subset \mathbb{R}^n$ is a vector of state variables, e.g. the amount of drug in a PK model; $\boldsymbol{u}_{it} \in \mathcal{U} \subset \mathbb{R}^m$ is a vector of input variables, e.g. dose administration; $t \in \mathbb{R}$ is time; $t_{ij} \in \mathbb{R}$ is the j^{th} measurement time; $\boldsymbol{\phi}_i \in \Phi \subset \mathbb{R}^p$ is the vector of individual parameters; $\boldsymbol{\Pi}_{\boldsymbol{w}} d\boldsymbol{w}_{it}$ is the system noise, where $\boldsymbol{\Pi}_{\boldsymbol{w}}$ is a scaling term representing the magnitude of the system noise[2] and $\{\boldsymbol{w}_{it}\}$ is an n-dimensional standard Wiener process[3]; $\{\boldsymbol{e}_{ij}\}$ is an l-dimensional white noise process[4] with zero mean and variance $\boldsymbol{\Sigma}$; and finally, $\boldsymbol{g}(\cdot)$ and $\boldsymbol{f}(\cdot)$ are non-linear vector functions describing the dynamics of the states and the relationship between the state and the observations, respectively.

The notation $d\boldsymbol{x}_{it}/dt$ in equation (3.2) is not applicable, since the time derivative of the standard Wiener process $d\boldsymbol{w}/dt$ is poorly defined. Here, the Itô-method is adopted for numerical computation the integrals.

It is noted that the equation (3.2) simplifies to an ordinary differential equation in the case where the magnitude of the system noise term reduces to zero, $\boldsymbol{\Pi}_{\boldsymbol{w}} = \mathbf{0}$. In that case, all intra-individual variability is contained in the measurement error covariance $\boldsymbol{\Sigma}$.

Equations (3.2) and (3.3) are called the *state equation* and the *observation equation*, repectively, and specify the general, non-linear structural model for the intra-individual data.

At present, the proposed prototype exclusively handles linear models, which is a subset of the non-linear models. These will formally be defined next.

Special case: Linear time-varying model

The following notation applies for linear time-varying (LTV) stochastic state-space model

[2] In PK/PD literature, the magnitude of the system noise $\boldsymbol{\Pi}_{\boldsymbol{w}}$ is also commonly defined by the symbol $\boldsymbol{\sigma}_w$. In this thesis, the symbol $\boldsymbol{\Pi}_{\boldsymbol{w}}$ has been chosen in order to comply with the general bold-face capital letter notation of matrices.

[3] *Standard Wiener process:* (also called Brownian motion) is a continuous time Gaussian process, which for each increment $(\boldsymbol{w}_{t_1} - \boldsymbol{w}_{t_2})$ is characterized by mean $E[\boldsymbol{w}_{t_2} - \boldsymbol{w}_{t_1}]$ and variance $V[\boldsymbol{w}_{t_2} - \boldsymbol{w}_{t_1}]$:

$$\begin{aligned} E[\boldsymbol{w}_{t_2} - \boldsymbol{w}_{t_1}] &= \mathbf{0} \\ V[\boldsymbol{w}_{t_2} - \boldsymbol{w}_{t_1}] &= |t_2 - t_1|\boldsymbol{I} \end{aligned}$$

[4] *White noise process:* independent and identically distributed Gaussian measurements noise with zero mean and covariance $\boldsymbol{\Sigma}$.

$$
\begin{aligned}
d\boldsymbol{x}_{it} &= (\boldsymbol{A}(\boldsymbol{x}_{it}, \boldsymbol{u}_{it}, t, \boldsymbol{\phi}_i)\boldsymbol{x}_{it} + \boldsymbol{B}(\boldsymbol{x}_{it}, \boldsymbol{u}_{it}, t, \boldsymbol{\phi}_i)\boldsymbol{u}_{it})dt \\
&\quad + \boldsymbol{\Pi}_{\boldsymbol{w}}(\boldsymbol{u}_{it}, t, \boldsymbol{\phi}_i)d\boldsymbol{w}_{it} \qquad (3.4) \\
\boldsymbol{y}_{ij} &= \boldsymbol{C}(\boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}, t, \boldsymbol{\phi}_i)\boldsymbol{x}_{ij} + \boldsymbol{D}(\boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}, t, \boldsymbol{\phi}_i)\boldsymbol{u}_{ij} + \boldsymbol{e}_{ij} \qquad (3.5)
\end{aligned}
$$

where for the i^{th} individual, the explanations of t, t_{ij}, $\boldsymbol{x}_{it}$, $\boldsymbol{u}_{ij}$, $\boldsymbol{\phi}_i$, $\boldsymbol{\Pi}_{\boldsymbol{w}}$, $d\boldsymbol{w}_{it}$ and $\boldsymbol{e}_{ij}$ remain unchanged. $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ are coefficient matrices in the state equation for the states and the outputs, respectively; $\boldsymbol{C} \in \mathbb{R}^{l\times n}$ and $\boldsymbol{D} \in \mathbb{R}^{l\times m}$ are correspondingly cofficient matrices in the observation equation.

Special case: Linear time-invariant model

Similarly, the linear time-invariant (LTI) stochastic state-space model is expressed by:

$$
\begin{aligned}
d\boldsymbol{x}_{it} &= (\boldsymbol{A}(\boldsymbol{\phi}_i)\boldsymbol{x}_{it} + \boldsymbol{B}(\boldsymbol{\phi}_i)\boldsymbol{u}_{it})dt \\
&\quad + \boldsymbol{\Pi}_{\boldsymbol{w}}(\boldsymbol{\phi}_i)d\boldsymbol{w}_{it} \qquad (3.6) \\
\boldsymbol{y}_{ij} &= \boldsymbol{C}(\boldsymbol{\phi}_i)\boldsymbol{x}_{ij} + \boldsymbol{D}(\boldsymbol{\phi}_i)\boldsymbol{u}_{ij} + \boldsymbol{e}_{ij} \qquad (3.7)
\end{aligned}
$$

where for the i^{th} individual, the explanations of t, t_{ij}, $\boldsymbol{x}_{it}$, $\boldsymbol{u}_{ij}$, $\boldsymbol{\phi}_i$, $\boldsymbol{\Pi}_{\boldsymbol{w}}$, $d\boldsymbol{w}_{it}$ and $\boldsymbol{e}_{ij}$ are the same. $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ are coefficient matrices in the state equation for the states and the outputs, respectively; $\boldsymbol{C} \in \mathbb{R}^{l\times n}$ and $\boldsymbol{D} \in \mathbb{R}^{l\times m}$ are cofficient matrices in the observation equation.

The Kalman filter, which in thesis is the preferred method for modeling the states of the systems, is defined in the following section.

3.4 Kalman filter

The objective of state-space modeling is to estimate the unobservable states $\boldsymbol{x}_{it}$ for a series of responses $\{\boldsymbol{y}_{ij}\}, j = 1, 2, \ldots, n_i$, influenzed by unknown variability. This section presents the Kalman filter technique for state-space modeling.

The Kalman filter is a recursive minimum variance-covariance estimator for the states of dynamic systems. Given a set of individual parameters $\boldsymbol{\phi}_i$ and

the initial conditions of the state $\hat{\boldsymbol{x}}_{i0}$ and $\boldsymbol{P}_{i0}$, the Kalman filter algorithm recursively estimates the current state and state covariance of the system [1].

In the general case of non-linear evolution in the states and/or responses, i.e. when $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}(\cdot)$ in equations (3.2)-(3.3) are non-linear, the extended Kalman filter (EKF) is applicable. Essentially, the extended Kalman filter creates a linearization around the current states through computation of the Jacobian matrix[5] using an appropriate ODE-solver.

At this moment, the proposed prototype only permits specification of linear time-varying (LTV) or linear time-invariant (LTI) models as defined in equations (3.4)-(3.5) and (3.6)-(3.7), respectively. Since, by assumption, the system and measurement noise are independent of the states, the extended Kalman filter reduces to an ordinary Kalman filter (KF).

The ordinary Kalman filtering scheme for the first-stage model, expressed as a linear stochastic state-space model, is the topic of the remainder of this section. The theory is based on "CTSM Math Guide" by Kristensen and Madsen [8], which should be consulted if additional theoretical insight is desired.

The Kalman filter formally consists of two parts called the *updating* and the *prediction* part. In prediction part, predictions of the states, covariance and observation at time point t_{ij} are derived from the current state and covariance given observations at time point t_{ij-1}. In the updating part, the Kalman filter updates the estimate of the states and covariances in presence of new observations.

To initiate the recursive Kalman filter, the initial state $\hat{\boldsymbol{x}}_{i(1|0)} = \hat{\boldsymbol{x}}_{i0}$ and state covariance $\boldsymbol{P}_{i(1|0)} = \boldsymbol{P}_{i0}$ must be defined for all $i = 1, \ldots, N$ individuals.

Initial states and covariances

Initial conditions for the recursive Kalman filter are defined by

$$\hat{\boldsymbol{x}}_{i(1|0)} = \hat{\boldsymbol{x}}_{i0} \tag{3.8}$$

$$\boldsymbol{P}_{i(1|0)} = P_s \int_{t_1}^{t_2} e^{\boldsymbol{A}_{it}\, s} \boldsymbol{\Pi}_{\boldsymbol{w}} \boldsymbol{\Pi}_{\boldsymbol{w}}{}^T (e^{\boldsymbol{A}_{it}\, s})^T ds = \boldsymbol{P}_{i0} \tag{3.9}$$

[5] Jacobian matrix: matrix of first-order partial derivatives.

where P_s is an arbitrary scaling factor, which punishes for unknown uncertainty of the initial estimate for the state prediction covariance.

Output prediction equations

The prediction equations identifies the one-step output prediction $\hat{\boldsymbol{y}}_{i(j|j-1)}$ and corresponding output prediction covariance matrix $\boldsymbol{R}_{i(j|j-1)}$

$$\begin{aligned} \hat{\boldsymbol{y}}_{i(j|j-1)} &= \boldsymbol{C}\hat{\boldsymbol{x}}_{i(j|j-1)} + \boldsymbol{D}\boldsymbol{u}_{ij} && (3.10) \\ \boldsymbol{R}_{i(j|j-1)} &= \boldsymbol{C}\boldsymbol{P}_{i(j|j-1)}\boldsymbol{C}^T + \boldsymbol{\Sigma} && (3.11) \end{aligned}$$

Given output prediction $\hat{\boldsymbol{y}}_{i(j|j-1)}$, the innovation $\boldsymbol{\epsilon}_{ij}$ at time t_{ij} indicates the residual difference between the incoming observation $\boldsymbol{y}_{ij}$ and the output prection $\hat{\boldsymbol{y}}_{i(j|j-1)}$ derived at the preceding time point t_{ij-1}, i.e.:

$$\boldsymbol{\epsilon}_{ij} = \boldsymbol{y}_{ij} - \hat{\boldsymbol{y}}_{i(j|j-1)} \quad (3.12)$$

State updating equations

The state updating part of the Kalman filter describes the situation, when a new observation $\boldsymbol{y}_{ij}$ is obtained at time point t_{ij}. Given of new information, the state prediction $\hat{\boldsymbol{x}}_{i(j|j-1)}$ can be updated. For this purpose the Kalman gain $\boldsymbol{K}_{ij}$ is defined

$$\boldsymbol{K}_{ij} = \boldsymbol{P}_{i(j|j-1)}\boldsymbol{C}^T\boldsymbol{R}^{-1}_{i(j|j-1)} \quad (3.13)$$

The Kalman gain $\boldsymbol{K}_{ij}$ dictates to which the extend the updated states $\hat{\boldsymbol{x}}_{ij}$ should rely on the new observations. The state updating equations are given by

$$\begin{aligned} \hat{\boldsymbol{x}}_{i(j|j)} &= \hat{\boldsymbol{x}}_{i(j|j-1)} + \boldsymbol{K}_{ij}\boldsymbol{\epsilon}_{ij} && (3.14) \\ \boldsymbol{P}_{i(j|j)} &= \boldsymbol{P}_{i(j|j-1)} - \boldsymbol{K}_{ij}\boldsymbol{R}_{i(j|j-1)}\boldsymbol{K}^T_{ij} && (3.15) \end{aligned}$$

From equation (3.13) noted that the Kalman gain is proportional to the state covariance $\boldsymbol{P}_{i(j|j-1)}$ and inverse proportional to the output covariance $\boldsymbol{R}_{i(j|j-1)}$. Thus, for relatively larger state covariances the Kalman gain becomes larger, indicating that the updated state $\hat{\boldsymbol{x}}_{i(j|j)}$ in equation (3.14) should rely more on the incoming observation.

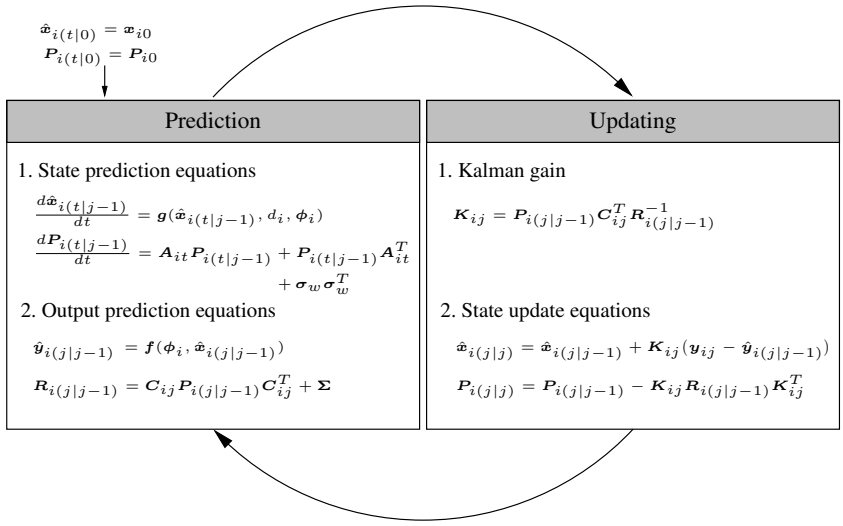

Figure 3.1: Illustration of the Kalman filtering algorithm for non-linear models. The following variations in notation applies: individual index i is omitted; time index k; magnitude of system noise σ [18].

State prediction equations

Finally, the one-step state prediction equations are expressed as the evolution in the states for $t \in [t_{ij}, t_{ij+1}]$:

$$\frac{d\hat{\boldsymbol{x}}_{i(t|j)}}{dt} = \boldsymbol{A}\hat{\boldsymbol{x}}_{i(t|j)} + \boldsymbol{B}\boldsymbol{u}_{it}, \quad t \in [t_{ij}, t_{ij+1}] \tag{3.16}$$

$$\frac{d\boldsymbol{P}_{i(t|j)}}{dt} = \boldsymbol{A}\boldsymbol{P}_{i(t|j)} + \boldsymbol{P}_{i(t|j)}\boldsymbol{A}^T + \boldsymbol{\Pi}_{\boldsymbol{w}}{\boldsymbol{\Pi}_{\boldsymbol{w}}}^T, \quad t \in [t_{ij}, t_{ij+1}] \tag{3.17}$$

where the abbreviated notation for the linear time-varying (LTV) and the linear

time-invariant (LTI) case are represented in equation (3.18) and (3.19), respectively:

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{A}(\hat{\boldsymbol{x}}_{i(t|j-1)}, \boldsymbol{u}_{it}, t, \boldsymbol{\phi}_i), & \boldsymbol{B} &= \boldsymbol{B}(\hat{\boldsymbol{x}}_{i(t|j-1)}, \boldsymbol{u}_{it}, t, \boldsymbol{\phi}_i) \\ \boldsymbol{C} &= \boldsymbol{C}(\hat{\boldsymbol{x}}_{i(t|j-1)}, \boldsymbol{u}_{it}, t, \boldsymbol{\phi}_i), & \boldsymbol{D} &= \boldsymbol{D}(\hat{\boldsymbol{x}}_{i(t|j-1)}, \boldsymbol{u}_{it}, t, \boldsymbol{\phi}_i) \\ \boldsymbol{\Pi}_{\boldsymbol{w}} &= \boldsymbol{\Pi}_{\boldsymbol{w}}(\boldsymbol{u}_{it}, t, \boldsymbol{\phi}_i), & \boldsymbol{\Sigma} &= \boldsymbol{\Sigma}(\boldsymbol{u}_{it}, t, \boldsymbol{\phi}_i) \end{aligned} \tag{3.18}$$

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{A}(\boldsymbol{\phi}_i), & \boldsymbol{B} &= \boldsymbol{B}(\boldsymbol{\phi}_i) \\ \boldsymbol{C} &= \boldsymbol{C}(\boldsymbol{\phi}_i), & \boldsymbol{D} &= \boldsymbol{D}(\boldsymbol{\phi}_i) \\ \boldsymbol{\Pi}_{\boldsymbol{w}} &= \boldsymbol{\Pi}_{\boldsymbol{w}}(\boldsymbol{\phi}_i), & \boldsymbol{\Sigma} &= \boldsymbol{\Sigma}(\boldsymbol{\phi}_i) \end{aligned} \tag{3.19}$$

Figure 3.1 illustrates the Kalman filtering algorithm for a non-linear stochastic state-space model.

Solutions to SDEs

This section defines the mathematical equations for solving the SDEs of the state prediction equations expressed in equations (3.14)-(3.15). No derivations are stated, as the purpose is solely to define the equations involved in the implementation of the prototype.

Again, the mathematical equations presented here originates from Kristensen and Madsen [8], where detailed information of the mathematical derivations and theoretical background is found.

Expressing the state prediction equations (3.16) and (3.17) with respect to discrete time and integrating yields

$$\begin{aligned} \hat{\boldsymbol{x}}_{i(j+1|j)} &= E\{\boldsymbol{x}_{i(t_{ij+1})}|\boldsymbol{x}_{i(t_{ij})}\} \\ &= e^{\boldsymbol{A}(t_{ij+1}-t_{ij})}\hat{\boldsymbol{x}}_{i(j|j)} + \int_{t_{ij}}^{t_{ij+1}} e^{\boldsymbol{A}(t_{ij+1}-s)}\boldsymbol{B}\boldsymbol{u}_s ds \end{aligned} \tag{3.20}$$

$$\begin{aligned}\boldsymbol{P}_{i(j+1|j)} &= E\{\boldsymbol{x}_{i(t_{ij+1})}\boldsymbol{x}_{i(t_{ij+1})}^T|\boldsymbol{x}_{i(t_{ij})}\} \\ &= e^{\boldsymbol{A}(t_{ij+1}-t_{ij})}\boldsymbol{P}_{i(j|j)}\left(e^{\boldsymbol{A}(t_{ij+1}-t_{ij})}\right)^T \\ &\quad + \int_{t_{ij}}^{t_{ij+1}} e^{\boldsymbol{A}(t_{ij+1}-s)}\boldsymbol{\Pi_w}\boldsymbol{\Pi_w}^T\left(e^{\boldsymbol{A}(t_{ij+1}-s)}\right)^T \end{aligned} \tag{3.21}$$

Defining the time step between two on each other following time points, $\tau_s = t_{ij+1} - t_{ij}$ as well as the matrix exponential $\boldsymbol{\Psi}_s = e^{\boldsymbol{A}\tau_s}$, equations (3.20) and (3.21) can be expressed as

$$\begin{aligned}\hat{\boldsymbol{x}}_{i(j+1|j)} &= \boldsymbol{\Psi}_s\hat{\boldsymbol{x}}_{i(j|j)} - \int_0^{\tau_s} e^{\boldsymbol{A}s}sds\boldsymbol{B}\boldsymbol{\alpha} \\ &\quad + \int_0^{\tau_s} e^{\boldsymbol{A}s}ds\boldsymbol{B}(\boldsymbol{\alpha}\tau_s + \boldsymbol{u}_{ij}) \end{aligned} \tag{3.22}$$

$$\boldsymbol{P}_{i(j+1|j)} = \boldsymbol{\Psi}_s\boldsymbol{P}_{i(j|j)}{\boldsymbol{\Psi}_s}^T + \int_0^{\tau_s} e^{\boldsymbol{A}s}\boldsymbol{\Pi_w}\boldsymbol{\Pi_w}^T\left(e^{\boldsymbol{A}s}\right)^T ds \tag{3.23}$$

where $\boldsymbol{\alpha}$ defines the hold on the inputs:

$$\boldsymbol{\alpha} = \frac{\boldsymbol{u}_{ij+1} - \boldsymbol{u}_{ij}}{t_{ij+1} - t_{ij}} \tag{3.24}$$

Zero-order hold is defined by $\boldsymbol{\alpha} = \boldsymbol{0}$, while *first-order hold* is given by $\boldsymbol{\alpha} \neq \boldsymbol{0}$. Currently, the proposed prototype computes state predictions $\hat{\boldsymbol{x}}_{i(j+1|j)}$ based on zero-order hold only.

The matrix exponential $\boldsymbol{\Psi}_s = e^{\boldsymbol{A}\tau_s}$ as well as the integral in equation (3.21) is computed simultanously by means of a Padé approximation with repeated scaling and squaring

$$\exp\left(\begin{bmatrix}-\boldsymbol{A} & \boldsymbol{\Pi_w}\boldsymbol{\Pi_w}^T \\ \boldsymbol{0} & \boldsymbol{A}^T\end{bmatrix}\tau_s\right) = \begin{bmatrix}\boldsymbol{H}_1(\tau_s) & \boldsymbol{H}_2(\tau_s) \\ \boldsymbol{0} & \boldsymbol{H}_3(\tau_s)\end{bmatrix} \tag{3.25}$$

from which is found:

$$\boldsymbol{\Psi}_s = \boldsymbol{H}_3^T(\tau_s) \tag{3.26}$$

$$\int_0^{\tau_s} e^{\boldsymbol{A}s}\boldsymbol{\Pi}_{\boldsymbol{w}}{\boldsymbol{\Pi}_{\boldsymbol{w}}}^T(e^{\boldsymbol{A}s})^T ds = \boldsymbol{H}_3^T(\tau_s)\boldsymbol{H}_2(\tau_s) \tag{3.27}$$

The state prediction $\boldsymbol{P}_{i(j+1|j)}$ is, thus, completely expressed by application of equations (3.26) and (3.27) to equation (3.23).

In the case of singular matrix $\boldsymbol{A}$, singular value decomposition (SVD) of $\boldsymbol{A}$ and $\boldsymbol{\Psi}_s$ is needed to compute the integrals in equation (3.22), i.e.

$$\tilde{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \boldsymbol{U}^T\boldsymbol{A}\boldsymbol{U} = \begin{bmatrix} \tilde{\boldsymbol{A}}_1 & \tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{3.28}$$

$$\tilde{\boldsymbol{\Psi}}_s = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \boldsymbol{U}^T\boldsymbol{\Psi}_s\boldsymbol{U} = \begin{bmatrix} \tilde{\boldsymbol{\Psi}}_s^{\,1} & \tilde{\boldsymbol{\Psi}}_s^{\,2} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{3.29}$$

The prototype treats two special cases for computations of the state predictions $\hat{\boldsymbol{x}}_{i(j+1|j)}$, namely

1. Case A: Singular A, zero-order hold on inputs
2. Case B: Non-singular B, zero-order hold on inputs

Mathematical treatment of the two special cases A and B terminates the presentation of the ordinary Kalman filtering scheme applied in the prototype.

Case A: State prediction for singular A, zero-order hold

For singular matrix $\boldsymbol{A}$ and zero-order hold on the inputs ($\boldsymbol{\alpha} = \boldsymbol{0}$), the state prediction $\hat{\boldsymbol{x}}_{i(j+1|j)}$ identified by:

$$\hat{\boldsymbol{x}}_{i(j+1|j)} = \boldsymbol{\Psi}_s\hat{\boldsymbol{x}}_{i(j|j)} - \boldsymbol{U}\int_0^{\tau_s} e^{\tilde{\boldsymbol{A}}s}ds\boldsymbol{U}^T\boldsymbol{B}\boldsymbol{u}_{ij} \tag{3.30}$$

$$\int_0^{\tau_s} e^{\tilde{\boldsymbol{A}}s}ds = \begin{bmatrix} \tilde{\boldsymbol{A}}_1^{-1}\left(\tilde{\boldsymbol{\Psi}}_s^{\,1} - \boldsymbol{I}\right) & \tilde{\boldsymbol{A}}_1^{-1}\left(\tilde{\boldsymbol{A}}_1^{-1}\left(\tilde{\boldsymbol{\Psi}}_s^{\,1} - \boldsymbol{I}\right) - \boldsymbol{I}\tau_s\right)\tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{I}\tau_s \end{bmatrix} \tag{3.31}$$

Case B: State prediction for non-singular A, zero-order hold

For non-singular matrix $\boldsymbol{A}$ and zero-order hold on the inputs ($\boldsymbol{\alpha} = \mathbf{0}$), the state prediction $\hat{\boldsymbol{x}}_{i(j+1|j)}$ identified by

$$\hat{\boldsymbol{x}}_{i(j+1|j)} = \boldsymbol{\Psi}_s \hat{\boldsymbol{x}}_{i(j|j)} + \int_0^{\tau_s} e^{\boldsymbol{A}s} ds \boldsymbol{B} \boldsymbol{u}_{ij} \tag{3.32}$$

$$\int_0^{\tau_s} e^{\boldsymbol{A}s} ds = \boldsymbol{A}^{-1}(\boldsymbol{\Psi}_s - \boldsymbol{I}) \tag{3.33}$$

This completes the mathematical formulation of the Kalman filtering algorithm for modeling the states in the structural model.

Section 3.5 defines the structural type model for the individual parameters, which comprise the population-level modeling part of the hierachical NLME model structure.

3.5 Population-level modeling

The second-stage model describes the inter-individual variations. In this thesis, the individual parameters $\boldsymbol{\phi}_i$ is modeled as a function of the fixed-effects $\boldsymbol{\theta}$, the individual covariates $\boldsymbol{z}_i$ and random-effects $\boldsymbol{\eta}_i$, i.e.

$$\boldsymbol{\phi}_i = \boldsymbol{h}(\boldsymbol{\theta}, \boldsymbol{z}_i) \exp(\boldsymbol{\eta}_i), \qquad i = 1, \ldots, N \tag{3.34}$$

where $\boldsymbol{h}(\cdot)$ denotes the structural type parameter model; $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^q$ is a vector of fixed-effects parameters; $\boldsymbol{z}_i$ is a r-dimensional covariate vector of the ith individual; and $\boldsymbol{\eta}_i \sim N(\mathbf{0}, \boldsymbol{\Omega})$ are vectors of individual random-effects parameters. This model formulation restricts the random-effects $\boldsymbol{\eta}_i$ from changing the sign of $\boldsymbol{h}(\boldsymbol{\theta}, \boldsymbol{z}_i)$.

The presentation of the individual-level modeling part in Sections 3.3 and 3.4 and the population-level modeling part expressed here concludes the mathematical formulation of the two-stage hierarchical model.

The final section defines mathematical setup for estimating the fixed- and random-effects parameters in the population PK/PK model using NLME models based on SDEs.

3.6 Parameter estimation

This thesis adopts the maximum likelihood method for estimating the entire set of population parameters ($\boldsymbol{\theta}$, $\boldsymbol{\eta}_i$) in the NLME model based SDEs, which was first proposed by Overgaard [12]. This section presents the maximum likelihood theory in short.

When the intra-individual model contains correlated residuals, the first-stage joint probability density must be approximated by the product of probability densities conditional on both the individual parameters $\boldsymbol{\phi}_i$ and the previous observations.

Without further introduction, application of Bayes rule $P(A \cap B) = P(B|A)P(A)$ gives rise to the following definition of the first-stage condition density function

$$p_1(\mathcal{Y}_{in_i}|\boldsymbol{\phi}_i, \boldsymbol{u}_i) \quad = \quad \left(\prod_{j=2}^{n_i} p(\boldsymbol{y}_{ij}|\mathcal{Y}_{i(j-1)}, \boldsymbol{\phi}_i, \boldsymbol{u}_i) \right) p(\boldsymbol{y}_{i1}|\boldsymbol{\phi}_i, \boldsymbol{u}_i) \qquad (3.35)$$

where $\mathcal{Y}_{ij} = [\boldsymbol{y}_{i1}, \ldots, \boldsymbol{y}_{ij}]$ represents all observations until time t_{ij} for individual i.

The first-stage likelihood function L_i for the ith individual is defined as the product of the probabilities of the individual observations $\boldsymbol{y}_{ij}$, $j = 1, \ldots, n_i$, so that

$$L_i(\boldsymbol{\phi}_i|\mathcal{Y}_{in_i}) \quad = \quad p_1(\mathcal{Y}_{in_i}|\boldsymbol{\phi}_i, \boldsymbol{u}_i) \qquad (3.36)$$

Assuming that the first-stage conditional densities are Gaussian distributions, the quasi-likelihood function can be derived subsequently. When modeling, this assumption should readily be tested, e.g. using standardized residuals.

The approximate Gaussian conditional densities are completely described by means of the conditional mean and covariance of the outputs

$$\hat{\boldsymbol{y}}_{i(j|j-1)} = E(\boldsymbol{y}_{ij}|\mathcal{Y}_{i(j|j-1)}, \boldsymbol{\phi}_i, \boldsymbol{u}_i) \tag{3.37}$$
$$\boldsymbol{R}_{i(j|j-1)} = V(\boldsymbol{y}_{ij}|\mathcal{Y}_{i(j|j-1)}, \boldsymbol{\phi}_i, \boldsymbol{u}_i) \tag{3.38}$$

The one-step prediction error $\boldsymbol{\epsilon}_{ij}$ at time t_{ij} is given by:

$$\boldsymbol{\epsilon}_{ij} = \boldsymbol{y}_{ij} - \hat{\boldsymbol{y}}_{i(j|j-1)} \in N(\boldsymbol{0}, \boldsymbol{R}_{i(j|j-1)}) \tag{3.39}$$

Using the preceding formulas, the Gaussian approximation of the first-stage density function given in equation (3.35) is expressed as

$$p_1(\mathcal{Y}_{ij}|\boldsymbol{\phi}_i, \boldsymbol{u}_i) \approx \prod_{j=1}^{n_i} \frac{\exp\left(-\frac{1}{2}\boldsymbol{\epsilon}_{ij}^T \boldsymbol{R}_{i(j|j-1)}^{-1} \boldsymbol{\epsilon}_{ij}\right)}{\sqrt{|2\pi \boldsymbol{R}_{i(j|j-1)}|}} \tag{3.40}$$

which is exact in the special case of linear models. Inserting in equation (3.40) into (3.36) and taking the logarithm yields the *individual log-likelihood*

$$l_i(\boldsymbol{\phi}_i|\mathcal{Y}_{ij}) \approx -\frac{1}{2}\sum_{j=1}^{n_i}\left(\boldsymbol{\epsilon}_{ij}^T \boldsymbol{R}_{i(j|j-1)}^{-1} \boldsymbol{\epsilon}_{ij} + |2\pi \boldsymbol{R}_{i(j|j-1)}|\right) \tag{3.41}$$

The individual log-likelihood l_i constitutes the objective function first-stage model. The optimal set of individual random-effects parameters $\hat{\boldsymbol{\eta}}_i$ maximize the objective function.

$$\arg\max_{\boldsymbol{\eta}_i} l_i(\boldsymbol{\phi}_i|\mathcal{Y}_{in_i}) \tag{3.42}$$

The second-stage density $p_2(\boldsymbol{\eta}_i|\boldsymbol{\Omega})$ is assumed a multivariate Gaussian density, which is identical to the assumption when modeling based on ODEs. The *population likelihood* L is derived through combination of the second-stage probability density with the first-stage probability density $p_1(\mathcal{Y}_{ij}|\boldsymbol{\phi}_i, \boldsymbol{u}_i)$ using Bayes theorem

$$L(\boldsymbol{\theta}, \boldsymbol{\eta}|\mathcal{Y}) \propto \prod_{i=1}^{N} \int p_1(\mathcal{Y}_{ij}|\boldsymbol{\phi}_i, \boldsymbol{u}_i) p_2(\boldsymbol{\eta}_i|\boldsymbol{\Omega}) d\boldsymbol{\eta}_i \tag{3.43}$$

$$= \prod_{i=1}^{N} \int \exp(l_{p,i}) d\boldsymbol{\eta}_i \tag{3.44}$$

where $\boldsymbol{\eta} = \{\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \ldots, \boldsymbol{\eta}_N\}$; $\mathcal{Y} = \{\mathcal{Y}_{1n_1}, \ldots, \mathcal{Y}_{Nn_N}\}$; and $l_{p,i}$ is the *approximate individual a posteriori log-likelihood* for the ith individual based on one-step prediction errors $\boldsymbol{\epsilon}_{ij}$, namely

$$\begin{aligned} l_{p,i} &= -\frac{1}{2} \sum_{j=1}^{n_i} \left(\boldsymbol{\epsilon}_{ij}^T \boldsymbol{R}_{i(j|j-1)}^{-1} \boldsymbol{\epsilon}_{ij} + \log|2\pi \boldsymbol{R}_{i(j|j-1)}| \right) \\ &\quad -\frac{1}{2} \boldsymbol{\eta}_i^T \boldsymbol{\Omega}^{-1} \boldsymbol{\eta}_i - \frac{1}{2} \log|2\pi \boldsymbol{\Omega}| \end{aligned} \tag{3.45}$$

The approximate population likelihood function L defined in equation (3.44) cannot be solved analytically. Numerical approximation using a second-order Taylor expansion is given equation (3.46).

The First-Order Conditional Estimation (FOCE) method (using only first-order derivatives) with expansion around the optimal set of random-effects $\hat{\boldsymbol{\eta}}_i$ is used to evaluate the approximate population likelihood L.

$$L(\boldsymbol{\theta}, \boldsymbol{\eta}|\mathcal{Y}) \approx \prod_{i=1}^{N} |\boldsymbol{\Delta} l_{p,i}|^{-\frac{1}{2}} \exp\left[l_{p,i} - \frac{1}{2} \boldsymbol{\nabla} l_{p,}^T \boldsymbol{\Delta} l_{p,i}^{-1} \boldsymbol{\nabla} l_{p,i} \right] \tag{3.46}$$

$$\approx \prod_{i=1}^{N} |\boldsymbol{\Delta} l_{p,i}|^{-\frac{1}{2}} \exp\left(l_{p,i}\right) \Big|_{\hat{\boldsymbol{\eta}}_i} \tag{3.47}$$

since the maximum individual log-likelihood is characterized by zero gradient, $\boldsymbol{\nabla} l_{i|\hat{\boldsymbol{\eta}}_i} = 0$, thus equation (3.46) reduces to (3.47).

Determination of the approximate population log-likelihood function L requires evaluation of the individual log-likelihood function l_i. Using the Laplacian approximation, the Hessian matrix of the individual likelihood function $\boldsymbol{\Delta} L_i$ is expressed by

$$\boldsymbol{\Delta} l_i \approx -\sum_{i=1}^{n_i} \left[\boldsymbol{\nabla} \boldsymbol{\epsilon}_{ij}^T \boldsymbol{R}_{i(j|j-1)}^{-1} \boldsymbol{\nabla} \boldsymbol{\epsilon}_{ij} \right] - \boldsymbol{\Omega}^{-1} \tag{3.48}$$

This concludes the description of the maximum likelihood principles for estimating the entire set of parameters in the population PK/PD setup using NLME models based on SDEs.

In Chapter 4, the design of the Fortran 95 prototype is presented.

Chapter 4

Design of Prototype

This thesis' primary objective is to propose a prototype that – so is the vision – could become the basis of a next generation software for population pharmacokinetic and pharmacodynamic (PK/PD) modeling. In this chapter, the implementation of such prototype in the programming language Fortran 95 is presented.

From the very beginning of the model development, strict attention has been given to the formulation of a favorable design that, at best, addresses the requirements and challenges of future model extensions and, in the worst case, is prevented from imposing any potential restrictions.

The prototype exploits features of the Fortran 95 language, for example *derived types*[1], to specify generic procedure interfaces that easily allow modifications to data transfer without requiring changes to be made in the argument lists. *Modules* constitute another capacity of the Fortran 95 language, which the implementation makes much use of, as it supports a logical modular structure and also extends the options for handling and protecting data.

The prototype is named "PSM" being the acronym of "Population Stochastic Modeling". Consequently, this thesis adopts the originally proposed by Mortensen and Klim [10]. Per definition, the prototype provides a preliminary

[1] *Derived type*: in other object-oriented programming languages also refered to as *objects*.

model for a next generation software for population PK/PD modeling. As such, and motivated by the perspective of fueling future model developments, the following chapters aims at pinpointing possible shortcomings and give suggestions for improvements of the current implementation.

Due to the size and, thus, relative complex nature of the PSM prototype, an overall perspective is pursued in the descriptions of the invidual model components. This approach is supported by the extensive interface documentation for all procedures, derived types and include files supplied in appendix A. This, it is hoped, will help new developers experience a quick start despite the complexity of the model.

First, Section 4.1 restates the objectives and defines the particular specifications for the PSM prototype development. Section 4.2 presents the programming platform and the rationale for selecting Fortran 95 as programming language. Section 4.4 provides an elaborate summary of the entire file tree structure and model interface of the PSM prototype. Section 4.5 ends the chapter with a presentation of the calling sequences illustrated by means of flowcharts.

4.1 Objectives and specifications

As conveyed in Chapter 1, the goals of this thesis are:

1. To develop a prototype for population PK/PD modelling based non-linear mixed-effects models with stochastic differential equations.
2. To choose a functional programming language that ensures:
    - High computational speed
    - Availability of efficient procedures for numerical manipulations
    - Parallelization (future work)
3. To formulate a conceptual design that provides a flexible, transparent and generic program interface. Importantly, the model construction should facilitate the needs of future model modifications and development.

Based on the objectives, a set of specifications were imposed to the model development. Table 4.1 defines the model specifications and, additionally, indicates the major restrictions to the PSM prototype.

Table 4.1: Design specifications

Target	Specifications	Restrictions
Models	Non-linear mixed-effects model based on stochastic differential equations:	
	1) Linear time-varying (LTV) 2) Linear time-invariant (LTI)	Non-linear (NL) not implemented
	with state predictions estimated for:	
	1) Singular $\boldsymbol{A}$ 2) Non-singular $\boldsymbol{A}$	
	using zero-order hold on inputs ($\alpha = 0$)	First-order hold ($\alpha = 1$)
Model declaration	Models must be entirely specified by user in a set of model files[†]. In addition, user must specify:	
	1) Number of individuals 2) Number of rows in datafiles	Not automatically identified
Data input	Data is obtained from three datafiles, which are specified at run-time:	
	1) inputs/outputs 2) doses 3) covariates	Doses not analyzed
	Required: outputs Optional: inputs, doses, covariates	
	Data should be complete	No missing data
	Individual number of observations (input/output/doses) allowed.	
Results	Results are rendered to screen:	No output files
	1) Approx. population likehood L 2) Fixed-effects $\boldsymbol{\theta}$ 3) Random-effects $\boldsymbol{\eta}_i$	No graphical display of results
	Random-effects are printed in chronological order starting with the individual no. 1 in datafiles.	

[†] See Chapter 6 for detailed information on how to declare models.

It is made clear that, although the prototype currently only permits declaration of linear state-space models, the notation *non-linear* mixed-effects is kept with reference to the general framework of PK/PD analysis. The ambition is that the prototype will become subject for extensions, including the incorporation of non-linear state-space models.

Considering the magnitude of the PSM prototype, it was decided to give relatively higher priority to implementation rather than validation. Consequently, individual model entities were validated on a routinely basis. In the end, however, the completion of the approximate population likelihood procedure (`APL`) was favorized on the expense of outstanding validation of this part. Validation is discussed in Chapter 5.

The proposed PSM prototype has been implemented in Fortran 95 and the rationale herefore is addressed in Section 4.2.

4.2 Programming platform

The programming language of choice for the PSM prototype is Fortran 95. Fortran is a standardized, procedural programming language particularly well-suited for numerical computation and scientific computing.

The choice of Fortran 95 is motivated by:

- Availability of Sun Studio's Sun Performance Library[15], which comprise a set of optimized, high-speed mathematical subroutines for solving linear algebra and other numerically intensive problems. The Sun Performance Library is based on *BLAS 1-3*[2] and *LAPACK*[3] standard libraries.
- Availability of scientific packages implemented in Fortran for, for instance, parameter optimization, procedures for matrix exponential and automatic differentiation.
- Availability of OpenMP[4] package, which is a shared-memory multiprocessing API for developing parallel models. Multi-processor parallelization via

[2]BLAS: Basic Linear Algebra Subroutines are procedures for basic linear algebra operations, e.g. vector and matrix multiplication, and is written in FORTRAN 77.

[3]LAPACK: Linear Algebra Package is a software library for numerical computing written in FORTRAN 77.

[4]OpenMP API: Open Multi-Processing Application Program Interface, see also: `http://www.openmp.org`

OpenMP is encompassed by a set of meta tags and may be implemented without significant modification of the serial source code.

- Fortran 90/95 provides object-oriented programming features and matrix allocation options that were not available to FORTRAN 77.

The PSM prototype has been developed on a Unix platform running on Ultra-SPARC-IV processors using the Sun Studio Fortran Compiler `f95`[5]. The processor settings are summarized in Table 4.2.

Table 4.2: Summary of CPU settings.

```
Informations on CPU:
        CPU name: UltraSPARC-IV
           clock: 1200
        l1-assoc: 4
     l1-linesize: 32
         l1-size: 65536
        l2-assoc: 2
     l2-linesize: 128
         l2-size: 8388608
     tlb-entries: 16
        tlb-size: 8192
```

4.3 File structure

The PSM prototype consists of a total of 19 files that comprise more than 6,000 lines of Fortran 95 code (not including external procedures). The files are distributed in three directories, namely the main directory ('`/`') and the two sub-directories '`/INCLUDE`' and '`/USER`'.

The files may be separated into three categories according to their respective kind and function. The three parts are described by:

Format specification part: consists of a single Fortran 95 include file located in the sub-library '`/INCLUDE`'. Defines all input/output formats used PSM.

Model declaration part: contains 13 files which comprise an entire model declaration. Model declaration files are uniquely located in the sub-directory '`/USER`' and should be modified appropriately by the user prior to model building.

[5]For improved compilation speed, the most recent Sun Studio Express Fortran compiler available on G-bar was accessed via the command '`init.ssxp`'

Source code part: defines the engine of the PSM program, that is, the entire set of subroutines, functions, modules and makefile. Consisting of 5 files placed in the main directory, these files should normally not be altered when setting up models in PSM.

Figure 4.1 identifies the files in PSM and illustrates the location of the files in the respective directories.

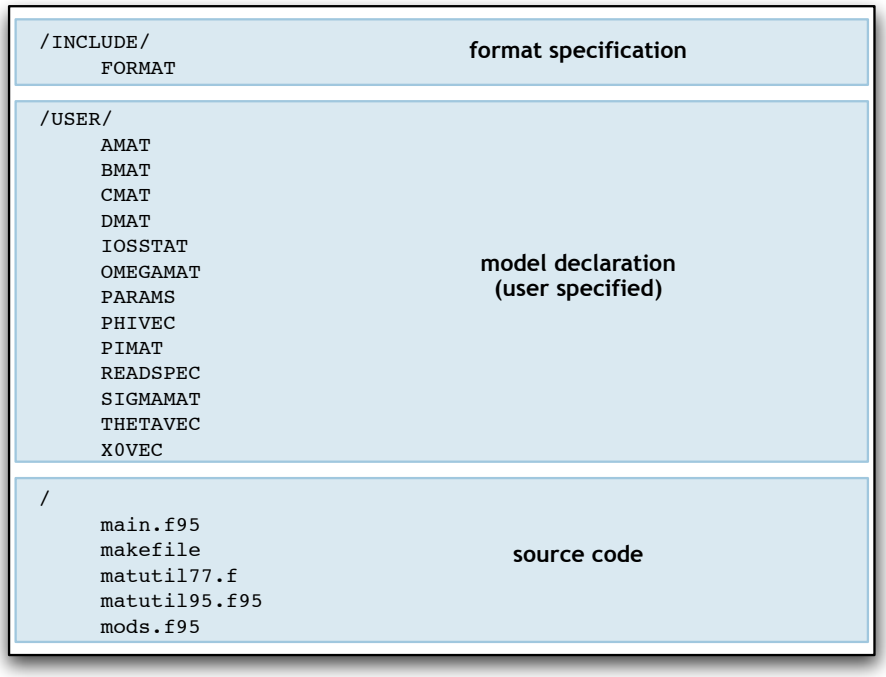

Figure 4.1: File tree structure with individual parts highlighted.

The architecture was chosen in order accommodate proper order and file maintenance. In particular, it was considered preferable to separate user-modifiable model declaration files from the source code.

Later, if a Graphical User Interface (GUI) is implemented, the architectural considerations may become irrelevant. In that case, the user may be given the option of choosing specific directories for each model declaration file.

4.4 Program units

The PSM prototype is composed by several individual units of different kinds. Five *modules* define the major program units. The scope of the modules is to define *procedures*, *derived types* and/or to hold data, which are considered the minor entities of the program.

As indicated in the introduction of this chapter, the interface descriptions for all procedures, derived types and include files are supplemented in appendix A. The ambition in the following sections is, therefore, to provide a concise summary of the program structure while largely sticking to a rather global perspective.

The modules will be defined next in Section 4.4.1. Secondly, the derived types and procedures are listed in Sections 4.4.3 and 4.4.4, respectively.

4.4.1 Modules

A Fortran module is unit designed with the intention of making definitions, data and procedures available to other units. Five modules with specific purpose and properties have been defined, see Table 4.3.

Table 4.3: Summary of modules in PSM.

Name	Description	Location
`MODATA`	Defines procedures for data acquisition and displaying data.	mods.f95
`MOIOS`	Defines input/output control parameters.	mods.f95
`MOPARAMS`	Defines major global parameters.	mods.f95
`MOPROCS`	Defines procedures written and/or modified for PSM[†]	mods.f95
`MOTYPES`	Defines the derived types applied in PSM.	mods.f95

[†] Remaining external, non-modified procedures are located in the source files `matutil95.f95` and `matutil77.f`.

Modules are accessed through `USE` association, which makes available all module definitions, data and procedures to the calling program unit. Alternatively, using the `'USE, ONLY:'` statement, specific units within a module may be accessed while leaving others out. This feature is widely applied in PSM as means to

ensure proper data protection.

It is mentioned that bugs in the Sun Studio 11 Fortran compiler signficantly increase compilation speeds, when applying `'USE, ONLY'`. In order to avoid this, latest Sun Studio Express compiler (beta-version) should be used.

The organization of components of similar kinds in separate modules, which is good programming practice for large-scale programs, has been adopted in PSM. For instance, `MOTYPES` is limited to defining the derived types, and `MOPARAMS` is restricted to the definitions of global parameters. This arrangement helps to provide intutive procedure interfaces and makes modifications simple and intuitive.

The module `MOPROCS` contains the subroutines and functions written explicitly for PSM. Furthermore, it contains the *ucminf* parameters optimization procedures, which have been slightly modified in order to match the requirements of PSM. Changes to *ucminf* are discussed in Section 5.3

The module `MODATA` defines input/output procedures. Currently, `MOIOS` only hold a single input/output control parameter `IOVAR`. Displaying results has not been throughly considered in this work. Therefore, future developers should decide whether modifications to this setup should be made, i.e. if input/output control is represented by a single parameter, it might possibly be included in `MOPARAMS`.

4.4.2 Global parameters

The major global parameters are defined during model declaration. Global parameters are predominantly defined in the model declaration files `PARAMS`, `READSPEC` and `IOSSTAT`.

Table 4.4 describes the global parameters based on the interface description of `PARAMS` in Table A.8, `READSPEC` in Table A.11 and `IOSSTAT` in Table A.2.

It is noted that the prototype not automatically identifies neither the number of patients in the input/output datafile nor the number of lines in the datafiles.

Table 4.4: Summary of global parameters in PSM.

Name	Description	Location
NPHI	Number of individual parameters p.	PARAMS
NU	Number of input variables m.	PARAMS
NX	Number of state variables n.	PARAMS
NY	Number of output variables l.	PARAMS
NZ	Number of covariates r.	PARAMS
NETA	Number of random-effects s.	PARAMS
NTHETA	Number of fixed-effects q.	PARAMS
IMODEL	Defines model for Kalman filtering procedure (depends on data and model structure). For population modeling: = 0: LTI-model (linear time-invariant) = 1: LTV-model (linear time-varying) For individual modeling: =10: LTI-model (linear time-invariant) =11: LTV-model (linear time-varying)	PARAMS
PS	Pre-specified initial state covariance scaling factor P_s, see Eq. 3.9	
LB	Lower bounds for mapping of fixed-effects parameters $\boldsymbol{\theta}$.	PARAMS
LU	Upper bounds for mapping of fixed-effects parameters $\boldsymbol{\theta}$.	PARAMS
IOVAR	I/O control variable. = 0: No print of intermediate results = 1: Print intermediate results	IOSSTAT
NID	Number of patients N in model (must be identical to number of individuals contained in inputs/outputs datafile).	READDAT
NROWS_DATA	Number of rows in inputs/outputs datafile.	READDAT
NROWS_DATA	Number of rows in dosing datafile.	READDAT

4.4.3 Data objects

Module `MOTYPES` is restricted to the definitions of derived types, also called data objects. Table 4.5 summarizes the six types of objects encountered in PSM and refers to the individual interface descriptions given in appendix A.3.

Table 4.5: Summary of data objects in PSM. All data objects are defined in module `MOTYPES`

Name	Description	Interface
`DOSE`	Data object for storing individual patient data related to dose administration of pharmaceutical compounds. `DOSE` is part of the derived type `PATIENT`, which contains data for a single patient.	Table A.15
`ETAOBJECT`	Data object containing variables used for computation of individual random-effects $\boldsymbol{\eta}_i$.	Table A.16
`KALOBJ`	Data object containing variables used in Kalman filtering.	Table A.17
`OPTIMOBJECT`	Object containing variables used for parameter optimization of $\boldsymbol{\theta}$ and $\boldsymbol{\eta}_i$.	Table A.18
`PATIENT`	Data object for storing individual patient data.	Table A.19
`THETAOBJECT`	Data object containing variables used for computation of the fixed-effects $\boldsymbol{\theta}$.	Table A.20

Additionally, it is emphasized that:

- The data object `PATIENT` stores the information provided in input datafiles, for example inputs, outputs, covariates and dosing data. No results are stored in `PATIENT`. Data protection is therefore ensured using `INTENT(IN)` attribute, which prevents data from being modified throughout the entire program.

- Acquisition and storage of dose data from input dose datafile is implemented. Dose data is stored in derived type `DOSE`, which is a part of the derived type `PATIENT`.
- Parameter estimation results for the fixed-effects $\boldsymbol{\theta}$ and the random-effects $\boldsymbol{\eta}_i$, $i = 1, \ldots, N$, are allocated in `THETAOBJECT` and `ETAOBJECT`, respectively. A vector of N element of the type `ETAOBJECT's` are generated in each model.
- The `OPTIMOBJECT` is used in the definitions of `THETAOBJECT` and `ETAOBJECT`. It allows structuring of population/individual optimization data, which may be derived from *ucminf*. Analysis of optimization data not yet incorporated in the current model, but may become an important tool for increasing the speed of the parameter estimation procedures.

4.4.4 Procedures

Table 4.6 provides a complete list of the procedures written for the PSM prototype. The table provides a short description of each procedure. For additional details, the interface descriptions supplied in appendix A.4 should be consulted.

Generally, importance has been given to ensure generic argument interfaces and breaking computational tasks into separate procedures. The generic interfaces contribute to the overall objective of creating a flexible program architecture, where future extensions to the source code may be instituted without requiring alterations of existing program entities.

Likewise, by splitting computational parts into independent building blocks it is made easy to analyze, compare or replace particular computational operations. For example, the calls to procedures for computing gradients numerically using the central difference method, namely `CNTDIFF_APL` and `CNTDIFF_AIAPLL`, may easily be replaced by procedures based on alternative approximation methods such as forward differencing.

Next, Section 4.5 illustrates the calling sequences by means of flowcharts.

Table 4.6: Summary of procedures in PSM.

Name	Purpose	Interface
`AIAPLL`	Compute approximate individual a posteriori log-likelihood $l_{p,i}$.	Table A.21
`ALLOC_KALMANOBJECT`	Allocate Kalman object.	Table A.22
`APL`	Compute approximate population log-likelihood l.	Table A.23
`APLDAPL`	Compute approximate population log-likelihood l and its gradient $dl/d\boldsymbol{\theta}$.	Table A.24
`CNTDIFF_AIAPLL`	Compute gradient of approximate individual a posteriori log-likelihood $dl_{p,i}/d\boldsymbol{\eta}_i$ using central difference scheme.	Table A.25
`CNTDIFF_APL`	Compute gradient of approximate population log-likelihood $dl/d\boldsymbol{\theta}$ using central difference scheme.	Table A.26
`DEXPM`	Double precision matrix exponentials $\boldsymbol{\Psi}_s = e^{\boldsymbol{A}\,\tau_s}$ and $\int_0^{\tau_s} e^{\boldsymbol{A}\,s}\boldsymbol{\Pi}_{\boldsymbol{w}}\boldsymbol{\Pi}_{\boldsymbol{w}}{}^T(e^{\boldsymbol{A}\,s})^T ds$.	Table A.27
`DISPLAY_RESULTS`	Display results of fixed-effects $\boldsymbol{\theta}$ and random-effects $\boldsymbol{\eta}_i$.	Table A.28
`ERRORSTAT`	Display error status.	Table A.29
`FAIAPLL`	Compute approximate individual a posteriori log-likelihood $l_{p,i}$ given individual log-likelihood l_i.	Table A.30
`FAPL`	Compute individual contribution to approximate population log-likelikelihood L.	Table A.31
`FIDNM`	Define identity matrix $\boldsymbol{I}$.	Table A.32
`FOMEGA`	Define random-effects covariance matrix $\boldsymbol{\Omega}$.	Table A.33
`FPHI`	Define individual parameters $\boldsymbol{\phi}_i$.	Table A.34
`FPI`	Define magnitude of system noise matrix $\boldsymbol{\Pi}_{\boldsymbol{w}}$.	Table A.35
`FSIGMA`	Define observation error covariance $\boldsymbol{\Sigma}$.	Table A.36
`FTHETA`	Define fixed-effects $\boldsymbol{\theta}$.	Table A.37
`HESSIAN_AIAPLL`	Compute Hessian of approximate individual a posteriori log-likelihood $\boldsymbol{\Delta} l_{p,i}$.	Table A.38
`INIT_ETAOBJECT`	Initialize random-effects data object `ETAOBJECT`.	Table A.39
`INIT_THETAOBJECT`	Initialize fixed-effects data object `THETAOBJECT`.	Table A.40
`LINEAR_MODEL`	Define coefficient matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ in the LTI and LTV state-space models.	Table A.41
`LLDLL`	Compute individual log-likelihood l_i and its gradient $dl_i/d\boldsymbol{\eta}_i$.	Table A.42
`LTI_KALMAN`	Kalman filtering procedure for linear time-invariant (LTI) model.	Table A.43
`LTV_KALMAN`	Kalman filtering procedure for linear time-varying (LTV) model.	Table A.44
`READDAT`	Reads data given three datafiles (dose and covariates datafiles are optional).	Table A.45

4.5 Flowcharts

This section concludes the presentation of the design and structure of the PSM prototype by illustrating the calling sequences in the PSM prototype using flowcharts.

It has been chosen split the entire flowchart of the PSM prototype into two separate the flowcharts, namely:

1. A flowchart for the program `MAIN`.
2. A flowchart for the parameter estimation algorithm starting in the approximate population procedure `APL`.

This way, it is possible to draw attentention to details in the structure of `MAIN`. Oppositely, and because of inherent complexity of the calling sequences of `APL`, details has been extracted from the flowchart of the parameter estimation algorithm leaving room for the procedure names only.

As mentioned previously, better understanding of the program may be achieved by keeping track of the procedures' interface descriptions supplied in the appendix A.4 while examining the flowcharts.

Flowchart of program `MAIN`

Figure 4.2 shows the flowchart[6] of the program `MAIN`. The algorithm is briefly commented here:

1. Initiating PSM, the `READDAT` acquires data from input datafiles and then allocates data in a N-dimensional vector of the type `PATIENT`, where N is the number of patients in the model. Datafiles are specified by the user during run-time.
2. Next, the a single `THETAOBJECT` and a N-dimensional vector of the type `ETAOBJECT` are initialized. Initialization of `ETAOBJECT` is carried out so that the patient represented in first element in the `ETAOBJECT`-vector is the same as in the `PATIENT`-vector.
3. Thirdly, the parameter estimation algorithm is invoked by calling one of the following procedures, i.e.:
   - `APL` procedure for estimating fixed- and random-effects
   - `AIAPLL` procedure for estimating random-effects only

[6] A list of flowchart symbols in supplied in appendix C.

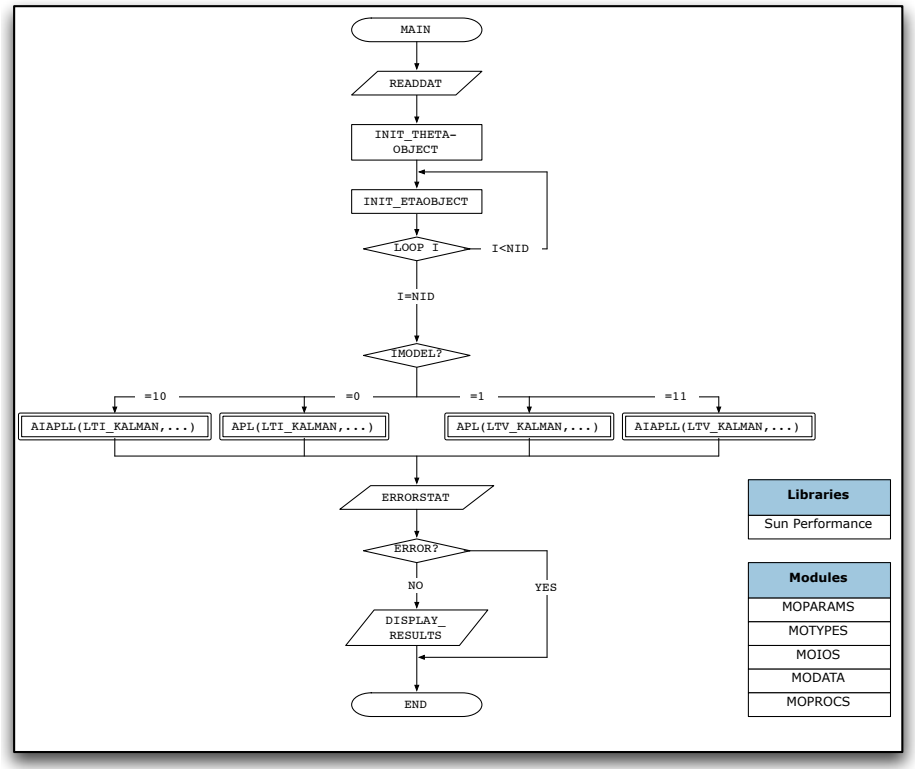

Figure 4.2: Flowchart for `PROGRAM MAIN`. Names stated in textboxes identifies the subroutines called by the program.

The model type, namely linear time-invariant or linear time-varying, is specified in the argument list by indicating the name of the Kalman filtering procedure to be used, i.e. `LTI_KALMAN` or `LTV_KALMAN`.

The model decision is based on the model specification parameter `IMODEL`, which is defined in the model declaration file `PARAMS`.

4. Next step in the algorithm of `PROGRAM MAIN` is calling the error message procedure `ERRORSTAT`. Error warnings are provided in Appendix B.
5. Finally, if no errors have been encountered, the results of the fixed-effects $\boldsymbol{\theta}$ and the individual random-effects $\boldsymbol{\eta}_i$ are displayed by calling `DISPLAY_RESULTS`.

An important characteristics of the model design appears in part 3), which shows that the model selection takes place on a "top-level" of the algorithm. The benefit of this concept is that lower-level `IF`-constructs at each optimization step vanishes. This potentially implies relativey higher computational speeds, particularly when the number of subjects in the model increases.

As implied in the PROGRAM MAIN flowchart in Figure 4.2, the model allows single subject modeling using the AIAPLL procedure. This temporary design has been chosen, since the program currently only has been validated for AIAPLL, which computes the individual random-effects.

The remainder of this section shortly summarizes the approximate population log-likelihood procedure APL for estimating the fixed- and random-effects parameters in the population model.

Flowchart of procedure APL

Figures 4.4 and 4.5 show the flowcharts for APL for the linear time-invariant (LTI) and linear time-variant (LTV) case, respectively. The only distinction between the two flowcharts lies within which Kalman filtering procedure is called, namely LTI_KALMAN and LTV_KALMAN.

As emphasized in the description of the PROGRAM MAIN flowchart, the Kalman filtering scheme is selected on the top-level of the algorithm. This is possible since the LTI_KALMAN and LTV_KALMAN procedure share the same generic interface. The argument list of the LTV_KALMAN procedure is illustrated in Figure 4.3 and is identical to that of LTI_KALMAN procedure, see Tables A.44 and A.43, respectively.

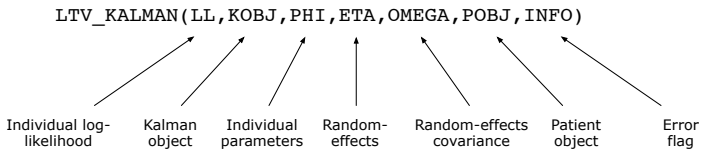

Figure 4.3: Argument interface for LTV KALMAN procedure.

Since model specification entities, for example coefficient matrices, do not appear in the argument lists, the prototype can easily be extented to handle non-linear models by implementation of a non-linear Kalman filtering procedure based on an identical interface.

The following paragraphs briefly describes the features of the algorithm for identifying the parameters in the population model. Again, additional information on the individual procedures is found in Appendix A.4.

The APL procedure identifies the optimal fixed- and random-effects using the approximate population log-likelihood as objective function for the optimization procedure. Both the fixed- and random-effects are found by means of the unconstrained non-linear

minimization package *ucminf*, which requires evaluation of the Cholesky decomposition (`SPCHOL` and `CHKDFN`) and soft line search (`SLINE`).

The parameter optimization procedure for the approximate population log-likelihood `UCMINF_APL` takes `APLDAPL` as argument. `APLDAPL` computes the approximate population likelihood, its gradient with respect to the fixed-effects parameters and the Hessian appoximation of the individual log-likelihood by looping over all individuals in the model.

`AIAPLL` is called by `APLDAPL` and computes the approximate individual a posteriori log-likelihood. For a particular set of fixed-effects parameters, the `AIAPLL` procedure identifies random-effects. The individual log-likelihood function is the objective function. The individual log-likelihood is computed in the Kalman filtering procedure and returned to the `LLDLL` subroutine, which also computes the gradient of the individual log-likelihood with respect to the random-effects. `LLDLL` is called by `UCMINF_AIAPLL`.

Gradients of both the approximate population log-likelihood (`CNTDIFF_APL`) and the individual log-likelihood (`CNTDIFF_AIAPLL`) are approximated numerically using a central difference method.

The linear Kalman filtering procedures `LTI_KALMAN` and `LTV_KALMAN` call the same procedures, although several more evalutations are required for time-varying models. The matrix exponentials are computes by means of *expokit* that uses a Padé approximation with repeated scalin and squaring.

At each evaluation of the approximate population likelihood, the fixed-effects are stored as a column vector in the `THETA`-array in `THETAOBJECT`. Similarly, historic estimates of the random-effects are stored in `ETA`-array in `ETAOBJECT`, when the approximate individual a posteriori log-likelihood is determined. Since dynamic re-allocation of arrays is not possible in Fortran, the array size is defined explicitly by the integer parameter `IMAX=200` defined in the module `MOTYPES`.

This chapter presented the overall file structure and design of the PSM prototype. Chapter 5 describes important aspects of the model implementation as well as validation of individual parts of the programs. As mentioned with regard to the thesis objectives in Section 4.1, the approximate population log-likelihood procedure of estimating the remains to be validated.

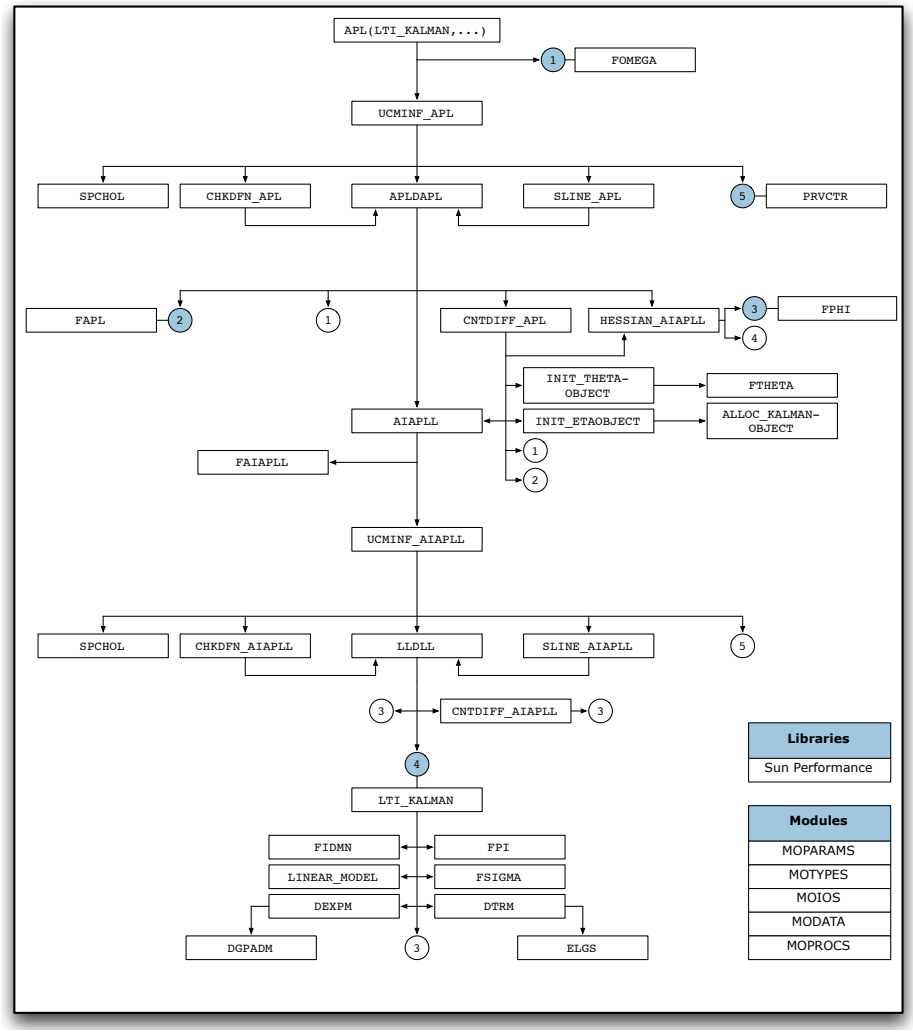

Figure 4.4: Flowchart of the approximate population log-likelihood procedure `APL` based on the LTI Kalman filtering. Colored circles defines the sequence of a particular number.

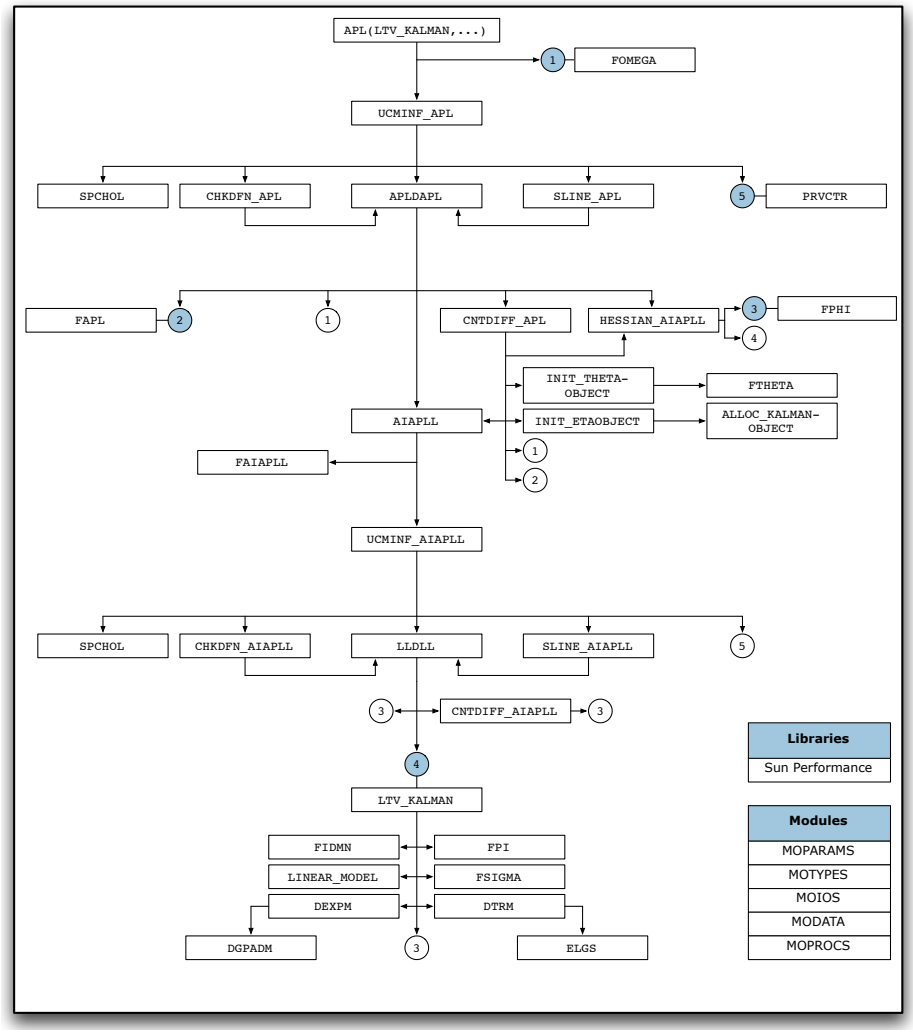

Figure 4.5: Flowchart of the approximate population log-likelihood procedure `APL` based on the LTV Kalman filtering. Colored circles defines the sequence of a particular number.

Chapter 5

Implementation and Validation

Chapter 3 presented the theory of population PK/PD modeling based on non-linear mixed-effects (NLME) models using stochastic differential equations (SDEs). Chapter 4 presented the design and composition of the PSM prototype with particular attention to the underlying design concepts. Succeeding the formal introduction, this chapter aims at highlighting particular implementation details.

The ambition of this thesis was to propose the first steps towards the construction of a next generation software for population PK/PD modeling. Given the size and complexity of the model, implementation was prioritized over validation. Section 5.1 presents selected validation results for the PSM prototype and, importantly, points at the parts of the PSM algorithm that remains to be validated.

Section 5.2 briefly discusses aspects related to the implementation of the Kalman filtering procedures. Features of the parameter estimation procedures used in this work is the topic of Section 5.3. Finally, data protection issues and optimization considerations are presented in Sections 5.4 and 5.5, respectively.

5.1 Model validation

Due to time constraints, only the individual-level optimization of the PSM prototype has been validated. Validation of individual-level optimization has been accomplished by comparison of results from PSM and CTSM.

The individual-level modeling comprises the Kalman filtering procedures `LTI_KALMAN` and `LTV_KALMAN`; the procedure for evaluating the approximate individual log-likelihood and its gradient `LLDLL`; and the optimization procedure `UCMINF_AIAPLL`.

Tables 5.1 and 5.2 reports the objective function values obtained from linear time-invariant (LTI) and a linear time-varying (LTV) Kalman filtering, respectively.

Table 5.1: Objective function for linear time-invariant (LTI) model obtained from CTSM and PSM Fortran 95 prototype.

Model	Objective function
CTSM	–623.3686
PSM Fortran 95	–623.3564

For LTI Kalman filtering, the objective function value found in PSM ($l_1 = -623.3686$) deviates from its CTSM counterpart ($l_1 = -623.3564$) on the fifth significant figure. The result is considered acceptable and `LTI_KALMAN` valid.

For LTV Kalman filtering, the objective function value found in PSM ($l_1 = 9.357946$) deviates from its CTSM counterpart ($l_1 = 9.392968$) on the third significant figure.

Table 5.2: Objective function values for linear time-varying (LTV) model obtained from CTSM, PSM Fortran 95 and PSM Matlab® prototypes.

Model	Objective function
CTSM	9.392968
PSM Fortran 95	9.357946
PSM Matlab®	9.357946

In order to support the validity of the LTV Kalman filtering procedure `LTV_KALMAN`, a LTV Kalman filtering procedure was constructed in Matlab® based on the PSM prototype presented by Mortensen and Klim [10]. Here, the objective function ($l_1 = 9.357946$) was found identical to that of PSM. Based on this results `LTV_KALMAN` is considered valid, although additional analysis is recommended in later work.

Table 5.3 summarizes the estimates of the individual parameters $\boldsymbol{\phi}$ obtained from the approximate individual log-likelihood optimization procedure `UCMINF_AIAPLL` using a LTI Kalman filtering scheme. It appears that the PSM and CTSM results are approximately identical. Both `UCMINF_AIAPLL` and `LLDLL` are consequently found valid.

Table 5.3: Estimates of individual parameters $\boldsymbol{\phi}$ based on optimization of individual log-likelihood for a linear time-invariant (LTI) model. Results from CTSM and PSM Fortran 95.

	PSM	**CTSM**
$\boldsymbol{\phi}(1)$	1.3130E+01	1.3134E+01
$\boldsymbol{\phi}(2)$	2.5330E+01	2.5330E+01
$\boldsymbol{\phi}(3)$	1.0397E+02	1.0394E+02
$\boldsymbol{\phi}(4)$	9.6462E–01	9.6509E–01
$\boldsymbol{\phi}(5)$	2.0219E+00	2.0215E+00
$\boldsymbol{\phi}(6)$	4.9323E+01	4.9320E+01
$\boldsymbol{\phi}(7)$	5.0930E–01	5.0929E–01
$\boldsymbol{\phi}(8)$	1.0338E–02	1.0330E–02

As mentioned in Chapter 4 and illustrated in the flowcharts in Figure 4.4 and 4.5, the choice of Kalman filtering procedure is passed from the `PROGRAM MAIN` to `LLDLL`. Since both the LTI and LTV Kalman filtering procedures are called from `LLDLL`, it follows that the validity of the parameter optimization procedure `UCMINF_AIAPLL` is independent of the type of Kalman filtering performed.

Secondly, it is noted that although the population-level modeling has yet not been validated, both the approximate population log-likelihood optimization procedure `UCMINF_APL` and the procedure for evaluating the approximate population log-likelihood (objective function) and its gradient `APLDAPL` are based on identical principles.

This concludes the validation part. In the next section, the implementation of the Kalman filtering procedures is commented.

5.2 Implementation of Kalman filter

Two ordinary Kalman filtering procedures have been defined in accordance with the mathematical theory presented in Section 3.4, namely an LTI Kalman filtering procedure `LTI_KALMAN` and an LTV Kalman filtering procedure `LTV_KALMAN`, respectively.

The PSM prototype takes in the *expokit* by Sidje [14] for numerical computation of matrix exponentials. *Expokit* uses a Padé approximation with repeated scaling and

squaring. Matrix determinants are determined by means of the procedures `DTRM` and `ELGS`, which uses a partial-pivoting Gaussian elimination scheme[1].

The Kalman filtering procedure handles both singular and non-singular state coefficient matrix $\boldsymbol{A}$. Furthermore, the optimized Sun Performance Library procedure are used extensively for numerical manipulations. The interface descriptions are given in Tables A.43 and A.44.

Next, the parameter optimization is discussed thoroughly.

5.3 Parameter optimization

An unconstrained, non-linear quasi-Newton method based on a Broyden-Fletcher-Goldfarb-Shanno (BFGS) updating scheme is employed for estimating the parameters in PSM. For this purpose, the software package *ucminf* has been implemented.

The optimal parameter estimate $\boldsymbol{x}^*$ is found via minimization of the scalar *objective function* $F(\boldsymbol{x})$:

$$\boldsymbol{x}^* \quad = \quad \arg\min_{\boldsymbol{x}\in\mathcal{D}}\{F(\boldsymbol{x})\} \tag{5.1}$$

where $F: \ \mathbb{R}^n \rightarrow \mathbb{R}$ is a given, continously differentiable function and $\mathcal{D}$ defines the proximity of $\boldsymbol{x}^*$.

Given an initial guess for the parameters, the Newton's method iteratively uses a second-order Taylor expansion of the objective function to find a new parameter estimate. The Newton's method ensures quadratic convergence around a (local) minimum, but requires evaluation of the gradient vector and the corresponding Hessian matrix.

Evaluating the Hessian matrix may be computational laborious, when not expressed analytically. The quasi-Newton's method gradually constructs a numerical approximation of the Hessian based on previous parameter estimates and, thus, reduces the computational workload of each iteration step.

Finally, it has been reported that the BFGS method with soft line search, in general, provides better convergence results compared to other updating methods [5].

[1] `http://www.physics.unlv.edu/~pang/comp3/code42.f90`

5.3.1 Fortran minimizer *ucminf*

In this work, estimation of both fixed- and random-effects parameters is performed using the software package *ucminf*, which is a quasi-Newton method using BFGS update, line search and trust-regions. For detailed introduction to *ucminf*, which is written in FORTRAN 77, see Nielsen [11].

The *ucminf* package consists of five subroutines, namely the unconstrained, non-linear optimization procedure `UCMINF`; the Cholesky decomposition procedures `SPCHOL` and `CHKDFN`; the line search algorithm `SLINE`; and the procedure for outputting information about individual optimization steps `PRVCTR`.

The choice of deploying *ucminf* is particularly motivated by

1. Quadratic convergence near objective function minimum
2. Robustness of the minimization algorithm
3. Available and modifiable source code
4. Customizable minimization conditions

When called, `UCMINF` allows specification of the initial (inverse positive-definite) Hessian matrix. On exit, the final Hessian approximation is returned to the calling procedure. In future work, it would be interesting to investigate whether enhanced computational speeds for the optimization of the individual random-effects may be achieved by providing the final Hessian approximation obtained in a previous step as initial guess of the Hessian in the succeeding optimization step. Possibly, some criteria on, for example, the fixed-effects parameters may be required. Past optimization information is currently not exploited in PSM.

The optimal set of parameters are found by minimization of the objective function. Following the general notation defined in the beginning of Section 5.3, the parameter estimation procedure is subjected to three stopping criteria:

$$\begin{aligned} \left\| dF(\boldsymbol{x})/d\boldsymbol{x} \right\|_\infty &< c_1 && (5.2)\\ \left\| \Delta\boldsymbol{x} \right\|_2 &< c_2(c_2 + \left\| \boldsymbol{x} \right\|_2) && (5.3)\\ \nu &\geq \nu_{max} && (5.4) \end{aligned}$$

where $(c_1, c_2) \in \mathbb{R}^2$ are arbitrary constants; and $\nu_{max} \in \mathbb{N}\backslash\{0\}$ is the maximum number of iterations allowed.

Application of *ucminf* required some general modifications to the *ucminf* procedures.

First, the hierarchic organization of the PSM prototype, where fixed- and random-effects are estimated in multiple steps, made it necessary to expand the argument list of *ucminf* procedures in order pass information between the individual-level and population-level modeling. Consequently, `MODEL`, `THETAOBJ`, `ETAOBJ`, `POBJ`, `OMEGA` and `INFO` was added to the argument list and the generic procedure interface for the Kalman filtering procedure (`MODEL`) was defined.

Since FORTRAN 77 does not support derived types, it was, secondly, necessary to update the entire *ucminf* package from FORTRAN 77 to Fortran 95. Validation of the Fortran 95 *ucminf* was accomplished using a simple function for which the gradient could be determined analytically.

Thirdly, with the goal of improving data protection, aidin future parallelization and potentially customize the population-level and/or individual-level optimization algorithm, two copies of the *ucminf* procedures were created.

The resulting individual-level and population-level minimizers are summarized in Table 5.4.

Table 5.4: Individual- and population-level miminization procedures.

Individual-level	Population-level
`UCMINF_AIAPLL`	`UCMINF_APL`
`CHKDFN_AIAPLL`	`CHKDFN_APL`
`SLINE_AIAPLL`	`SLINE_APL`

Sections 5.3.3 and 5.3.2 review the individual settings for the individual-level and population-level minimizers based on *ucminf*. Stop criteria are based on the recommendations of Mortensen and Klim [10].

5.3.2 Individual-level optimization

Individual-level optimization identifies the optimal individual random-effects parameters $\hat{\boldsymbol{\eta}}_i$. The dimension of the parameter space is defined by the number of random-effects parameters.

The gradient of the individual log-likelihood $dl_i/d\boldsymbol{\eta}_i$ is computed using the central difference method. The number of function evaluations required by a central difference scheme is almost two-fold higher than that of the corresponding forward or backward difference method. However, experience with *ucminf* suggests that the additional computation time required by the central difference scheme is likely to regained through a more accurate gradient approximation and, which then improves the iterative procedure for parameter optimization.

The initial guess is defined by $\boldsymbol{\eta} = \mathbf{0}$ and the stopping criteria

$$\begin{aligned} \|dl_i/d\boldsymbol{\eta}_i\|_\infty &< c_1 && (5.5)\\ \|\Delta\boldsymbol{\eta}_i\|_2 &< c_2(c_2 + \|\boldsymbol{\eta}_i\|_2) && (5.6)\\ \nu &\geq \nu_{max} && (5.7) \end{aligned}$$

where both c_1 and c_2 are set to 10^{-5}.

As mentioned previously, *ucminf* allows definition of the initial guess for the inverse, positive-definite Hessian matrix $\boldsymbol{\Delta}_0$ in it argument list. If no guess is given, the algorithm takes an identity matrix as the initial guess of the inverse Hessian $\boldsymbol{\Delta}_0 = \mathbf{I}$ and requries a minimum of $s/2$ iterations, where s is the number random-effects parameters, before the approximation to the inverse Hessian is complete.
The choice of *ucminf* was, besides the fact that it is a robust open-source optimizer, motivated by the possibility of aiding the optimization process by exploiting previous optimization results. Given an appropriate criterion, it would be interesting to evaluate the consequences of supplying optimum conditions for the random-effects obtained the preceding step as initial guess at the following optimization step.

5.3.3 Population-level optimization

Population-level optimization estimates the optimal fixed-effects parameters $\hat{\boldsymbol{\theta}}$. The dimension of the parameter space is in this case defined by the number of fixed-effects parameters.

As for the individual-level modeling, the gradient of the approximate population log-likelihood $dl/d\boldsymbol{\theta}$ derived by means of the central difference method.

The initial guess $\boldsymbol{\theta}_0$ is defined by the user through modification of the model declaration file `THETAVEC`, see also Section 6.1 in Chapter 6. The stopping criteria for the estimating the fixed-effects are

$$\begin{aligned} \|dl/d\boldsymbol{\theta}\|_\infty &< c_1 && (5.8)\\ \|\Delta\boldsymbol{\theta}\|_2 &< c_2(c_2 + \|\boldsymbol{\theta}\|_2) && (5.9)\\ \nu &\geq \nu_{max} && (5.10) \end{aligned}$$

where both c_1 and c_2 are set to 10^{-4}.

As indicated in Section 5.1, the population-level modeling awaits validation. Based on experience with validation of the individual-level part of the PSM prototype, it is

strongly recommended that careful attention is directed towards the iteration count variable in the fixed-effects data object. It should be ensured that evaluation of the gradients and Hessians does not increase the count variable, since these components are essentially *part of* each iteration step. This aspect did create minor difficulties during validation of the individual-level part of the PSM program, thus this warning.

5.4 Data protection

The `INTENT`-attribute in Fortran 95 is – as is good programming practice – used in the declaration of all procedure arguments. For a given argument variable, it allows specification of `INTENT(IN)`, `INTENT(OUT)` or `INTENT(INOUT)`, indicating whether the argument variable remains constant, is created or changed within a procedure. Expections to the use of the `INTENT`-attribute are found in external procedures only, namely the matrix exponential package *expokit* and the parameter optimization package *ucminf*.

As mentioned in Section 4.4.1, the derived type `PATIENT` only contains raw data. Following data acquisition in `READDAT`, the derived type `PATIENT` is strictly defined using the `INTENT(IN)`. This protects raw data from being altered or deleted.

Modules are accessed through `USE`-association, which makes the available all module definitions, data and procedures. Furthermore, the '`USE, ONLY:`' limits the range of information visible to the `USE`-associated program unit. It has been used extensively in order secure highest degree of data protection.

Finally, compilations via 'make' are performed with compiler flag `-C`, which examines array references for potential subscript violations and conformance.

Next, Section 5.5 terminates this chapter with comments on optimization issues and considerations.

5.5 Optimization considerations

Standard linear algebraic computations comprise the majority of the numerical manipulations in PSM. The optimized procedures made available by the Sun Performance Library have been applied (almost) whenever possible.

The computations are predominantly carried out in the Kalman filtering procedures. In comparison to CTSM, PSM contains separate Kalman filter procedures for the LTI and LTV models, respectively. Thereby, `IF`-statements related to the identification of the type of filtering are completely avoided.

Furthermore, the design of the "high-level" selection of the Kalman filterering procedure expectedly contributes to enhanced computational speed. This was illustrated in Figures 4.2–4.4, where the Kalman filtering procedure is transmitted from `PROGRAM MAIN` to `LLDLL`.

Due to time constraints, neither manual tuning of the source code nor parallization of the computational tasks have been set up. However, compiler optimization of source code is achieved by invoking the flag '`-fast`' during compilation with 'make'.

This completes the description of validation and implementation issues. The final piece of information related to the introduction of the PSM prototype in presented in Chapter 6. Therein, the input/output interface is described.

Chapter 6

Input/Output Interface

In the preceding chapters, the theoretical background, the design and implementation of the PSM prototype has been presented. This chapter concludes the presentation of the prototype with a description of the steps involved in setting up and building a model in PSM.

Declaring a new model is accomplished in three major steps, namely:

1. Preparation of datafiles
2. Modification of model declaration files
3. Building model using *make*

The following sections go through each step with attention to both the flexibility and limitation of the current model setup.

6.1 Datafiles

At Novo Nordisk A/S, experience with clinical trials encourages splitting (1) input and output data, (2) dose information data and (3) covariate data into separate datafiles.

The arrangement of three separate datafiles is expected to address the general organisation of clinical data in databases. This contrasts with the data formats in NONMEM®, where inputs, outputs, dose information and covariate are contained in a single input file.

The following points must be fulfilled for proper data handeling:

- Each patients must supplied with a unique identifier represented by a character string of maximum eight letters and/or numbers, for example '`PAT001`'.
- Unique patient identifier must be stated in first column of all datafiles.
- Data must be comma-delimited.
- Dimensions of the datafiles should comply with the specifications in the model declaration files `PARAMS` and `READSPEC`, see also the summary of global parameters in Table 4.4.

The following paragraphs describe how each datafile should be prepared for correct data input.

Input and output datafiles

Figure 6.1 illustrates an inputs/outputs datafile, in which observations for two subjects (`NID = 2`). The model contain a single output variable (`NY = 1`) and two input variables (`NU = 2`).

In brief, the unique patient identifier is stated in the left-most column in each line of the datafile. The prototype allows individual number of observations. The prototype handles both existing (`NU > 0`) and non-existing inputs (`NU = 0`).

The number of lines (`NROWS_DATA`)in the inputs/outputs datafiles must be supplied in the file `READDAT` as a part of the model declaration.

Dose datafiles

The general format of the dose datafiles is defined by means of an example illustrated in Figure 6.1. Particular attention should be given to the data sequence.

Definition of the *pre-/post-sampling* term, also named the dose identifier, and *method of administration* is provided in the interface description of the derived type `DOSE`, see Table A.15 on page 88.

The PSM prototype provides a flexible environment for handling dose data. Firstly, an individual number of doses is allowed. Secondly, administration of doses to only

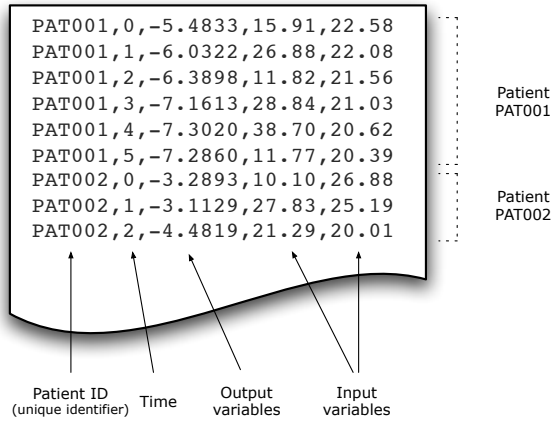

Figure 6.1: Illustration of datafile containing input and output variables.

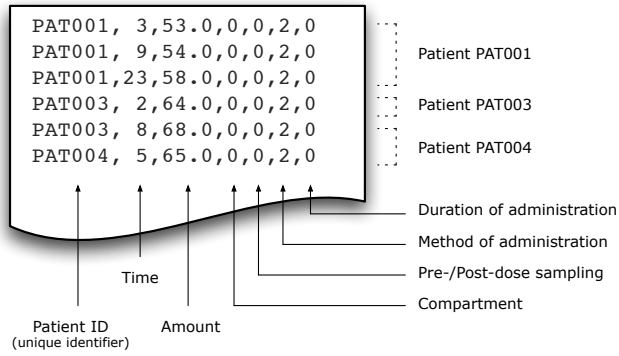

Figure 6.2: Illustration of datafile containing dose information.

a subgroup of patients within a population is allowed. In Figure 6.1, for example, patient `PAT001` has received three, patient `PAT002` zero and patient `PAT003` two doses. Furthermore, doses are given on different times.

Just as for inputs/outputs datafiles, the total number of lines (`NROWS_DOSE`)in the dose datafiles must be supplied in the file `READDAT` as a part of the model declaration.

Covariate datafiles

As in the preceding paragraphs, the sequence of data appearance in covariate datafiles is defined by an illustration, see Figure 6.3.

If present, the covariate file must contain covariate information for all individual. The number of covariates (`NZ`) is defined in the model declaration file `PARAMS`. In Figure 6.3 three covariates are defined for each patient.

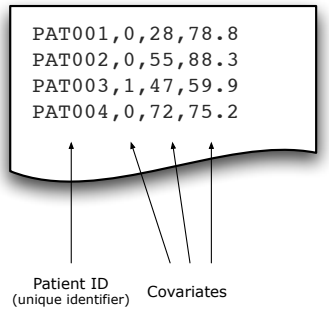

Figure 6.3: Illustration of datafile containing covariates.

An important shortcoming of the current design is that all covariate information on input is defined as the numeric data type `REAL`.

The PSM prototype automatically formats of data according to the dimensions of the individual parameters. As mentioned previously, however, both the number of patients in the model and the dimensions of the inputs/outputs and dose datafiles must be supplied during model declaration. This setup was chosen for simplification.

In order to accommodate flexible transition from NONMEM to PSM, it is recommendable to develop a data filter procedure that is able to translate standard NONMEM formats into PSM.

6.2 Model declaration

A complete model declaration requires specification of a total of 13 model declaration files that must comply with standard Fortran 95 syntax. The model declaration files are uniquely located in the '`/USER`' directory as illustrated in Figure 6.4.

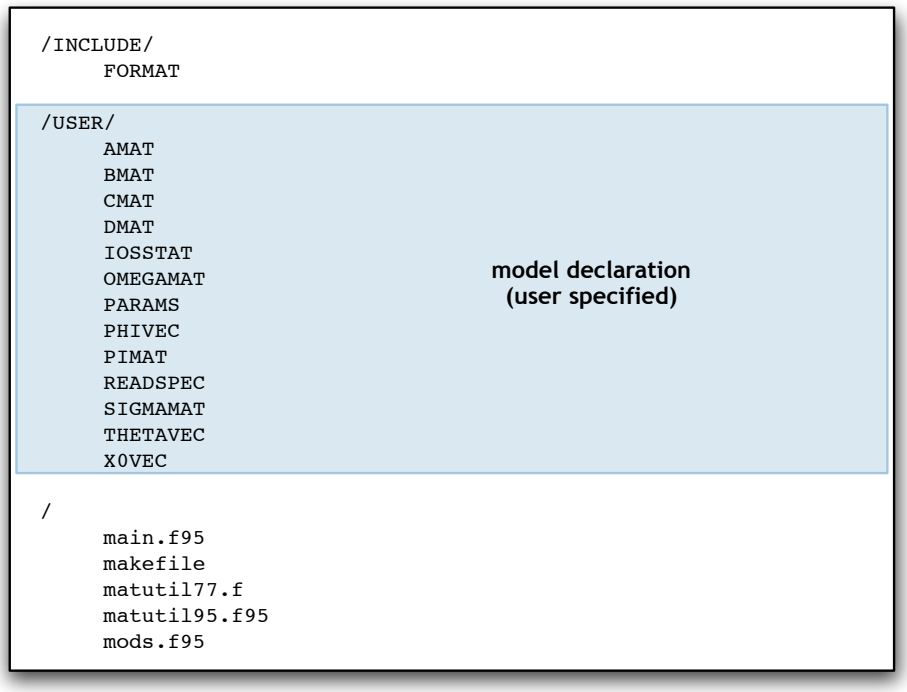

Figure 6.4: File tree structure with model declaraction part hightlighted.

Table 6.1 briefly summarizes the purpose of each model declaration file and supplies reference to the interface descriptions located in Appendix A.2.

Given the extensive interface description for each model declaration file, setting up models are rather straigth forward. Definitions of vectors and matrices must comply with standard Fortran 95 syntax and are defined element-wise.

As indicated in the interface description of the model declaration file `PARAMS`, the PSM prototype allows both single-subject and population modeling through specification of the global parameter `IMODEL`. The choice single-subject modeling is given to allow modeling of the validated part of the program.

Next, Section 6.3 completes the description of the declaring and building models with PSM.

Table 6.1: Summary of model declaration files, including references to the file interfaces supplied in the appendix A.2.

Name	Description	Interface	Ref. Eqns.
`IOSTAT`	Defines input/output control.	Table A.2	–
`AMAT`	Defines coefficient matrix $\boldsymbol{A}$ in LTI and LTV state equations.	Table A.3	(3.6), (3.4)
`BMAT`	Defines coefficient matrix $\boldsymbol{B}$ in LTI and LTV state equations.	Table A.4	(3.6), (3.4)
`CMAT`	Defines coefficient matrix $\boldsymbol{C}$ in LTI and LTV state equations.	Table A.5	(3.7), (3.5)
`DMAT`	Defines coefficient matrix $\boldsymbol{D}$ in LTI and LTV state equations.	Table A.6	(3.7), (3.5)
`OMEGAMAT`	Defines variance-covariance matrix $\boldsymbol{\Omega}$ for the random-effects $\boldsymbol{\eta}_i$.	Table A.7	(3.34)
`PARAMS`	Defines global parameters and selects type of model, LTI or LTV.	Table A.8	–
`PHIVEC`	Defines individual parameter vector $\boldsymbol{\phi}$.	Table A.9	(3.34)
`PIMAT`	Defines magnitude of system noise matrix $\boldsymbol{\Pi}_{\boldsymbol{w}}$ in LTI and LTV state equations.	Table A.10	(3.6), (3.4)
`READSPEC`	Defines number of patients in model, N, and indicates number of rows in input/output and dosing datafiles.	Table A.11	–
`SIGMAMAT`	Defines variance-covariance matrix $\boldsymbol{\Sigma}$ for the measurement error $\boldsymbol{e}$ in observation equations.	Table A.12	(3.7), (3.5)
`THETAVEC`	Defines the initial guess of the fixed-effects parameters $\boldsymbol{\theta}$.	Table A.13	(3.34)
`XOVEC`	Defines the initial guess of the state variables $\hat{\boldsymbol{x}}_{i(1\|0)}$.	Table A.14	(3.8)

6.3 Building the model

Having completed the model declaration step by proper manipulation of the model declaration files discussed in the preceding paragraph, the final step necessary for running a particular model is to build the PSM prototype.

The PSM prototype takes advantages for the *make* utility for building the prototype. Having specified the dependencies for each target, *make* generates the executable `./psm` file.

The model building is accomplished in following steps:

1. Complete modification of all model declaration files located in the `'/USER'` directory, see Section 6.2.
2. Verify that the formats of the inputs/outputs datafile, the dose information datafile and covariates datafile are congruent with the specifications described in Section 6.1.
3. Place the datafiles in main directory (`'/'`).
4. Initialize the latest Sun Studio Express `f95` (on the DTU G-bar run 'init.ssxp').
5. Run 'make realclean', then 'make'.
6. After successful compilation, run PSM using the command './psm' and enter the datafile names in the mentioned order.

The command '`make realclean`' is required, since modification to include files, that is, the model declaration files, are otherwise not evaluated.

6.4 Displaying results

A minimum of efforts has been put into the output features of PSM. Effectively, it constitutes an entire new project to analyze optimal data rendering methods.

Prior to displaying results the error message flag `INFO` supplied to the procedure `ERRORSTAT`. See Table B for a complete list of error messages returned by `ERRORSTAT`. If no errors are present, the program proceeds to displaying the results.

Currently, output control is defined by the parameter `IOVAR` in the model declaration file `IOSSTAT`.

`IOVAR = 0:` final results of the fixed-effects parameters $\boldsymbol{\theta}$ and the random-effects parameters $\boldsymbol{\eta}_i$ are outputted to the screen.

`IOVAR = 1:` intermediate and final results of the fixed-effects parameters $\boldsymbol{\theta}$ and the random-effects parameters $\boldsymbol{\eta}_i$ are outputted to the screen.

Results the following results are display on call to `DISPLAY_RESULTS`. From the population-level optimization is given for each patient:

1. Fixed-effects parameters $\boldsymbol{\theta}$
2. Approximate population log-likelihood l
3. Number of iteration step

From the individual-level optimization is given:

1. Random-effects parameters $\boldsymbol{\eta}_i$
2. Approximate individual a posteriori log-likelihood $l_{p,i}$
3. Approximate individual log-likelihood l_i
4. Number of iteration step

In the case of `IOVAR = 0`, the iteration step indicates the final and total number of iterations performed for both the population-level or individual-level optimizations.

This concludes the presentation of input/output features of the PSM prototype. Chapter 7 discusses particular aspects of the implementation and brings forward a set of recommendations for future work.

Chapter 7

Discussion and Recommendations

In accordance with the thesis objectives, the construction of a prototype for population PK/PD modeling based on non-linear mixed-effects (NLME) models using stochastic differential equations (SDEs) has been achieved. In this chapter, issues related to the current model implementation are discussed. The discussion gives rise to a set of recommendations for future work, which are summarized in the end.

7.1 Discussion

The programming language of choice is Fortran 95. This choice was supported by the availability of Sun Performance Library (optimized) as well as external software packages for computing matrix exponentials and parameters optimization. Fortran 95, furthermore, supports OpenMP shared-library multiprocessing API for parallel computing.

Due to the size and relative complex nature of the population PK/PD algorithm, the implementation was accomplished on the cost of relatively limited validation measures. Consequently, only the individual-level modeling part of the PSM prototype has been validated. Validation was performed by comparing results obtained in PSM with corresponding outcomes derived in CTSM, which is a program for single-subject PK/PD modeling based on NLME models using SDEs.

A minor discrepancy between the PSM prototype and CTSM in the evaluation of the approximate individual log-likelihood for the linear time-varying (LTV) case was identified. A later implementation of a LTV Kalman filtering procedure in Matlab® based on the validated Kalman filtering routine for linear time-invariant (LTI) models proposed by Mortensen and Klim [10] supported the function value obtained in PSM. However, additional validation efforts are recommended.

At present, single-subject modeling has been made available in the model declaration, thereby, creating a simple alternative to CTSM. This addition to the capacity of the PSM prototype is motivated by the following reasons: firstly, it allows immediate use of the validated part of the program; and secondly, it is instituted easily by replacing in the main program the call to the approximate population likelihood procedure (`APL`) by a call to the approximate individual a posteriori log-likelihood procedure (`AIAPLL`). No modifications to the source code are required.

The next step with respect to the model development is to validate the population-level modeling part for estimating the fixed-effects parameters. This move should potentially be accompanied by the implementation of a mapping function with the purpose of creating bound for the fixed-effects parameters. Mortensen and Klim [10] recommended an inverse tangent (arc-tangent) mapping function $f_a(X_k)$ of the type $\bar{X}_k = f_a(X_k),\ \ f_a : \mathbb{R} \rightarrow [X_k^{\min}; X_k^{\max}]$, i.e.

$$\bar{X}_k = f_a(X_k) = \frac{\arctan(X_k) + \pi/2}{\pi}(X_k^{\max} - X_k^{\min}) + X_k^{\min} \tag{7.1}$$

where k is the vector index; X_k is the kth original parameter; $\bar{X}_k$ is the kth mapped parameter; and $(X_k^{\min}, X_k^{\max}) \in \mathbb{R}^2$ are the corresponding lower and upper bounds, respectively. It follows from equation (7.1) that $X_k^{\min} < \bar{X}_k < X_k^{\max}$. Compared with a logistic mapping function, the inverse tangent mapping function results in smaller gradients $d\bar{X}_k/dX_k$, which holds the advantage of providing relatively more moderate changes in the mapped parameter during parameter optimization.

Mortensen and Klim [10] did not find it necessary to map the random-effects parameters in the PSM Matlab® prototype. With the introduction of single-subject modeling in the PSM Fortran 95 prototype, however, the need of mapping the individual parameters may rise. In CTSM, mapping of the individual parameters is accomplished by means of a logarithmic mapping function f_l so that

$$\bar{X}_k = f_l(X_k) = \ln\left(\frac{X_k - X_k^{\min}}{X_k^{\max} - X_k}\right) \tag{7.2}$$

where the notation is unchanged. In comparison to CTSM, experience with the PSM prototype indicates that it is relatively more fragile towards abrupt evolution in the parameters. Therefore, as long as the population-part of the PSM algorithm remains to be validated, it is recommended to institute mapping of the random-effects parameters.

In this thesis, estimation of the fixed- and random-effects parameters in the population PK/PD model is performed using a quasi-Newton method based on Broyden-Fletcher-Goldfarb-Shanno (BFGS) updating scheme for computing the inverse, positive-definite Hessian matrix. This is accomplished by means of the unconstrained non-linear mimization software *ucminf*, which deploys line search and trust regions.

As a part of its argument list, *ucminf* allows definition of the initial guess for the inverse, positive-definite Hessian matrix. If no guess is given, the algorithm takes an identity matrix as the initial guess of the inverse Hessian.
The choice of *ucminf* was, besides the fact that it is a robust open-source optimizer, motivated by the possibility of aiding the optimization process by potentially taking advantage of previous optimization information.

An interesting future task is, thus, to investigate the effects of setting initial conditions for the random-effects at a given step equal to optimum conditions at the preceding step. Probably, this should only be instituted for a given set of criteria, for example, related to the difference between the current and the previous estimate of the fixed-effects parameters. Basically, the rationale is that if the fixed-effects at one point are approximately unchanged compared to a preceding estimate, then the random-effects likely also to be approximately unchanged.

The recommendations expressed in this section are summarized and combined with additional recommendations derived from experiences with the PSM prototype in Section 7.2

7.2 Recommendations

Arising from the previous discussion as well as experience with the PSM prototype, a set of recommendations for future work are listed below in a "kind-of" prioritized order:

1. Validation of the population-level modeling part of the PSM prototype should be given highest priority. Special attention is required for determining the correctness of the iteration step count for the iterative optimization of the approximate population log-likelihood function.
2. Mapping function for fixed-effects parameters should be implemented to improve the robustness of the PSM algorithm. Based on the recommendation of Mortensen and Klim [10], the inverse tangent mapping function defined equation (7.1) is suggested for mapping the fixed-effects.
   Likewise, mapping of the random-effects should be analyzed. This may particularly be important when performing single-subject PK/PD analysis. It is recommended to implement the logarithmic mapping function defined in equation (7.2).

3. There are several parts of the PSM model that evidently encourage parallel computing using OpenMP, e.g.:

   (a) The computation of the optimal approximate individual log-likelihood

   (b) The evaluation of the approximate population log-likelihood gradient

   (c) The evaluation of the approximate individual a posteriori log-likelihood gradient

   For optimal performance, tuning of the serial source code should precede any attempts of parallelizing the computational tasks.

4. Implementation of a data acquisition filter that reads input data structured according to standard NONMEM formats is advised. Such a move may improve the odds of attracting current NONMEM users to use PSM as time consuming restructuring of data is eluded.

5. All gradients are currently evaluated using a central difference scheme as suggested by the author of the parameter optimization software *ucminf*. Particularly, computation of the gradient of the approximate population log-likelihood constitute a significant computational burden. An assessment of the accuracy versus computational speed is recommended for varying differencing methods.

6. No assessment has, this far, been made as to determine the relative computational efficency of *ucminf* compared to alternative optimizers. Here, it is advocated that attempts to exploit the information obtained at previous optimization steps should be pursued. For example, if two fixed-effects estimates are approximately identical it may be possible to supply the final optimum conditions returned in the preceding optimization step as input arguments in the new optimization step, thereby increasing the speed of estimating the random-effects.
   Also, it is recommended to compare the *ucminf* with alternative optimizers in terms of accuracy versus computational speed.

7. Currently, the PSM prototype offers very limited options for outputting results of the analysis. An important task is to identify and construct a favorable output interface.

8. Both the fixed-effects and random-effects objects are defined containing 200-dimensional vectors for storing results obtained in each optimization step. With gained experience, it may be advisable to adjust the vector dimensions with respect to some average number of iterations.

9. On long-term basis, priority to the implementation of a Graphical User Interface (GUI) should be given.

This concludes the recommendations for future work. Next, Chapter 8 concludes on the objectives of the thesis.

Chapter 8

Conclusion

In accordance with the thesis objectives, the following results has been achieved:

- A prototype for population PK/PD modeling based on non-linear mixed-effects models using stochastic differential equations has been proposed. The prototype handles linear time-invariant and linear time-varying models. The individual-level modeling has been validated, whereas the population-level algorithm awaits validation.
- The prototype is implemented in Fortran 95, which has been chosen with consideration to known high computational speed, availability of scientific software packages and support of OpenMP shared-memory multiprocessing API for creating parallel programs.
- The design of the prototype has given high priority to the construction of generic procedure interfaces.
- To assist future model extensions and modifications, interfaces for procedures, model declaration files and definitions of data objects have been thoroughly documented.
- A set of recommendations has been proposed for future development of the prototype.

Appendix A

Interface Description

Appendix A presents the interface descriptions found in the headers of each entity in the PSM source code. In order to improve usability of the PSM prototype, detailed commenting is found particularly in the headings of the model declaration files.

The contents of Appendix A are summerized here:

Appendix A.1 documents the header of the main program.

Appendix A.2 documents the headers of the model declaration files.

Appendix A.3 documents the interfaces of the derive types.

AppendixA.4 documents the interfaces of the source code files.

A.1 Program interface

Table A.1: Interface description: `MAIN`

```
!---------------------------------------------------------------------!
!                                                                     !
!      POPULATION PHARMACOKINETICS/PHARMACODYNAMICS MODELLING         !
!                                                                     !
!          Non-Linear Mixed-Effects (NLME) models using               !
!            Stochastic Differential Equations (SDEs)                 !
!                                                                     !
!---------------------------------------------------------------------!
!                                                                     !
!      Department of Informatics and Mathematical Modelling (IMM)     !
!      Technical University of Denmark                                !
!      Contact: Henrik Madsen, e-mail: hm[at]imm.dtu.dk               !
!                                                                     !
!      Copyright 2007. All rights reserved.                           !
!                                                                     !
!      * NOT TO BE REPRODUCED OR DISTRIBUTED WITHOUT PERMISSION       !
!        FROM DEPARTMENT OF INFORMATICS & MATHEMATICAL MODELLING      !
!        (IMM), TECHNICAL UNIVERSITY OF DENMARK.                      !
!                                                                     !
!---------------------------------------------------------------------!
!
!COM-----------------------------------------------------------------!
!COM  VERSION    DATE               NAME                             !
!COM-----------------------------------------------------------------!
!COM      0.1    2007-02-01         Andreas S. Christensen           !
!COM                                                                 !
!COM                                                                 !
!COM-----------------------------------------------------------------!
!COM
!COM-----------------------------------------------------------------!
!COM  SPECIFICATIONS & CONSTRAINTS                                   !
!COM-----------------------------------------------------------------!
!COM                                                                 !
!COM  The prototype handles:                                         !
!COM                                                                 !
!COM       - Multivariate inputs, outputs, covariates and states     !
!COM       - Multiple individuals                                    !
!COM       - Individual number of observations for each individual   !
!COM                                                                 !
!COM       - Population modelling - linear models:                   !
!COM                                                                 !
!COM           1) linear time-invariant (LTI)                        !
!COM           2) linear time-varying (LTV)                          !
!COM                                                                 !
!COM       - Individual modelling - linear models:                   !
!COM                                                                 !
!COM           3) linear time-invariant (LTI)                        !
!COM           4) linear time-varying (LTV)                          !
!COM                                                                 !
```

Continued on next page...

Table A.1: Interface description: `MAIN` – Continued.

```
!COM-----------------------------------------------------------------!
!COM                                                                 !
!COM   COMMENTS                                                      !
!COM                                                                 !
!COM        :: Population modeling part NOT validated.               !
!COM                                                                 !
!COM        :: Optimization of individual parameters (single subject !
!COM           modeling) allowed, since this part has been validated !
!COM           for linear models (LTI/LTV).                          !
!COM                                                                 !
!COM        :: Not implemented:                                      !
!COM                                                                 !
!COM           - Non-linear models                                   !
!COM           - Missing observations                                !
!COM           - Mapping of population parameters (THETA)            !
!COM           - First-order hold                                    !
!COM           - Dose administration in evaluation of states         !
!COM                                                                 !
!COM-----------------------------------------------------------------!
```

A.2 Model declaration files interfaces

Table A.2: Interface description: IOSSTAT

```
!-----------------------------------------------------------------------!
!                                                                       !
!    USER SPECIFICATIONS OF INPUT/OUTPUT CONTROL                        !
!                                                                       !
!-----------------------------------------------------------------------!
!
!COM-----------------------------------------------------------------------!
!COM                                                                     !
!COM   FILE NAME                                                        !
!COM        'IOSSTAT'                                                   !
!COM                                                                     !
!COM   FILE DIRECTORY                                                   !
!COM        '/USER/'                                                    !
!COM                                                                     !
!COM   INCLUDED IN                                                      !
!COM        Module 'MOIOS'                                              !
!COM                                                                     !
!COM   PURPOSE                                                          !
!COM        To specify I/O conditions during execution of PSM.          !
!COM                                                                     !
!COM   PARAMETERS                                                       !
!COM        IOVAR                                                       !
!COM              I/O control variable of type INTEGER.                 !
!COM                                                                     !
!COM              =0:  No print of intermediate results                 !
!COM                                                                     !
!COM              =1:  Print intermediate results                       !
!COM                                                                     !
!COM-----------------------------------------------------------------------!
```

Table A.3: Interface description: AMAT

```
!-----------------------------------------------------------------------!
!                                                                       !
!    SPECIFICATION OF MATRIX 'A' IN LINEAR MODELS (LTI/LTV)             !
!                                                                       !
!-----------------------------------------------------------------------!
!
!COM-----------------------------------------------------------------------!
!COM                                                                     !
!COM   FILE NAME                                                        !
!COM        'AMAT'                                                      !
!COM                                                                     !
!COM   FILE DIRECTORY                                                   !
!COM        '/USER/'                                                    !
!COM                                                                     !
!COM   INCLUDED IN                                                      !
!COM        Subroutine 'LINEAR_MODEL'                                   !
!COM                                                                     !
!COM   PURPOSE                                                          !
!COM        To define matrix A in the linear models (LTI/LTV):          !
!COM                                                                     !
!COM           #1: dX = [A*X + B*U]dt + PI*dW                           !
!COM                                                                     !
!COM           #2:  Y =  C*X + D*U + e, e~N(0,SIGMA)                    !
!COM                                                                     !
!COM        For linear time-invariant (LTI) model:                      !
!COM                                                                     !
!COM                A = A(PHI)         B = B(PHI)                       !
!COM                C = A(PHI)         D = D(PHI)                       !
!COM                PI = PI(PHI)       SIGMA = SIGMA(PHI)               !
!COM                                                                     !
!COM        For linear time-varying (LTV) model:                        !
!COM                                                                     !
!COM                A = A(X,U,t,PHI)   B = B(X,U,t,PHI)                 !
!COM                C = A(X,U,t,PHI)   D = D(X,U,t,PHI)                 !
!COM                PI= PI(U,t,PHI)    SIGMA = SIGMA(U,t,PHI)           !
!COM                                                                     !
!COM   PARAMETERS                                                       !
!COM        A                                                           !
!COM             State coefficient matrix in state eqn., see (#1)       !
!COM             - matrix dimension (NX,NX).                            !
!COM                                                                     !
!COM        NPHI                                                        !
!COM             Number of individual parameters (dimension of PHI).    !
!COM                                                                     !
!COM        NU                                                          !
!COM             Number of input variables (dimension of U).            !
!COM                                                                     !
!COM        NX                                                          !
!COM             Number of state variables (dimension of X).            !
!COM                                                                     !
!COM        PHI                                                         !
!COM             Individual parameters vector - vector dimension        !
!COM             (NPHI).                                                !
!COM                                                                     !
!COM        U                                                           !
!COM             Input variables vector - vector dimension (NU).        !
!COM                                                                     !
!COM        X                                                           !
!COM             State variables vector - vector dimension (NX).        !
!COM                                                                     !
!COM-----------------------------------------------------------------------!
```

Table A.4: Interface description: `BMAT`

```
!-----------------------------------------------------------------------!
!                                                                       !
!    SPECIFICATION OF MATRIX 'B' IN LINEAR MODELS (LTI/LTV)             !
!                                                                       !
!-----------------------------------------------------------------------!
!
!COM-----------------------------------------------------------------------!
!COM                                                                     !
!COM   FILE NAME                                                        !
!COM        'BMAT'                                                      !
!COM                                                                     !
!COM   FILE DIRECTORY                                                   !
!COM        '/USER/'                                                    !
!COM                                                                     !
!COM   INCLUDED IN                                                      !
!COM        Subroutine 'LINEAR_MODEL'                                   !
!COM                                                                     !
!COM   PURPOSE                                                          !
!COM        To define matrix B in the linear models (LTI/LTV):          !
!COM                                                                     !
!COM           #1: dX = [A*X + B*U]dt + PI*dW                           !
!COM                                                                     !
!COM           #2:  Y =  C*X + D*U + e, e~N(0,SIGMA)                    !
!COM                                                                     !
!COM        For linear time-invariant (LTI) model:                      !
!COM                                                                     !
!COM                A = A(PHI)         B = B(PHI)                       !
!COM                C = A(PHI)         D = D(PHI)                       !
!COM                PI = PI(PHI)       SIGMA = SIGMA(PHI)               !
!COM                                                                     !
!COM        For linear time-varying (LTV) model:                        !
!COM                                                                     !
!COM                A = A(X,U,t,PHI)   B = B(X,U,t,PHI)                 !
!COM                C = A(X,U,t,PHI)   D = D(X,U,t,PHI)                 !
!COM                PI= PI(U,t,PHI)    SIGMA = SIGMA(U,t,PHI)           !
!COM                                                                     !
!COM   PARAMETERS                                                       !
!COM        B                                                           !
!COM             Output coefficient matrix in state eqn., see (#1)      !
!COM             - matrix dimension (NX,NU).                            !
!COM                                                                     !
!COM        NPHI                                                        !
!COM             Number of individual parameters (dimension of PHI).    !
!COM                                                                     !
!COM        NU                                                          !
!COM             Number of input variables (dimension of U).            !
!COM                                                                     !
!COM        NX                                                          !
!COM             Number of state variables (dimension of X).            !
!COM                                                                     !
!COM        PHI                                                         !
!COM             Individual parameters vector - vector dimension        !
!COM             (NPHI).                                                !
!COM                                                                     !
!COM        U                                                           !
!COM             Input variables vector - vector dimension (NU).        !
!COM                                                                     !
!COM        X                                                           !
!COM             State variables vector - vector dimension (NX).        !
!COM                                                                     !
!COM-----------------------------------------------------------------------!
```

Table A.5: Interface description: `CMAT`

```
!-------------------------------------------------------------------!
!                                                                   !
!    SPECIFICATION OF MATRIX 'C' IN LINEAR MODELS (LTI/LTV)         !
!                                                                   !
!-------------------------------------------------------------------!
!
!COM---------------------------------------------------------------!
!COM                                                                !
!COM   FILE NAME                                                   !
!COM        'CMAT'                                                 !
!COM                                                                !
!COM   FILE DIRECTORY                                              !
!COM        '/USER/'                                               !
!COM                                                                !
!COM   INCLUDED IN                                                 !
!COM        Subroutine 'LINEAR_MODEL'                              !
!COM                                                                !
!COM   PURPOSE                                                     !
!COM        To define matrix C in the linear models (LTI/LTV):     !
!COM                                                                !
!COM           #1: dX = [A*X + B*U]dt + PI*dW                       !
!COM                                                                !
!COM           #2:  Y =  C*X + D*U + e, e~N(0,SIGMA)                !
!COM                                                                !
!COM        For linear time-invariant (LTI) model:                  !
!COM                                                                !
!COM                A = A(PHI)         B = B(PHI)                   !
!COM                C = A(PHI)         D = D(PHI)                   !
!COM                PI = PI(PHI)       SIGMA = SIGMA(PHI)           !
!COM                                                                !
!COM        For linear time-varying (LTV) model:                    !
!COM                                                                !
!COM                A = A(X,U,t,PHI)   B = B(X,U,t,PHI)             !
!COM                C = A(X,U,t,PHI)   D = D(X,U,t,PHI)             !
!COM                PI= PI(U,t,PHI)    SIGMA = SIGMA(U,t,PHI)       !
!COM                                                                !
!COM   PARAMETERS                                                  !
!COM        C                                                       !
!COM             State coefficient matrix in observation eqn.,      !
!COM             see (#2) - matrix dimension (NY,NX).              !
!COM                                                                !
!COM        NPHI                                                    !
!COM             Number of individual parameters (dimension of PHI).!
!COM                                                                !
!COM        NU                                                      !
!COM             Number of input variables (dimension of U).        !
!COM                                                                !
!COM        NX                                                      !
!COM             Number of state variables (dimension of X).        !
!COM                                                                !
!COM        NY                                                      !
!COM             Number of output variables (dimension of Y).       !
!COM                                                                !
!COM        PHI                                                     !
!COM             Individual parameters vector - vector dimension    !
!COM             (NPHI).                                            !
!COM                                                                !
!COM        U                                                       !
!COM             Input variables vector - vector dimension (NU).    !
!COM                                                                !
!COM        X                                                       !
!COM             State variables vector - vector dimension (NX).    !
!COM                                                                !
!COM---------------------------------------------------------------!
```

Table A.6: Interface description: DMAT

```
!-----------------------------------------------------------------------!
!                                                                       !
!    SPECIFICATION OF MATRIX 'D' IN LINEAR MODELS (LTI/LTV)             !
!                                                                       !
!-----------------------------------------------------------------------!
!
!COM-----------------------------------------------------------------------!
!COM                                                                     !
!COM   FILE NAME                                                        !
!COM        'DMAT'                                                       !
!COM                                                                     !
!COM   FILE DIRECTORY                                                   !
!COM        '/USER/'                                                     !
!COM                                                                     !
!COM   INCLUDED IN                                                      !
!COM        Subroutine 'LINEAR_MODEL'                                    !
!COM                                                                     !
!COM   PURPOSE                                                          !
!COM        To define matrix D in the linear models (LTI/LTV):           !
!COM                                                                     !
!COM           #1: dX = [A*X + B*U]dt + PI*dW                            !
!COM                                                                     !
!COM           #2:  Y =  C*X + D*U + e, e~N(0,SIGMA)                     !
!COM                                                                     !
!COM        For linear time-invariant (LTI) model:                       !
!COM                                                                     !
!COM                A = A(PHI)         B = B(PHI)                       !
!COM                C = A(PHI)         D = D(PHI)                       !
!COM                PI = PI(PHI)       SIGMA = SIGMA(PHI)               !
!COM                                                                     !
!COM        For linear time-varying (LTV) model:                         !
!COM                                                                     !
!COM                A = A(X,U,t,PHI)   B = B(X,U,t,PHI)                 !
!COM                C = A(X,U,t,PHI)   D = D(X,U,t,PHI)                 !
!COM                PI= PI(U,t,PHI)    SIGMA = SIGMA(U,t,PHI)           !
!COM                                                                     !
!COM   PARAMETERS                                                       !
!COM        D                                                            !
!COM             Output coefficient matrix in observation eqn.,          !
!COM             see (#2) - matrix dimension (NY,NU).                    !
!COM                                                                     !
!COM        NPHI                                                         !
!COM             Number of individual parameters (dimension of PHI).     !
!COM                                                                     !
!COM        NU                                                           !
!COM             Number of input variables (dimension of U).             !
!COM                                                                     !
!COM        NX                                                           !
!COM             Number of state variables (dimension of X).             !
!COM                                                                     !
!COM        NY                                                           !
!COM             Number of output variables (dimension of Y).            !
!COM                                                                     !
!COM        PHI                                                          !
!COM             Individual parameters vector - vector dimension         !
!COM             (NPHI).                                                 !
!COM                                                                     !
!COM        U                                                            !
!COM             Input variables vector - vector dimension (NU).         !
!COM                                                                     !
!COM        X                                                            !
!COM             State variables vector - vector dimension (NX).         !
!COM                                                                     !
!COM-----------------------------------------------------------------------!
```

Table A.7: Interface description: `OMEGAMAT`

```
!-----------------------------------------------------------------------!
!                                                                       !
!    SPECIFICATION OF MATRIX 'OMEGA' IN LINEAR AND NON-LINEAR MODELS    !
!                                                                       !
!-----------------------------------------------------------------------!
!
!COM-----------------------------------------------------------------------!
!COM                                                                     !
!COM   FILE NAME                                                        !
!COM        'OMEGAMAT'                                                   !
!COM                                                                     !
!COM   FILE DIRECTORY                                                   !
!COM        '/USER/'                                                     !
!COM                                                                     !
!COM   INCLUDED IN                                                      !
!COM        Function 'FOMEGA' - part of Module MOPROCS                   !
!COM                                                                     !
!COM   PURPOSE                                                          !
!COM        To define the matrix OMEGA, the variance-covariance          !
!COM        matrix of the random-effects ETA. The random-effects         !
!COM        vector ETA is expressed in the second-stage model:           !
!COM                                                                     !
!COM           #1: PHI = h(THETA,Z)*EXP(ETA), ETA~N(0,OMEGA)             !
!COM                                                                     !
!COM        Where:                                                       !
!COM                                                                     !
!COM             OMEGA = OMEGA(THETA)                                    !
!COM                                                                     !
!COM   PARAMETERS                                                       !
!COM        FOMEGA                                                       !
!COM             Variance-covariance matrix of the random-effects ETA    !
!COM             in the second stage model, see (#1) - matrix            !
!COM             dimension (NETA,NETA).                                  !
!COM                                                                     !
!COM        ETA                                                          !
!COM              Random-effects vector - vector dimension (NETA).       !
!COM                                                                     !
!COM        NETA                                                         !
!COM             Number of random-effects (dimension of ETA).            !
!COM                                                                     !
!COM        NPHI                                                         !
!COM             Number of individual parameters (dimension of PHI).     !
!COM                                                                     !
!COM        NTHETA                                                       !
!COM             Number of fixed-effects (dimension of THETA).           !
!COM                                                                     !
!COM        NZ                                                           !
!COM             Number of covariates (dimension of Z).                  !
!COM                                                                     !
!COM        FPHI                                                         !
!COM             Individual parameters vector - vector dimension         !
!COM             (NPHI).                                                 !
!COM                                                                     !
!COM        THETA                                                        !
!COM             Fixed-effects vector - vector dimension (NTHETA).       !
!COM                                                                     !
!COM        Z                                                            !
!COM             Covariates vector - vector dimension (NZ).              !
!COM                                                                     !
!COM-----------------------------------------------------------------------!
```

Table A.8: Interface description: PARAMS

```
!-----------------------------------------------------------------------!
!                                                                       !
!    SPECIFICATION OF GLOBAL PARAMETERS                                 !
!                                                                       !
!-----------------------------------------------------------------------!
!
!COM-----------------------------------------------------------------------!
!COM                                                                     !
!COM   FILE NAME                                                        !
!COM        'PARAMS'                                                    !
!COM                                                                     !
!COM   FILE DIRECTORY                                                   !
!COM        '/USER/'                                                    !
!COM                                                                     !
!COM   INCLUDED IN                                                      !
!COM        Module 'MODATA'                                             !
!COM                                                                     !
!COM   PURPOSE                                                          !
!COM        To define global parameters: NU, NX, NY, NZ, NTHETA,        !
!COM        NETA, NPHI, IMODEL and PS.                                  !
!COM                                                                     !
!COM   PARAMETERS                                                       !
!COM        NPHI                                                        !
!COM            Number of individual parameters (dimension of PHI).     !
!COM                                                                     !
!COM        NU                                                          !
!COM            Number of input variables (dimension of U).             !
!COM                                                                     !
!COM        NX                                                          !
!COM            Number of state variables (dimension of X).             !
!COM                                                                     !
!COM        NY                                                          !
!COM            Number of output variables (dimension of Y).            !
!COM                                                                     !
!COM        NZ                                                          !
!COM            Number of covariates (dimension of Z).                  !
!COM                                                                     !
!COM        NETA                                                        !
!COM            Number of random-effects (dimension of ETA).            !
!COM                                                                     !
!COM        NTHETA                                                      !
!COM            Number of fixed-effects (dimension of THETA).           !
!COM                                                                     !
!COM        IMODEL                                                      !
!COM            Defines model for Kalman filtering procedure (depends   !
!COM            on data and model structure).                           !
!COM                                                                     !
!COM            For population modeling:                                !
!COM                                                                     !
!COM            = 0: LTI-model (linear time-invariant)                  !
!COM            = 1: LTV-model (linear time-varying)                    !
!COM            = 2:  NL-model (non-linear model)                       !
!COM                                                                     !
```

Continued on next page...

Table A.8: Interface description: `PARAMS` – Continued.

```
!COM            For individual modeling:                                !
!COM                                                                    !
!COM            =10: LTI-model (linear time-invariant)                  !
!COM            =11: LTV-model (linear time-varying)                    !
!COM            =12:  NL-model (non-linear model)                       !
!COM                                                                    !
!COM        PS                                                          !
!COM            Pre-specified 'initial state covariance scaling         !
!COM            factor (see Eqn. (1.117) [CTSM 2.3 Math Guide, Dec.     !
!COM            2003, Kristensen, N.R.]).                               !
!COM                                                                    !
!COM-------------------------------------------------------------------!
!COM                                                                    !
!COM   COMMENTS                                                         !
!COM        ::  NL-model not implemented [CHRISTENSEN,A.S.,2007-02-04]!
!COM                                                                    !
!COM        ::  Dimension of data files must be specified in            !
!COM            '/USER/READSPEC' [CHRISTENSEN,A.S.,2007-02-04]          !
!COM                                                                    !
!COM-------------------------------------------------------------------!
```

Table A.9: Interface description: PHIVEC

```
!-----------------------------------------------------------------------!
!                                                                       !
!    SPECIFICATIONS OF VECTOR 'PHI' - INDIVIDUAL PARAMETERS             !
!                                                                       !
!-----------------------------------------------------------------------!
!
!COM-----------------------------------------------------------------------!
!COM                                                                     !
!COM   FILE NAME                                                        !
!COM        'PHIVEC'                                                    !
!COM                                                                     !
!COM   FILE DIRECTORY                                                   !
!COM        '/USER/'                                                    !
!COM                                                                     !
!COM   INCLUDED IN                                                      !
!COM        Function 'FPHI' - part of Module MOPROCS                    !
!COM                                                                     !
!COM   PURPOSE                                                          !
!COM        To define the vector PHI, the individual parameters         !
!COM        vector. PHI is expressed in the second-stage model:         !
!COM                                                                     !
!COM           #1: PHI = h(THETA,Z)*EXP(ETA), ETA~N(0,OMEGA)            !
!COM                                                                     !
!COM        Where:                                                      !
!COM                                                                     !
!COM             OMEGA = OMEGA(THETA)                                   !
!COM                                                                     !
!COM   PARAMETERS                                                       !
!COM        FOMEGA                                                      !
!COM             Variance-covariance matrix of the random-effects ETA   !
!COM             in the second stage model, see (#1) - matrix           !
!COM             dimension (NETA,NETA).                                 !
!COM                                                                     !
!COM        ETA                                                         !
!COM              Random-effects vector - vector dimension (NETA).       !
!COM                                                                     !
!COM        NETA                                                        !
!COM             Number of random-effects (dimension of ETA).           !
!COM                                                                     !
!COM        NPHI                                                        !
!COM             Number of individual parameters (dimension of PHI).    !
!COM                                                                     !
!COM        NTHETA                                                      !
!COM             Number of fixed-effects (dimension of THETA).          !
!COM                                                                     !
!COM        NZ                                                          !
!COM             Number of covariates (dimension of Z).                 !
!COM                                                                     !
!COM        FPHI                                                        !
!COM             Individual parameters vector - vector dimension        !
!COM             (NPHI).                                                !
!COM                                                                     !
!COM        THETA                                                       !
!COM             Fixed-effects vector - vector dimension (NTHETA).      !
!COM                                                                     !
!COM        Z                                                           !
!COM             Covariates vector - vector dimension (NZ).             !
!COM                                                                     !
!COM-----------------------------------------------------------------------!
```

Table A.10: Interface description: PIMAT

```
!-------------------------------------------------------------------!
!                                                                   !
!    SPECIFICATION OF MATRIX 'PI' IN LINEAR AND NON-LINEAR MODELS   !
!                                                                   !
!-------------------------------------------------------------------!
!
!COM---------------------------------------------------------------!
!COM                                                                !
!COM   FILE NAME                                                   !
!COM        'PIMAT'                                                 !
!COM                                                                !
!COM   FILE DIRECTORY                                              !
!COM        '/USER/'                                                !
!COM                                                                !
!COM   INCLUDED IN                                                 !
!COM        Function 'FPI' - part of Module MOPROCS                 !
!COM                                                                !
!COM   PURPOSE                                                     !
!COM        To define matrix PI in the structural model. The linear !
!COM        models (LTI/LTV) are defined by:                        !
!COM                                                                !
!COM           #1: dX = [A*X + B*U]dt + PI*dW                       !
!COM                                                                !
!COM           #2:  Y =  C*X + D*U + e, e~N(0,SIGMA)                !
!COM                                                                !
!COM        For linear time-invariant (LTI) model:                  !
!COM                                                                !
!COM                A = A(PHI)         B = B(PHI)                   !
!COM                C = A(PHI)         D = D(PHI)                   !
!COM                PI = PI(PHI)       SIGMA = SIGMA(PHI)           !
!COM                                                                !
!COM        For linear time-varying (LTV) model:                    !
!COM                                                                !
!COM                A = A(X,U,t,PHI)   B = B(X,U,t,PHI)             !
!COM                C = A(X,U,t,PHI)   D = D(X,U,t,PHI)             !
!COM                PI= PI(U,t,PHI)    SIGMA = SIGMA(U,t,PHI)       !
!COM                                                                !
!COM   PARAMETERS                                                  !
!COM        FPI                                                     !
!COM             Magnitude of system variability in state eqn.,     !
!COM             see (#1) - matrix dimension (NX,NX).              !
!COM                                                                !
!COM        NPHI                                                    !
!COM             Number of individual parameters (dimension of PHI).!
!COM                                                                !
!COM        NU                                                      !
!COM             Number of input variables (dimension of U).       !
!COM                                                                !
!COM        NX                                                      !
!COM             Number of state variables (dimension of X).       !
!COM                                                                !
!COM        PHI                                                     !
!COM             Individual parameters vector - vector dimension   !
!COM             (NPHI).                                           !
!COM                                                                !
!COM        U                                                       !
!COM             Input variables vector - vector dimension (NU).   !
!COM                                                                !
!COM        X                                                       !
!COM             State variables vector - vector dimension (NX).   !
!COM                                                                !
!COM---------------------------------------------------------------!
```

Table A.11: Interface description: READSPEC

```
!---------------------------------------------------------------------!
!                                                                     !
!    SPECIFICATION OF 'DATAFILES DETAILS'                             !
!                                                                     !
!---------------------------------------------------------------------!
!
!COM-----------------------------------------------------------------!
!COM                                                                  !
!COM   FILE NAME                                                     !
!COM        'READSPEC'                                               !
!COM                                                                  !
!COM   FILE DIRECTORY                                                !
!COM        '/USER/'                                                 !
!COM                                                                  !
!COM   INCLUDED IN                                                   !
!COM        Module 'MODATA'                                          !
!COM                                                                  !
!COM   PURPOSE                                                       !
!COM        To specify the number of individuals in model, NID, and  !
!COM        dimensions of the data files:                            !
!COM                                                                  !
!COM           #1: inputs and outputs datafile                        !
!COM                                                                  !
!COM           #2: dosing datafile                                    !
!COM                                                                  !
!COM   PARAMETERS                                                    !
!COM        NID                                                      !
!COM              Number of patients in model (must be identical to  !
!COM              number of individuals contained in inputs/outputs  !
!COM              datafile, namely 'datfil1' in subroutine READDAT in !
!COM              module MODATA.                                     !
!COM                                                                  !
!COM        NROWS_DATA                                               !
!COM              Number of rows in inputs/outputs datafile (named   !
!COM              'datfil1' in subroutine READDAT in module MODATA). !
!COM                                                                  !
!COM        NROWS_DOSE                                               !
!COM              Number of rows in dosing datafile (named 'datfil2' !
!COM              in subroutine READDAT in module MODATA).           !
!COM                                                                  !
!COM-----------------------------------------------------------------!
```

Table A.12: Interface description: SIGMAMAT

```
!--------------------------------------------------------------------!
!                                                                    !
!    SPECIFICATION OF MATRIX 'SIGMA' IN LINEAR AND NON-LINEAR MODELS !
!                                                                    !
!--------------------------------------------------------------------!
!
!COM-----------------------------------------------------------------!
!COM                                                                  !
!COM   FILE NAME                                                     !
!COM        'SIGMAMAT'                                               !
!COM                                                                  !
!COM   FILE DIRECTORY                                                !
!COM        '/USER/'                                                 !
!COM                                                                  !
!COM   INCLUDED IN                                                   !
!COM        Function 'FSIGMA' - part of Module MOPROCS               !
!COM                                                                  !
!COM   PURPOSE                                                       !
!COM        To define matrix SIGMA, the measurement error variance-  !
!COM        covariance matrix,  the in the observation eqn., see (#2) !
!COM        The linear models (LTI/LTV) are defined by:              !
!COM                                                                  !
!COM           #1: dX = [A*X + B*U]dt + PI*dW                        !
!COM                                                                  !
!COM           #2:  Y =  C*X + D*U + e, e~N(0,SIGMA)                 !
!COM                                                                  !
!COM        For linear time-invariant (LTI) model:                   !
!COM                                                                  !
!COM                A = A(PHI)         B = B(PHI)                    !
!COM                C = A(PHI)         D = D(PHI)                    !
!COM                PI = PI(PHI)       SIGMA = SIGMA(PHI)            !
!COM                                                                  !
!COM        For linear time-varying (LTV) model:                     !
!COM                                                                  !
!COM                A = A(X,U,t,PHI)   B = B(X,U,t,PHI)              !
!COM                C = A(X,U,t,PHI)   D = D(X,U,t,PHI)              !
!COM                PI= PI(U,t,PHI)    SIGMA = SIGMA(U,t,PHI)        !
!COM                                                                  !
!COM   PARAMETERS                                                    !
!COM        FSIGMA                                                   !
!COM             Measurement error variance-covariance matrix in     !
!COM             observation eqn., see (#1) - matrix dimension       !
!COM             (NY,NY).                                            !
!COM                                                                  !
!COM        NPHI                                                     !
!COM             Number of individual parameters (dimension of PHI). !
!COM                                                                  !
!COM        NU                                                       !
!COM             Number of input variables (dimension of U).         !
!COM                                                                  !
!COM        NY                                                       !
!COM             Number of output variables (dimension of Y).        !
!COM                                                                  !
!COM        PHI                                                      !
!COM             Individual parameters vector - vector dimension     !
!COM             (NPHI).                                             !
!COM                                                                  !
!COM        U                                                        !
!COM             Input variables vector - vector dimension (NU).     !
!COM                                                                  !
!COM-----------------------------------------------------------------!
```

Table A.13: Interface description: THETAVEC

```
!-----------------------------------------------------------------------!
!                                                                       !
!    SPECIFICATION OF VECTOR 'THETA' - INITIAL GUESS OF FIXED-EFFECTS   !
!                                                                       !
!-----------------------------------------------------------------------!
!
!COM-----------------------------------------------------------------------!
!COM                                                                     !
!COM   FILE NAME                                                        !
!COM        'THETAVEC'                                                  !
!COM                                                                     !
!COM   FILE DIRECTORY                                                   !
!COM        '/USER/'                                                    !
!COM                                                                     !
!COM   INCLUDED IN                                                      !
!COM        Function 'FTHETA' - part of Module MOPROCS                  !
!COM                                                                     !
!COM   PURPOSE                                                          !
!COM        To define the vector THETA, the fixed-effects vector.       !
!COM        THETA is expressed in the second-stage model:               !
!COM                                                                     !
!COM           #1: PHI = h(THETA,Z)*EXP(ETA), ETA~N(0,OMEGA)            !
!COM                                                                     !
!COM        Where:                                                      !
!COM                                                                     !
!COM             OMEGA = OMEGA(THETA)                                   !
!COM                                                                     !
!COM   PARAMETERS                                                       !
!COM        FOMEGA                                                      !
!COM             Variance-covariance matrix of the random-effects ETA   !
!COM             in the second stage model, see (#1) - matrix           !
!COM             dimension (NETA,NETA).                                 !
!COM                                                                     !
!COM        ETA                                                         !
!COM              Random-effects vector - vector dimension (NETA).       !
!COM                                                                     !
!COM        NETA                                                        !
!COM             Number of random-effects (dimension of ETA).           !
!COM                                                                     !
!COM        NPHI                                                        !
!COM             Number of individual parameters (dimension of PHI).    !
!COM                                                                     !
!COM        NTHETA                                                      !
!COM             Number of fixed-effects (dimension of THETA).          !
!COM                                                                     !
!COM        NZ                                                          !
!COM             Number of covariates (dimension of Z).                 !
!COM                                                                     !
!COM        FPHI                                                        !
!COM             Individual parameters vector - vector dimension        !
!COM             (NPHI).                                                !
!COM                                                                     !
!COM        FTHETA                                                      !
!COM             Fixed-effects vector - vector dimension (NTHETA).      !
!COM                                                                     !
!COM        Z                                                           !
!COM             Covariates vector - vector dimension (NZ).             !
!COM                                                                     !
!COM-----------------------------------------------------------------------!
```

Table A.14: Interface description: `X0VEC`

```
!-------------------------------------------------------------------------!
!                                                                         !
!    SPECIFICATION OF VECTOR 'X0' - INITIAL STATE VECTOR                  !
!                                                                         !
!-------------------------------------------------------------------------!
!
!COM---------------------------------------------------------------------!
!COM                                                                       !
!COM   FILE NAME                                                          !
!COM        'X0VEC'                                                       !
!COM                                                                       !
!COM   FILE DIRECTORY                                                     !
!COM        '/USER/'                                                      !
!COM                                                                       !
!COM   INCLUDED IN                                                        !
!COM        Function 'X0VEC' - part of Module MOPROCS                     !
!COM                                                                       !
!COM   PURPOSE                                                            !
!COM        To define the vector X0, the initial state variables          !
!COM        vector. The linear models (LTI/LTV) are defined by:           !
!COM                                                                       !
!COM           #1: dX = [A*X + B*U]dt + PI*dW                             !
!COM                                                                       !
!COM           #2:  Y =  C*X + D*U + e, e~N(0,SIGMA)                      !
!COM                                                                       !
!COM        For linear time-invariant (LTI) model:                        !
!COM                                                                       !
!COM                A = A(PHI)         B = B(PHI)                         !
!COM                C = A(PHI)         D = D(PHI)                         !
!COM                PI = PI(PHI)       SIGMA = SIGMA(PHI)                 !
!COM                                                                       !
!COM        For linear time-varying (LTV) model:                          !
!COM                                                                       !
!COM                A = A(X,U,t,PHI)   B = B(X,U,t,PHI)                   !
!COM                C = A(X,U,t,PHI)   D = D(X,U,t,PHI)                   !
!COM                PI= PI(U,t,PHI)    SIGMA = SIGMA(U,t,PHI)             !
!COM                                                                       !
!COM   PARAMETERS                                                         !
!COM        FX0                                                           !
!COM             Initial state vector - vector of dimension (NX).         !
!COM                                                                       !
!COM        NX                                                            !
!COM             Number of state variables (dimension of X).              !
!COM                                                                       !
!COM                                                                       !
!COM---------------------------------------------------------------------!
```

A.3 Data object interfaces

Table A.15: Interface of derived type DOSE

```
!COM-------------------------------------------------------------------!
!COM                                                                   !
!COM   NAME                                                           !
!COM        'DOSE' - Derived type                                      !
!COM                                                                   !
!COM   PURPOSE                                                        !
!COM        Data object for storing individual patient data related    !
!COM        to dose administration of pharmaceutical compounds.        !
!COM        DOSE is part of PATIENT (the data object that contains     !
!COM        data for a single patient).                                !
!COM                                                                   !
!COM   PARAMETERS                                                     !
!COM        NT                                                         !
!COM             Number of doses administered for one patient.         !
!COM                                                                   !
!COM        T                                                          !
!COM             Time of dose administration. Vector of dimension      !
!COM             (DSNT)                                                !
!COM                                                                   !
!COM        AMT                                                        !
!COM             Amount of dose.                                       !
!COM                                                                   !
!COM        CMT                                                        !
!COM             Compartment to which dose is administered.            !
!COM                                                                   !
!COM             =1: compartment no. 1 in the model.                   !
!COM             =2: compartment no. 2 in the model, etc...            !
!COM                                                                   !
!COM        DID                                                        !
!COM             Dose identifier defines whether observations (if      !
!COM             defined on the exact same time as dosing) are         !
!COM             either pre-dose or post-dose observations.            !
!COM                                                                   !
!COM             =0:  pre-dose  (observations made immediately         !
!COM                             before administration of dose).       !
!COM             =1:  post-dose (observations made immediately         !
!COM                             after administration of  dose).       !
!COM                                                                   !
!COM        METH                                                       !
!COM             Method of dose administration.                        !
!COM                                                                   !
!COM             =1:  infusion                                         !
!COM                                                                   !
!COM             =2:  bolus-injection                                  !
!COM                                                                   !
!COM             =3:  intravenous injection                            !
!COM                                                                   !
!COM        DT                                                         !
!COM             Duration of dose administration.                      !
!COM                                                                   !
!COM             = 0.0D0:  instantanous (bolus dose).                  !
!COM             > 0.0D0:  infusion dose.                              !
!COM                                                                   !
!COM-------------------------------------------------------------------!
```

Table A.16: Interface of derived type `ETAOBJECT`

```
!COM-----------------------------------------------------------------!
!COM                                                                 !
!COM   NAME                                                         !
!COM        'ETAOBJECT' - Derived type                               !
!COM                                                                 !
!COM   PURPOSE                                                      !
!COM        Data object containing variables used for computation of !
!COM        eta (individual random effects vector).                  !
!COM                                                                 !
!COM   PARAMETERS                                                   !
!COM        AIAPLL                                                   !
!COM             Negative approximate individual a posteriori log-   !
!COM             likelihood value. Scalar.                           !
!COM                                                                 !
!COM        LL                                                       !
!COM             Negative individual log-likelihood value. Objective !
!COM             function for optimization of random-effects ETA.    !
!COM             Scalar.                                             !
!COM                                                                 !
!COM        ETA                                                      !
!COM             Individual random-effects vector.                   !
!COM                                                                 !
!COM        IETA                                                     !
!COM             Count variable. Counts the number of optimizations. !
!COM                                                                 !
!COM        OPT                                                      !
!COM             Object that stores parameters for optimization of   !
!COM             individual random-effects eta.                      !
!COM                                                                 !
!COM-----------------------------------------------------------------!
```

Table A.17: Interface of derived type KALOBJ

```
!COM-----------------------------------------------------------------------!
!COM                                                                       !
!COM   NAME                                                               !
!COM        'KALOBJ' - Derived type                                        !
!COM                                                                       !
!COM   PURPOSE                                                            !
!COM        Data object containing variables used in Kalman filtering.!
!COM                                                                       !
!COM                                                                       !
!COM   PARAMETERS                                                         !
!COM        R                                                              !
!COM             Output prediction covariance matrix - dimension           !
!COM             (NY,NY,NT).                                               !
!COM                                                                       !
!COM        PF                                                             !
!COM             Current state estimate covariance matrix -                !
!COM             dimension (NX,NX,NT).                                     !
!COM                                                                       !
!COM        PP                                                             !
!COM             State prediction covariance matrix - dimension            !
!COM             (NX,NX,NT).                                               !
!COM                                                                       !
!COM        XF                                                             !
!COM             Current state estimate vector - dimension (NX,NT).        !
!COM                                                                       !
!COM        XP                                                             !
!COM             State prediction vector - dimension (NX,NT).              !
!COM                                                                       !
!COM        YPERR                                                          !
!COM             Output prediction error vector - dimension (NY,NT).       !
!COM                                                                       !
!COM        YP                                                             !
!COM             Output prediction vector - dimension (NY,NT).             !
!COM                                                                       !
!COM-----------------------------------------------------------------------!
```

Table A.18: Interface of derived type `OPTIMOBJECT`

```
!COM-----------------------------------------------------------------------!
!COM                                                                       !
!COM   NAME                                                               !
!COM        'OPTIMOBJECT' - Derived type                                   !
!COM                                                                       !
!COM   PURPOSE                                                            !
!COM        Object containing variables used for optimization proce-       !
!COM        dure for eta (individual random effects vector).               !
!COM                                                                       !
!COM   PARAMETERS                                                         !
!COM        DFMAX                                                          !
!COM             Largest element in the absolute value of the gradi-       !
!COM             ent evaluated in X: MAX(ABS(F'(I)), I=1,2,...             !
!COM                                                                       !
!COM        DX                                                             !
!COM             DX(1):   Initial radius (step size) supplied to the       !
!COM                      optimization procedure 'UCMINF'.                 !
!COM                                                                       !
!COM             DX(2):   Final radius (step size) returned from the       !
!COM                      optimization procedure 'UCMINF'.                 !
!COM                                                                       !
!COM        HESSIAN                                                        !
!COM             Lower triangle of inverse positive definite inverse       !
!COM             Hessian matrix obtained from the minimization pro-        !
!COM             cedure 'UCMINF'.                                          !
!COM                                                                       !
!COM        DPOS                                                           !
!COM             DPOS(1): Index for initial element of inverse Hes-        !
!COM                      sian matrix returned by vector in 'UCMINF'.      !
!COM                                                                       !
!COM             DPOS(2): Index for final element of inverse Hessian       !
!COM                      matrix returned by vector in 'UCMINF'.           !
!COM                                                                       !
!COM        EPS                                                            !
!COM             Desired accuracy of parameter estimate. The               !
!COM             'UCMINF' procedure stops when either one of the           !
!COM             criteria is met:                                          !
!COM                                                                       !
!COM             EPS(1) >= infinity-norm of the computed gradient.         !
!COM                                                                       !
!COM             EPS(2)*(EPS(2)*2-Norm(ETA)) >= 2-Norm(ETA-ETA*).          !
!COM                                                                       !
!COM        MAXFUN                                                         !
!COM             Upper bound on number of calls to the FDF (i.e.           !
!COM             LLDLL or APLDAPL) procedure.                              !
!COM                                                                       !
!COM        DELTA                                                          !
!COM             Step size used for numerical approximation of the         !
!COM             1st order derivative.                                     !
!COM                                                                       !
!COM-----------------------------------------------------------------------!
```

Table A.19: Interface of derived type `PATIENT`

```
!COM-----------------------------------------------------------------------!
!COM                                                                       !
!COM   NAME                                                               !
!COM        'PATIENT' - Derived type                                       !
!COM                                                                       !
!COM                                                                       !
!COM   PURPOSE                                                            !
!COM        Data object for storing individual patient data.               !
!COM                                                                       !
!COM                                                                       !
!COM   PARAMETERS                                                         !
!COM        CHID                                                           !
!COM             Patient ID string (defined in data file).                 !
!COM                                                                       !
!COM        DOSE                                                           !
!COM             Dose administration object (derived type DOSE).           !
!COM                                                                       !
!COM        NT                                                             !
!COM             Number of observations (dimension of T).                  !
!COM                                                                       !
!COM        NOID                                                           !
!COM             Patient ID number (defined as the order of appeara-       !
!COM             nce in data file) between 1 and NID (total number         !
!COM             of patients).                                             !
!COM                                                                       !
!COM        NU                                                             !
!COM             Number of input variables (dimension of U).               !
!COM                                                                       !
!COM        NY                                                             !
!COM             Number of output variables (dimension of Y).              !
!COM                                                                       !
!COM        NZ                                                             !
!COM             Number of covariates (dimension of Z).                    !
!COM                                                                       !
!COM        T                                                              !
!COM             Time of observation vector of dimension (NT).             !
!COM                                                                       !
!COM        U                                                              !
!COM             Input variables matrix of dimension (NT,NU).              !
!COM                                                                       !
!COM        Y                                                              !
!COM             Derived (output) variables matrix of dimension            !
!COM             (NT,NY).                                                  !
!COM                                                                       !
!COM        Z                                                              !
!COM             Covariates vector - vector dimension (NZ).                !
!COM                                                                       !
!COM-----------------------------------------------------------------------!
```

Table A.20: Interface of derived type `THETAOBJECT`

```
!COM-----------------------------------------------------------------------!
!COM                                                                       !
!COM   NAME                                                               !
!COM        'THETAOBJECT' - Derived type                                   !
!COM                                                                       !
!COM   PURPOSE                                                            !
!COM        Data object containing variables used for computation of       !
!COM        theta (the fixed-effects vector).                              !
!COM                                                                       !
!COM                                                                       !
!COM   PARAMETERS                                                         !
!COM        APL                                                            !
!COM             Approximate population likelihood.                        !
!COM                                                                       !
!COM        ITHETA                                                         !
!COM             Count variable. Counts the number of optimizations.       !
!COM                                                                       !
!COM        OPT                                                            !
!COM             Object that stores parameters for optimization of         !
!COM             fixed-effects THETA.                                      !
!COM                                                                       !
!COM        THETA                                                          !
!COM             Fixed-effects vector.                                     !
!COM                                                                       !
!COM-----------------------------------------------------------------------!
```

A.4 Procedure interfaces

Table A.21: Interface of subroutine `AIAPLL`

```
!-----------------------------------------------SUBROUTINE AIAPLL-----!
!
!COM-------------------------------------------------------------------!
!COM  APPROXIMATE INDIVIDUAL A POSTERIORI LOG-LIKELIHOOD               !
!COM-------------------------------------------------------------------!
!COM                                                                   !
!COM   NAME                                                           !
!COM        'AIAPLL' - Subroutine                                      !
!COM                                                                   !
!COM   PURPOSE                                                        !
!COM        Given data for a single patient and a set of fixed-        !
!COM        effects parameters, subroutine AIAPLL determines the       !
!COM        optimal set of random-effects and computes the correspon-  !
!COM        ding 'approximate individual a posteriori log-likelihood',!
!COM        i.e. 'AIAPLL'.                                             !
!COM                                                                   !
!COM        The fixed-effects parameters 'THETA' are contained in the  !
!COM        theta object 'THETAOBJ' (derived type 'THETAOBJECT').      !
!COM        The individual random-effects 'ETA' and the approximate    !
!COM        individual a posteriori log-likelihood are stored in the   !
!COM        eta object 'ETAOBJ' (derived type 'ETAOBJECT').            !
!COM                                                                   !
!COM   REFERENCE                                                      !
!COM        -                                                          !
!COM                                                                   !
!COM   INCLUDES                                                       !
!COM        -                                                          !
!COM                                                                   !
!COM   CALLED BY                                                      !
!COM        Subroutine APLDAPL                                         !
!COM        Subroutine CNTDIFF_APL                                     !
!COM                                                                   !
!COM   CALLS                                                          !
!COM        Subroutine UCMINF_AIAPLL                                   !
!COM                                                                   !
!COM   F95 INTERFACE                                                  !
!COM        SUBROUTINE AIAPLL(MODEL,THETAOBJ,ETAOBJ,POBJ,OMEGA,INFO)  !
!COM                                                                   !
!COM        USE MOTYPES                                                !
!COM        USE MOPARAMS                                               !
!COM        USE MODATA, ONLY: NID                                      !
!COM        USE MOIOS                                                  !
!COM        USE SUNPERF                                                !
!COM        IMPLICIT NONE                                              !
!COM                                                                   !
```

Continued on next page...

Table A.21: Interface of subroutine `AIAPLL` – Continued.

```
!COM        INTERFACE                                                      !
!COM          SUBROUTINE MODEL(LL,KOBJ,PHI,ETA,OMEGA,POBJ,INFO)            !
!COM          USE SUNPERF                                                  !
!COM          USE MOTYPES,ONLY: DOSE,PATIENT,KALOBJ                        !
!COM          USE MOIOS                                                    !
!COM          IMPLICIT NONE                                                !
!COM          INTEGER,INTENT(OUT) :: INFO                                  !
!COM          REAL(8),INTENT(IN) :: PHI(:),ETA(:),OMEGA(:,:)               !
!COM          TYPE(PATIENT),INTENT(IN) :: POBJ                             !
!COM          TYPE(KALOBJ),INTENT(INOUT) :: KOBJ                           !
!COM          REAL(8),INTENT(OUT) :: LL                                    !
!COM          END SUBROUTINE MODEL                                         !
!COM        END INTERFACE                                                  !
!COM                                                                       !
!COM        TYPE(THETAOBJECT),INTENT(IN) :: THETAOBJ                       !
!COM        TYPE(ETAOBJECT),INTENT(INOUT) :: ETAOBJ                        !
!COM        TYPE(PATIENT),INTENT(IN) :: POBJ                               !
!COM        INTEGER,INTENT(OUT) :: INFO                                    !
!COM        REAL(8),INTENT(INOUT) :: OMEGA(:,:)                            !
!COM                                                                       !
!COM   ARGUMENTS                                                          !
!COM        ETAOBJ (input/output)                                          !
!COM             Random-effects object. On exit, ETAOBJ contains the       !
!COM             new set of optimal random-effects and corresponding       !
!COM             approximate individual a posteriori log-likelihood        !
!COM             'AIAPLL'.                                                 !
!COM                                                                       !
!COM        INFO (output)                                                  !
!COM             = 0:  successful exit.                                    !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.      !
!COM                                                                       !
!COM        KOBJ (input/output)                                            !
!COM             Kalman object (derived type 'KALOBJ') containing          !
!COM             results from Kalman filtering.                            !
!COM                                                                       !
!COM        MODEL (module procedure)                                       !
!COM             Kalman filtering procedure, which depends on the          !
!COM             model specification.                                      !
!COM                                                                       !
!COM             The parameter IMODEL dictates, which Kalman filter-       !
!COM             ing method should be used for modelling, i.e.:            !
!COM                                                                       !
!COM                 = 0:  MODEL = LTI_KALMAN                              !
!COM                 = 1:  MODEL = LTV_KALMAN                              !
!COM                 = 2:  MODEL =  NL_KALMAN                              !
!COM                                                                       !
```

Continued on next page...

Table A.21: Interface of subroutine `AIAPLL` – Continued.

```
!COM        OMEGA (input)                                                  !
!COM             Variance-covariance matrix of the random-effects ETA      !
!COM             in the second stage model defined by:                     !
!COM                                                                       !
!COM                #1: PHI = h(THETA,Z)*EXP(ETA), ETA~N(0,OMEGA)          !
!COM                                                                       !
!COM             Matrix of dimension (NETA,NETA).                          !
!COM                                                                       !
!COM        POBJ (input)                                                   !
!COM             Patient object (derived type 'PATIENT') containing        !
!COM             individual patient data. Unchanged on exit.               !
!COM                                                                       !
!COM        THETAOBJ (input)                                               !
!COM             Fixed-effects object (derived type THETAOBJECT).          !
!COM             Unchanged on exit.                                        !
!COM                                                                       !
!COM-----------------------------------------------------------------------!
!COM                                                                       !
!COM   COMMENTS                                                            !
!COM        :: Default conditions for optimization:                        !
!COM                                                                       !
!COM           1) Random variables :  ETA = 0.0D0                          !
!COM           2) Inverse Hessian  :    D = Identity matrix                !
!COM           3) Step size        :   DX = 1.0D0                         !
!COM                                                                       !
!COM           Consider inserting conditions for "improved" initial        !
!COM           conditions based on information on previous optimiza-       !
!COM           tion information stored in the thetaobject (and/or          !
!COM           etaobject).                                                 !
!COM                                                                       !
!COM           [CHRISTENSEN, A.S., 2007-02-01]                             !
!COM                                                                       !
!COM-----------------------------------------------------------------------!
```

Table A.22: Interface of subroutine ALLOC KALMANOBJECT

```
!-----------------------------------SUBROUTINE ALLOC_KALMANOBJECT-----!
!
!COM---------------------------------------------------------------------!
!COM                                                                     !
!COM   NAME                                                             !
!COM        'ALLOC_KALMANOBJECT' - Subroutine                            !
!COM                                                                     !
!COM   PURPOSE                                                          !
!COM        Allocates variables in the Kalman object (derived type       !
!COM        'KALOBJ') given the number of observations (NT) for a        !
!COM        patient.                                                     !
!COM                                                                     !
!COM   REFERENCE                                                        !
!COM        -                                                            !
!COM                                                                     !
!COM   INCLUDES                                                         !
!COM        -                                                            !
!COM                                                                     !
!COM   CALLED BY                                                        !
!COM        Subroutine INIT_ETAOBJECT                                    !
!COM                                                                     !
!COM   CALLS                                                            !
!COM        -                                                            !
!COM                                                                     !
!COM   F95 INTERFACE                                                    !
!COM                                                                     !
!COM        USE MOTYPES                                                  !
!COM        IMPLICIT NONE                                                !
!COM                                                                     !
!COM        INTEGER,INTENT(IN) :: NT                                     !
!COM        INTEGER,INTENT(OUT) :: INFO                                  !
!COM        TYPE(KALOBJ),INTENT(INOUT) :: KOBJ                           !
!COM                                                                     !
!COM   PARAMETERS                                                       !
!COM        KOBJ (output)                                                !
!COM             Kalman object (derived type 'KALOBJ') containing        !
!COM             results from Kalman filtering.                          !
!COM                                                                     !
!COM        NT (input)                                                   !
!COM             Number of observations for an individual patient.       !
!COM                                                                     !
!COM        INFO (output)                                                !
!COM             = 0:  successful exit.                                  !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.    !
!COM                                                                     !
!COM---------------------------------------------------------------------!
```

Table A.23: Interface of subroutine APL

```
!-------------------------------------------------SUBROUTINE APL-----!
!
!COM-----------------------------------------------------------------!
!COM  APPROXIMATE POPULATION LIKELIHOOD                              !
!COM-----------------------------------------------------------------!
!COM                                                                 !
!COM   NAME                                                         !
!COM        'APL' - Subroutine                                       !
!COM                                                                 !
!COM   PURPOSE                                                      !
!COM        Non-linear mixed-effects model based on stochastic       !
!COM        differential equations.                                  !
!COM                                                                 !
!COM        Given data for a population of patients, subroutine APL  !
!COM        computes the optimal set of fixed-effects and random-    !
!COM        effects for each individual.                             !
!COM                                                                 !
!COM        The fixed-effects parameters 'THETA' are contained in the !
!COM        theta object 'THETAOBJ' (derived type 'THETAOBJECT').    !
!COM        The individual random-effects 'ETA' and the approximate  !
!COM        individual a posteriori log-likelihood are stored in the !
!COM        eta objects 'ETAOBJ' (derived type 'ETAOBJECT').         !
!COM                                                                 !
!COM   REFERENCE                                                    !
!COM        -                                                        !
!COM                                                                 !
!COM   INCLUDES                                                     !
!COM        -                                                        !
!COM                                                                 !
!COM   CALLED BY                                                    !
!COM        Program MAIN                                             !
!COM                                                                 !
!COM   CALLS                                                        !
!COM        Subroutine UCMINF_APL                                    !
!COM                                                                 !
!COM   F95 INTERFACE                                                !
!COM        SUBROUTINE APL(MODEL,THETAOBJ,ETAOBJ,POBJ,OMEGA,INFO)   !
!COM                                                                 !
!COM        USE MOTYPES                                              !
!COM        USE MOPARAMS                                             !
!COM        USE MODATA, ONLY: NID                                    !
!COM        USE MOIOS                                                !
!COM        USE SUNPERF                                              !
!COM        IMPLICIT NONE                                            !
!COM                                                                 !
!COM        INTERFACE                                                !
!COM          SUBROUTINE MODEL(LL,KOBJ,PHI,ETA,OMEGA,POBJ,INFO)      !
!COM          USE SUNPERF                                            !
!COM          USE MOTYPES,ONLY: DOSE,PATIENT,KALOBJ                  !
!COM          USE MOIOS                                              !
!COM          IMPLICIT NONE                                          !
!COM          INTEGER,INTENT(OUT) :: INFO                            !
!COM          REAL(8),INTENT(IN) :: PHI(:),ETA(:),OMEGA(:,:)         !
!COM          TYPE(PATIENT),INTENT(IN) :: POBJ                       !
```

Continued on next page...

Table A.23: Interface of subroutine `APL` – Continued.

```
!COM          TYPE(KALOBJ),INTENT(INOUT) :: KOBJ                    !
!COM          REAL(8),INTENT(OUT) :: LL                             !
!COM          END SUBROUTINE MODEL                                  !
!COM       END INTERFACE                                            !
!COM                                                                !
!COM       TYPE(THETAOBJECT),INTENT(INOUT) :: THETAOBJ              !
!COM       TYPE(ETAOBJECT),INTENT(INOUT) :: ETAOBJ(:)               !
!COM       TYPE(PATIENT),INTENT(IN) :: POBJ(:)                      !
!COM       REAL(8),INTENT(OUT) :: OMEGA(NETA,NETA)                  !
!COM       INTEGER,INTENT(OUT) :: INFO                              !
!COM                                                                !
!COM   ARGUMENTS                                                   !
!COM       ETAOBJ (input/output)                                    !
!COM            Random-effects objects of dimension (NID). On exit, !
!COM            ETAOBJ contains the optimal random-effects ETA and  !
!COM            corresponding approximate individual a posteriori   !
!COM            log-likelihood 'AIAPLL'.                            !
!COM            ETAOBJ contains information obtained for each iter- !
!COM            ation ('local' optimum for fixed-effects THETA).    !
!COM                                                                !
!COM       INFO (output)                                            !
!COM            = 0:  successful exit.                              !
!COM            > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.!
!COM                                                                !
!COM       MODEL (module procedure)                                 !
!COM            Kalman filtering procedure, which depends on the    !
!COM            model specification.                                !
!COM                                                                !
!COM            The parameter IMODEL dictates, which Kalman filter- !
!COM            ing method should be used for modelling, i.e.:      !
!COM                                                                !
!COM                = 0:  MODEL = LTI_KALMAN                        !
!COM                = 1:  MODEL = LTV_KALMAN                        !
!COM                = 2:  MODEL =  NL_KALMAN                        !
!COM                                                                !
!COM       OMEGA (output)                                           !
!COM            Variance-covariance matrix of the random-effects ETA !
!COM            in the second stage model defined by:               !
!COM                                                                !
!COM               #1: PHI = h(THETA,Z)*EXP(ETA), ETA~N(0,OMEGA)    !
!COM                                                                !
!COM            Matrix of dimension (NETA,NETA).                    !
!COM                                                                !
!COM       POBJ (input)                                             !
!COM            Patient object (derived type 'PATIENT') containing  !
!COM            individual patient data. Unchanged on exit.         !
!COM                                                                !
```

Continued on next page...

Table A.23: Interface of subroutine `APL` – Continued.

```
!COM        THETAOBJ (input/output)                                    !
!COM             Fixed-effects object (derived type THETAOBJECT).      !
!COM             On input, THETAOBJ contains the initial guess of      !
!COM             the fixed-effects parameter THETAOBJ%THETA(:,1).      !
!COM             On exit, THETAOBJ contains the optimal set of         !
!COM             fixed-effects THETA and corresponding approximate     !
!COM             population likelihood APL.                            !
!COM             THETAOBJ stores historic results for THETA and APL.   !
!COM                                                                   !
!COM-------------------------------------------------------------------!
!COM                                                                   !
!COM   COMMENTS                                                        !
!COM        :: Currently, default conditions for optimization used.    !
!COM                                                                   !
!COM           Consider inserting conditions for "improved" initial    !
!COM           conditions based on information on previous optimiza-   !
!COM           tion information stored in the thetaobject. E.g.,       !
!COM           using previous Hessian matrix may increase optimiza-    !
!COM           tion speed.                                             !
!COM                                                                   !
!COM           [CHRISTENSEN, A.S., 2007-02-01]                         !
!COM                                                                   !
!COM-------------------------------------------------------------------!
```

Table A.24: Interface of subroutine APLDAPL

```
!-----------------------------------------------SUBROUTINE APLDAPL-----!
!
!COM-------------------------------------------------------------------!
!COM  FUNCTION VALUE (APL) AND GRADIENT (DAPL) FOR APPROXIMATE        !
!COM  POPULATION LIKELIHOOD                                           !
!COM-------------------------------------------------------------------!
!COM                                                                  !
!COM   NAME                                                          !
!COM        'APLDAPL' - Subroutine                                    !
!COM                                                                  !
!COM   PURPOSE                                                       !
!COM        Given a set of fixed-effects parameters TMP_THETA, sub-   !
!COM        routine APLDAPL computes the approximate population log-  !
!COM        likelihood (APL) and corresponding gradient DAPL, which   !
!COM        are required for the parameter mimimization procedure     !
!COM        UCMINF_APL.                                               !
!COM                                                                  !
!COM        The fixed-effects parameters 'THETA' are contained in the !
!COM        theta object 'THETAOBJ' (derived type 'THETAOBJECT').     !
!COM        The individual random-effects 'ETA' and the approximate   !
!COM        individual a posteriori log-likelihood are stored in the  !
!COM        eta objects 'ETAOBJ' (derived type 'ETAOBJECT').          !
!COM                                                                  !
!COM   REFERENCE                                                     !
!COM        -                                                         !
!COM                                                                  !
!COM   INCLUDES                                                      !
!COM        -                                                         !
!COM                                                                  !
!COM   CALLED BY                                                     !
!COM        Subroutine UCMINF_APL                                     !
!COM        Subroutine CHKDFN_APL                                     !
!COM        Subroutine SLINE_APL                                      !
!COM                                                                  !
!COM   CALLS                                                         !
!COM        Function FOMEGA                                           !
!COM        Function FAPL                                             !
!COM        Subroutine MAPPING                                        !
!COM        Subroutine AIAPLL                                         !
!COM        Subroutine CNTDIFF_APL                                    !
!COM        Subroutine HESSIAN_AIAPLL                                 !
!COM                                                                  !
!COM   F95 INTERFACE                                                 !
!COM        SUBROUTINE APLDAPL(N,TMP_THETA,GRAD_APL,APL,MODEL,        !
!COM                                THETAOBJ,ETAOBJ,POBJ,OMEGA,INFO) !
!COM                                                                  !
!COM        USE MOTYPES                                               !
!COM        USE MOIOS                                                 !
!COM        USE MOPARAMS                                              !
!COM        USE MODATA, ONLY: NID                                     !
!COM        IMPLICIT NONE                                             !
!COM                                                                  !
```

Continued on next page...

Table A.24: Interface of subroutine `APLDAPL` – Continued.

```
!COM        INTERFACE                                                    !
!COM          SUBROUTINE MODEL(LL,KOBJ,PHI,ETA,OMEGA,POBJ,INFO)          !
!COM          USE SUNPERF                                                !
!COM          USE MOTYPES,ONLY: DOSE,PATIENT,KALOBJ                      !
!COM          USE MOIOS                                                  !
!COM          IMPLICIT NONE                                              !
!COM          INTEGER,INTENT(OUT) :: INFO                                !
!COM          REAL(8),INTENT(IN) :: PHI(:),ETA(:),OMEGA(:,:)             !
!COM          TYPE(PATIENT),INTENT(IN) :: POBJ                           !
!COM          TYPE(KALOBJ),INTENT(INOUT) :: KOBJ                         !
!COM          REAL(8),INTENT(OUT) :: LL                                  !
!COM          END SUBROUTINE MODEL                                       !
!COM        END INTERFACE                                                !
!COM                                                                     !
!COM        INTEGER,INTENT(IN) :: N                                      !
!COM        REAL(8),INTENT(IN) :: TMP_THETA(:)                           !
!COM        REAL(8),INTENT(INOUT) :: OMEGA(:,:)                          !
!COM        INTEGER,INTENT(OUT) :: INFO                                  !
!COM        REAL(8),INTENT(OUT) :: GRAD_APL(:),APL                       !
!COM        TYPE(THETAOBJECT),INTENT(INOUT) :: THETAOBJ                  !
!COM        TYPE(ETAOBJECT),INTENT(INOUT) :: ETAOBJ(:)                   !
!COM        TYPE(PATIENT),INTENT(IN) :: POBJ(:)                          !
!COM                                                                     !
!COM   ARGUMENTS                                                         !
!COM        APL (output)                                                 !
!COM             Negative approximate population log-likelihood, the     !
!COM             objective function for optimization of the fixed-       !
!COM             effects THETA.                                          !
!COM                                                                     !
!COM        ETAOBJ (input/output)                                        !
!COM             Random-effects objects of dimension (NID). On exit,     !
!COM             ETAOBJ has been updated with the optimimization         !
!COM             information returned by UCMINF_APL.                     !
!COM                                                                     !
!COM        GRAD_APL (output)                                            !
!COM             Function gradient required by UCMINF_APL optimiza-      !
!COM             tion procedure, i.e. gradient of the approximate        !
!COM             population likelihood value.                            !
!COM                                                                     !
!COM        INFO (output)                                                !
!COM             = 0:  successful exit.                                  !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.    !
!COM                                                                     !
!COM        MODEL (module procedure)                                     !
!COM             Kalman filtering procedure, which depends on the        !
!COM             model specification.                                    !
!COM                                                                     !
!COM             The parameter IMODEL dictates, which Kalman filter-     !
!COM             ing method should be used for modelling, i.e.:          !
!COM                                                                     !
!COM                 = 0:  MODEL = LTI_KALMAN                            !
!COM                 = 1:  MODEL = LTV_KALMAN                            !
!COM                 = 2:  MODEL =  NL_KALMAN                            !
!COM                                                                     !
```

Continued on next page...

Table A.24: Interface of subroutine APLDAPL – Continued.

```
!COM        N (input)                                                    !
!COM             Number of fixed-effects parameters (=NTHETA)            !
!COM                                                                     !
!COM        OMEGA (output)                                               !
!COM             Variance-covariance matrix of the random-effects ETA    !
!COM             in the second stage model defined by:                   !
!COM                                                                     !
!COM                #1: PHI = h(THETA,Z)*EXP(ETA), ETA~N(0,OMEGA)        !
!COM                                                                     !
!COM             Matrix of dimension (NETA,NETA). On exit, OMEGA has     !
!COM             updated using the current guess for the fixed-          !
!COM             effects parameters TMP_THETA.                           !
!COM                                                                     !
!COM        POBJ (input)                                                 !
!COM             Patient object (derived type 'PATIENT') containing      !
!COM             individual patient data. Unchanged on exit.             !
!COM                                                                     !
!COM        TMP_THETA (input)                                            !
!COM             Fixed-effects parameters for which the approximate      !
!COM             population likelihood 'APL' and its gradient            !
!COM             GRAD_APL is computed.                                   !
!COM                                                                     !
!COM        THETAOBJ (input/output)                                      !
!COM             Fixed-effects object (derived type THETAOBJECT).        !
!COM             On exit, THETAOBJ has been updated with the current     !
!COM             guess for the fixed-effects parameter TMP_THETA         !
!COM             supplied by the parameter optimization procedure        !
!COM             UCMINF_APL.                                             !
!COM                                                                     !
!COM-------------------------------------------------------------------!
```

Table A.25: Interface of subroutine `CNTDIFF_AIAPLL`

```
!---------------------------------------SUBROUTINE CNTDIFF_AIAPLL-----!
!
!COM-----------------------------------------------------------------!
!COM   GRADIENT OF INDIVIDUAL LOG-LIKELIHOOD FUNCTION USING CENTRAL   !
!COM   DIFFERENCE METHOD                                             !
!COM-----------------------------------------------------------------!
!COM                                                                  !
!COM   NAME                                                          !
!COM        CNTDIFF_AIAPLL - Subroutine                               !
!COM                                                                  !
!COM   PURPOSE                                                       !
!COM        Computes the gradient of the individual log-likelihood    !
!COM        'DLL' evaluated in 'TMP_ETA' using a central difference   !
!COM        scheme. The module procedure MODEL must be passed as a    !
!COM        calling argument.                                         !
!COM                                                                  !
!COM   REFERENCE                                                     !
!COM        -                                                         !
!COM                                                                  !
!COM   INCLUDES                                                      !
!COM        -                                                         !
!COM                                                                  !
!COM   CALLED BY                                                     !
!COM        Subroutine UCMINF_AIAPLL                                  !
!COM                                                                  !
!COM   CALLS                                                         !
!COM        Module procedure MODEL (LTI_KALMAN/LTV_KALMAN/NL_KALMAN)  !
!COM                                                                  !
!COM   F95 INTERFACE                                                 !
!COM        SUBROUTINE CNTDIFF_AIAPLL(MODEL,DLL,TMP_ETA,THETAOBJ,     !
!COM                                  ETAOBJ,OMEGA,POBJ,INFO) !
!COM                                                                  !
!COM        USE MOTYPES                                               !
!COM        USE MOPARAMS                                              !
!COM        USE SUNPERF                                               !
!COM        IMPLICIT NONE                                             !
!COM                                                                  !
!COM        INTERFACE                                                 !
!COM          SUBROUTINE MODEL(LL,KOBJ,PHI,ETA,OMEGA,POBJ,INFO)       !
!COM          USE SUNPERF                                             !
!COM          USE MOTYPES,ONLY: DOSE,PATIENT,KALOBJ                   !
!COM          USE MOIOS                                               !
!COM          IMPLICIT NONE                                           !
!COM          INTEGER,INTENT(OUT) :: INFO                             !
!COM          REAL(8),INTENT(IN) :: PHI(:),ETA(:),OMEGA(:,:)          !
!COM          TYPE(PATIENT),INTENT(IN) :: POBJ                        !
!COM          TYPE(KALOBJ),INTENT(INOUT) :: KOBJ                      !
!COM          REAL(8),INTENT(OUT) :: LL                               !
!COM          END SUBROUTINE MODEL                                    !
!COM        END INTERFACE                                             !
!COM                                                                  !
!COM        REAL(8), INTENT(IN) :: TMP_ETA(:),OMEGA(:,:)              !
!COM        REAL(8), INTENT(OUT) :: DLL(:)                            !
!COM        INTEGER, INTENT(OUT) :: INFO                              !
!COM        TYPE(THETAOBJECT),INTENT(IN) :: THETAOBJ                  !
!COM        TYPE(ETAOBJECT),INTENT(IN) :: ETAOBJ                      !
!COM        TYPE(PATIENT), INTENT(IN) :: POBJ                         !
!COM                                                                  !
```

Continued on next page...

Table A.25: Interface of subroutine `CNTDIFF AIAPLL` – Continued.

```
!COM   ARGUMENTS                                                        !
!COM        DLL (output)                                                !
!COM             Numerical approximation of the 1st order derivative    !
!COM             of the individual log-likehood function.               !
!COM                                                                    !
!COM        ETAOBJ (input)                                              !
!COM             Random-effects object (derived type 'ETAOBJECT')       !
!COM             containing results and optimization information.       !
!COM                                                                    !
!COM        INFO (output)                                               !
!COM             = 0:  successful exit.                                 !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.   !
!COM                                                                    !
!COM        MODEL (module procedure)                                    !
!COM             Subroutine computing the individual a posteriori       !
!COM             log-likelihood (either LTI_KALMAN, LTV_KALMAN or       !
!COM             NL_KALMAN).                                            !
!COM                                                                    !
!COM        OMEGA (input)                                               !
!COM             Covariance matrix (NETA,NETA) for the individual       !
!COM             random effects vector ETA.                             !
!COM                                                                    !
!COM        POBJ (input)                                                !
!COM             Patient object of type(patient) containing used for    !
!COM             function evaluation of MODEL.                          !
!COM                                                                    !
!COM        THETAOBJ (input)                                            !
!COM             Fixed-effects object (derived type 'THETAOBJECT')      !
!COM             containing results and optimization information.       !
!COM                                                                    !
!COM                                                                    !
!COM        TMP_ETA (input)                                             !
!COM             Current random-effects parameters for which the        !
!COM             approximate individual log-likelihood 'LL' and its     !
!COM             gradient GRAD_LL is computed. TMP_ETA is supplied      !
!COM             the parameter optimization procedure UCMINF_AIAPLL.    !
!COM                                                                    !
!COM------------------------------------------------------------------!
```

Table A.26: Interface of subroutine `CNTDIFF APL`

```
!-----------------------------------------SUBROUTINE CNTDIFF_APL-----!
!
!COM-----------------------------------------------------------------!
!COM   GRADIENT OF APPROXIMATE POPULATION LIKELIHOOD FUNCTION USING   !
!COM   CENTRAL DIFFERENCE METHOD                                     !
!COM-----------------------------------------------------------------!
!COM                                                                 !
!COM   NAME                                                          !
!COM        CNTDIFF_APL - Subroutine                                 !
!COM                                                                 !
!COM   PURPOSE                                                       !
!COM        Computes the gradient of the approximate population      !
!COM        likelihood 'DAPL' with respect to fixed-effects 'THETA'  !
!COM        using a central difference scheme. The module procedure  !
!COM        MODEL must be passed as a calling argument.              !
!COM                                                                 !
!COM   REFERENCE                                                     !
!COM        -                                                        !
!COM                                                                 !
!COM   INCLUDES                                                      !
!COM        -                                                        !
!COM                                                                 !
!COM   CALLED BY                                                     !
!COM        Subroutine UCMINF_APLDAPL                                !
!COM                                                                 !
!COM   CALLS                                                         !
!COM        Module procedure MODEL (LTI_KALMAN/LTV_KALMAN/NL_KALMAN) !
!COM        Function FOMEGA                                          !
!COM        Function FAPL                                            !
!COM        Subroutine INIT_THETAOBJECT                              !
!COM        Subroutine INIT_ETAOBJECT                                !
!COM        Subroutine AIAPLL                                        !
!COM        Subroutine HESSIAN_AIAPLL                                !
!COM                                                                 !
!COM   F95 INTERFACE                                                 !
!COM        SUBROUTINE CNTDIFF_APL(MODEL,DAPL,THETAOBJ,ETAOBJ,POBJ,  !
!COM                                                          INFO) !
!COM                                                                 !
!COM        USE MOTYPES                                              !
!COM        USE MOPARAMS                                             !
!COM        USE SUNPERF                                              !
!COM        IMPLICIT NONE                                            !
!COM                                                                 !
!COM        INTERFACE                                                !
!COM          SUBROUTINE MODEL(LL,KOBJ,PHI,ETA,OMEGA,POBJ,INFO)      !
!COM          USE SUNPERF                                            !
!COM          USE MOTYPES,ONLY: DOSE,PATIENT,KALOBJ                  !
!COM          USE MOIOS                                              !
!COM          IMPLICIT NONE                                          !
!COM          INTEGER,INTENT(OUT) :: INFO                            !
!COM          REAL(8),INTENT(IN) :: PHI(:),ETA(:),OMEGA(:,:)         !
!COM          TYPE(PATIENT),INTENT(IN) :: POBJ                       !
!COM          TYPE(KALOBJ),INTENT(INOUT) :: KOBJ                     !
!COM          REAL(8),INTENT(OUT) :: LL                              !
!COM          END SUBROUTINE MODEL                                   !
!COM        END INTERFACE                                            !
!COM                                                                 !
```

Continued on next page...

Table A.26: Interface of subroutine `CNTDIFF APL` – Continued.

```
!COM        REAL(8), INTENT(OUT) :: DAPL(:)                          !
!COM        INTEGER, INTENT(OUT) :: INFO                             !
!COM        TYPE(THETAOBJECT),INTENT(IN) :: THETAOBJ                 !
!COM        TYPE(ETAOBJECT),INTENT(IN) :: ETAOBJ(:)                  !
!COM        TYPE(PATIENT), INTENT(IN) :: POBJ(:)                     !
!COM                                                                 !
!COM   ARGUMENTS                                                     !
!COM        DAPL (output)                                            !
!COM             Numerical approximation of the 1st order derivative !
!COM             of the approximate population likehood.             !
!COM                                                                 !
!COM        ETAOBJ (input)                                           !
!COM             Random-effects object (derived type 'ETAOBJECT')    !
!COM             containing results and optimization information.    !
!COM                                                                 !
!COM        INFO (output)                                            !
!COM             = 0:  successful exit.                              !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.!
!COM                                                                 !
!COM        MODEL (module procedure)                                 !
!COM             Subroutine computing the individual a posteriori    !
!COM             log-likelihood (either LTI_KALMAN, LTV_KALMAN       !
!COM             NL_KALMAN).                                         !
!COM                                                                 !
!COM        POBJ (input)                                             !
!COM             Patient object of type(patient) containing used for !
!COM             function evaluation of MODEL.                       !
!COM                                                                 !
!COM        THETAOBJ (input)                                         !
!COM             Fixed-effects object (derived type 'THETAOBJECT')   !
!COM             containing results and optimization information.    !
!COM                                                                 !
!COM-----------------------------------------------------------------!
```

Table A.27: Interface of subroutine DEXPM

```
!-----------------------------------------------SUBROUTINE DEXPM-----!
!
!COM-------------------------------------------------------------------!
!COM   MATRIX EXPONENTIAL AND MATRIX INTEGRAL                          !
!COM-------------------------------------------------------------------!
!COM                                                                   !
!COM   NAME                                                           !
!COM        'DEXPM' - Subroutine                                       !
!COM                                                                   !
!COM   PURPOSE                                                        !
!COM        Computes matrix exponential and matrix integral based on   !
!COM        Eqn. (1.47), (1.48) and (1.49) [CTSM 2.3 Math Guide, Dec.  !
!COM        2003, Kristensen, N.R.].                                   !
!COM                                                                   !
!COM   F95 INTERFACE                                                   !
!COM        SUBROUTINE DEXPM(A,R1M,DT,H3T,R1SM,IDEG,INFO)              !
!COM                                                                   !
!COM        USE SUNPERF                                                !
!COM        USE MOPARAMS                                               !
!COM                                                                   !
!COM        INTEGER,INTENT(IN) :: IDEG                                 !
!COM        REAL(8),INTENT(IN) :: A(:,:),R1M(:,:),DT                   !
!COM        INTEGER,INTENT(OUT) :: INFO                                !
!COM        REAL(8),INTENT(OUT) :: H3T(NX,NX),R1SM(NX,NX)              !
!COM                                                                   !
!COM   REFERENCE                                                       !
!COM        Eqn. (1.47), (1.48) and (1.49) [CTSM 2.3 Math Guide, Dec.  !
!COM        2003, Kristensen, N.R.].                                   !
!COM                                                                   !
!COM   INCLUDES                                                        !
!COM        -                                                          !
!COM                                                                   !
!COM   CALLED BY                                                      !
!COM        Subroutine LTI_KALMAN                                      !
!COM        Subroutine LTV_KALMAN                                      !
!COM                                                                   !
!COM   CALLS                                                           !
!COM        Subroutine DGPADM                                          !
!COM                                                                   !
!COM   ARGUMENTS                                                       !
!COM        A (input)                                                  !
!COM             Matrix A of the linear (LTI) model of dimensions      !
!COM             (NX,NX).                                              !
!COM                                                                   !
!COM        R1M (input)                                                !
!COM             SIGMA*TRANSPOSE(SIGMA) matrix of dimensions (NX,NX).  !
!COM                                                                   !
!COM        DT (input)                                                 !
!COM             Time interval T(k)-T(k-1). Scalar.                    !
!COM                                                                   !
!COM        H3T (output)                                               !
!COM             Matrix exponential EXP(A*DT) computed by means of a   !
!COM             Pade approximation. Matrix of dimensions (NX,NX).     !
!COM             Reference: Eqn. (1.47) and (1.48) [CTSM 2.3 Math      !
!COM             Guide,Dec. 2003, Kristensen, N.R.].                   !
!COM                                                                   !
```

Continued on next page...

Table A.27: Interface of subroutine `DEXPM` – Continued.

```
!COM        R1SM (output)                                                  !
!COM             Integral of EXP(A*DT)*R1M*TRANSPOSE(EXP(A*DT)) from       !
!COM             zero to DT. Matrix of dimensions (NX,NX).                 !
!COM             Reference: Eqn. (1.47) and (1.49) [CTSM 2.3 Math          !
!COM             Guide,Dec. 2003, Kristensen, N.R.].                       !
!COM                                                                       !
!COM        IDEG (input)                                                   !
!COM             Pade approximation order used in DGPADM (order of 6       !
!COM             is recommended by author of DGPADM).                      !
!COM                                                                       !
!COM        INFO (output)                                                  !
!COM             = 0:  successful exit.                                    !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.      !
!COM                                                                       !
!COM-----------------------------------------------------------------------!
```

Table A.28: Interface of subroutine `DISPLAY RESULTS`

```
!--------------------------------------SUBROUTINE DISPLAY_RESULTS-----!
!
!COM-----------------------------------------------------------------!
!COM  DISPLAY_RESULTS: WRITES RESULTS TO SCREEN                       !
!COM-----------------------------------------------------------------!
!COM                                                                  !
!COM   NAME                                                          !
!COM        'DISPLAY_RESULTS' - Subroutine                            !
!COM                                                                  !
!COM   PURPOSE                                                       !
!COM        To display the results for the fixed-effects and random-  !
!COM        effects.                                                  !
!COM                                                                  !
!COM   REFERENCE                                                     !
!COM        -                                                         !
!COM                                                                  !
!COM   INCLUDES                                                      !
!COM        '/INCLUDE/FORMAT'                                         !
!COM                                                                  !
!COM   CALLED BY                                                     !
!COM        Program MAIN                                              !
!COM                                                                  !
!COM   CALLS                                                         !
!COM        -                                                         !
!COM                                                                  !
!COM   F95 INTERFACE                                                 !
!COM        SUBROUTINE DISPLAY_RESULTS(THETAOBJ,ETAOBJ)               !
!COM                                                                  !
!COM        IMPLICIT NONE                                             !
!COM                                                                  !
!COM        TYPE(THETAOBJECT),INTENT(IN) :: THETAOBJ                  !
!COM        TYPE(ETAOBJECT),INTENT(IN) :: ETAOBJ(:)                   !
!COM        TYPE(PATIENT),INTENT(IN) :: POBJ(:)                       !
!COM        INTEGER,INTENT(OUT) :: INFO                               !
!COM                                                                  !
!COM   ARGUMENTS                                                     !
!COM        ETAOBJ (input)                                            !
!COM             Random-effects object (derived type 'ETAOBJECT')     !
!COM             containing results and optimization information.     !
!COM                                                                  !
!COM        INFO (output)                                             !
!COM             = 0:  successful exit.                               !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'. !
!COM                                                                  !
!COM        POBJ (input)                                              !
!COM             Patient object (derived type 'PATIENT') containing   !
!COM             individual patient data. Unchanged on exit.          !
!COM                                                                  !
!COM        THETAOBJ (input)                                          !
!COM             Fixed-effects object (derived type 'THETAOBJECT')    !
!COM             containing results and optimization information.     !
!COM                                                                  !
!COM-----------------------------------------------------------------!
```

Table A.29: Interface of subroutine ERRORSTAT

```
!-------------------------------------------SUBROUTINE ERRORSTAT-----!
!
!COM---------------------------------------------------------------------!
!COM  ERRORSTAT: IDENTIFIES INFORMATION TRANSFERRED BY INFO           !
!COM---------------------------------------------------------------------!
!COM                                                                  !
!COM   NAME                                                          !
!COM        'ERRORSTAT' - Subroutine                                  !
!COM                                                                  !
!COM   PURPOSE                                                       !
!COM        Translates and outputs information transferred by 'INFO'. !
!COM                                                                  !
!COM   REFERENCE                                                     !
!COM        -                                                         !
!COM                                                                  !
!COM   INCLUDES                                                      !
!COM        -                                                         !
!COM                                                                  !
!COM   CALLED BY                                                     !
!COM        Program MAIN                                              !
!COM                                                                  !
!COM   CALLS                                                         !
!COM        -                                                         !
!COM                                                                  !
!COM   F95 INTERFACE                                                 !
!COM        SUBROUTINE ERRORSTAT(INFO)                                !
!COM                                                                  !
!COM        IMPLICIT NONE                                             !
!COM                                                                  !
!COM        INTEGER,INTENT(IN) :: INFO                                !
!COM                                                                  !
!COM   ARGUMENTS                                                     !
!COM        INFO (input)                                              !
!COM             Integer value of INFO depends on type of error. If   !
!COM             no error, INFO equals zero, i.e.:                    !
!COM                                                                  !
!COM             = 0:  successful exit.                               !
!COM             > 0:  unsuccessful exit                              !
!COM                                                                  !
!COM---------------------------------------------------------------------!
```

Table A.30: Interface of subroutine `FAIAPLL`

```
!-----------------------------------------------FUNCTION FAIAPLL-----!
!
!COM-------------------------------------------------------------------!
!COM  FUNCTION FAIAPLL: RETURNS NEGATIVE APPROXIMATE INDIVIDUAL A     !
!COM                    POSTERIORI LOG-LIKELIHOOD 'AIAPLL'            !
!COM-------------------------------------------------------------------!
!COM                                                                  !
!COM   NAME                                                          !
!COM        'FAIAPLL' - Pure function                                 !
!COM                                                                  !
!COM   PURPOSE                                                       !
!COM        Returns the negative approximate individual a posteriori  !
!COM        log-likelihood value 'AIAPLL' given the individual        !
!COM        maximum likelilood value 'LL' obtained in the Kalman      !
!COM        filtering procedure (LTI_KALMAN/LTV_KALMAN/NL_KALMAN).    !
!COM                                                                  !
!COM   REFERENCE                                                     !
!COM        Eqn. (15), [Overgaard et. al., 'Non-linear mixed-effects  !
!COM        models with stochastic differential equations', 2005]     !
!COM                                                                  !
!COM   INCLUDES                                                      !
!COM        -                                                         !
!COM                                                                  !
!COM   CALLED BY                                                     !
!COM        Subroutine AIAPLL                                         !
!COM                                                                  !
!COM   CALLS                                                         !
!COM        Subroutine DTRM                                           !
!COM                                                                  !
!COM   F95 INTERFACE                                                 !
!COM        REAL(8) FUNCTION FAIAPLL(LL,ETA,OMEGA,INFO)               !
!COM                                                                  !
!COM        USE MOTYPES                                               !
!COM        USE MOPARAMS                                              !
!COM        USE SUNPERF                                               !
!COM        IMPLICIT NONE                                             !
!COM                                                                  !
!COM        INTEGER,INTENT(OUT) :: INFO                               !
!COM        REAL(8),INTENT(IN) :: LL                                  !
!COM        REAL(8),DIMENSION(NETA),INTENT(IN) :: ETA                 !
!COM        REAL(8),DIMENSION(NETA,NETA),INTENT(IN) :: OMEGA          !
!COM                                                                  !
!COM   ARGUMENTS                                                     !
!COM        ETA (input)                                               !
!COM             Random-effects vector - vector dimension (NETA).     !
!COM                                                                  !
!COM        FAIAPLL (returned)                                        !
!COM             Negative approximate individual a posteriori log-    !
!COM             likelihood value. Scalar.                            !
!COM                                                                  !
!COM        INFO (output)                                             !
!COM             = 0:  successful exit.                               !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'. !
!COM                                                                  !
```

Continued on next page...

Table A.30: Interface of subroutine `FAIAPLL` – Continued.

```
!COM        OMEGA (input)                                                 !
!COM             Variance-covariance matrix of the random-effects ETA     !
!COM             in the second stage model defined by:                    !
!COM                                                                      !
!COM                #3: PHI = h(THETA,Z)*EXP(ETA), ETA~N(0,OMEGA)         !
!COM                                                                      !
!COM             Matrix of dimension (NETA,NETA).                         !
!COM                                                                      !
!COM        LL (input)                                                    !
!COM             Minus individual log-likelihood value. Objective         !
!COM             function for optimization of random-effects ETA.         !
!COM             Scalar.                                                  !
!COM                                                                      !
!COM----------------------------------------------------------------------!
```

Table A.31: Interface of subroutine FAPL

```
!-------------------------------------------------FUNCTION FAPL-----!
!
!COM---------------------------------------------------------------------!
!COM  COMPUTATION OF INDIVIDUAL CONTRIBUTION TO THE NEGATIVE          !
!COM  APPROXIMATE POPULATION LOG-LIKELIHOOD                           !
!COM---------------------------------------------------------------------!
!COM                                                                  !
!COM   NAME                                                          !
!COM        'FAPL' - Function                                         !
!COM                                                                  !
!COM   PURPOSE                                                       !
!COM        Returns the contribution to negative approximate popu-    !
!COM        lation log-likehood value 'APL' of an individual given    !
!COM        the approximate individual a posteriori log-likelihood    !
!COM         value 'AIAPLL' and its Hessian matrix.                   !
!COM                                                                  !
!COM   REFERENCE                                                      !
!COM        Eqn. (11), [Mortensen, S., 'NLME log-likelihood function',!
!COM        2006-11-06]                                               !
!COM                                                                  !
!COM   INCLUDES                                                      !
!COM        -                                                         !
!COM                                                                  !
!COM   CALLED BY                                                     !
!COM        Subroutine APLDAPL                                        !
!COM        Subroutine CNTDIFF_APL                                    !
!COM                                                                  !
!COM   CALLS                                                         !
!COM        Subroutine DTRM                                           !
!COM                                                                  !
!COM   F95 INTERFACE                                                 !
!COM        REAL(8) FUNCTION FAPL(AIAPLL,HESSIAN,INFO)                !
!COM                                                                  !
!COM        USE MOTYPES                                               !
!COM        USE MOPARAMS                                              !
!COM        USE SUNPERF                                               !
!COM        IMPLICIT NONE                                             !
!COM                                                                  !
!COM        INTEGER,INTENT(OUT) :: INFO                               !
!COM        REAL(8),INTENT(IN) :: AIAPLL,HESSIAN(:,:)                 !
!COM        INTEGER,DIMENSION(NETA) :: PIVETA                         !
!COM                                                                  !
!COM   ARGUMENTS                                                     !
!COM        AIAPLL (input)                                            !
!COM             Negative approximate individual a posteriori log-    !
!COM             likelihood value. Scalar.                            !
!COM                                                                  !
!COM        FAPL (returned)                                           !
!COM             Individual contribution to negative approximate      !
!COM             population a posteriori log-likelihood value. Scalar.!
!COM                                                                  !
```

Continued on next page...

Table A.31: Interface of subroutine `FAPL` – Continued.

```
!COM        HESSIAN (input)                                              !
!COM             The Hessian matrix corresponding to AIAPLL - matrix     !
!COM             of dimension (NETA,NETA).                               !
!COM                                                                     !
!COM        INFO (output)                                                !
!COM             = 0:  successful exit.                                  !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.    !
!COM                                                                     !
!COM-------------------------------------------------------------------!
```

Table A.32: Interface of subroutine `FIDNM`

```
!-------------------------------------------------FUNCTION FIDNM-----!
!
!COM-----------------------------------------------------------------!
!COM  FUNCTION IDMN: IDENTITY MATRIX                                 !
!COM-----------------------------------------------------------------!
!COM                                                                 !
!COM   NAME                                                         !
!COM        'FIDNM' - Pure function                                  !
!COM                                                                 !
!COM   PURPOSE                                                      !
!COM        Creates an identity matrix of dimension 'N'              !
!COM                                                                 !
!COM   REFERENCE                                                    !
!COM        -                                                        !
!COM                                                                 !
!COM   INCLUDES                                                     !
!COM        -                                                        !
!COM                                                                 !
!COM   CALLED BY                                                    !
!COM        Subroutine LTI_KALMAN                                    !
!COM        Subroutine LTV_KALMAN                                    !
!COM                                                                 !
!COM   CALLS                                                        !
!COM        -                                                        !
!COM                                                                 !
!COM   F95 INTERFACE                                                !
!COM        REAL(8) PURE FUNCTION FIDNM(N)                           !
!COM                                                                 !
!COM        IMPLICIT NONE                                            !
!COM                                                                 !
!COM        INTEGER,INTENT(IN) :: N                                  !
!COM                                                                 !
!COM   ARGUMENTS                                                    !
!COM        N (input)                                                !
!COM             Dimension of identity matrix. Unchanged on exit.    !
!COM                                                                 !
!COM-----------------------------------------------------------------!
```

Table A.33: Interface of subroutine `FOMEGA`

```
!-------------------------------------------------FUNCTION FOMEGA-----!
!
!COM-----------------------------------------------------------------!
!COM  FUNCTION FOMEGA: RANDOM-EFFECTS VARIANCE-COVARIANCE MATRIX      !
!COM-----------------------------------------------------------------!
!COM                                                                  !
!COM   NAME                                                          !
!COM        'FOMEGA' - Pure function                                  !
!COM                                                                  !
!COM   PURPOSE                                                       !
!COM        Defines random-effects variance-covariance matrix 'OMEGA',!
!COM        where random-effects ETA ~ N(0,OMEGA). The random-effects !
!COM        vector ETA is expressed in the second-stage model:        !
!COM                                                                  !
!COM           #1: PHI = h(THETA,Z)*EXP(ETA), ETA~N(0,OMEGA)          !
!COM                                                                  !
!COM        Where:                                                    !
!COM                                                                  !
!COM             OMEGA = OMEGA(THETA)                                 !
!COM                                                                  !
!COM   REFERENCE                                                      !
!COM        Eqn. (4), [Overgaard et. al., 'Non-linear mixed-effects   !
!COM        models with stochastic differential equations', 2005]     !
!COM                                                                  !
!COM   INCLUDES                                                       !
!COM        '/USER/OMEGAMAT'                                          !
!COM                                                                  !
!COM   CALLED BY                                                     !
!COM        Subroutine APL                                            !
!COM        Subroutine APLDAPL                                        !
!COM        Subroutine CNTDIFF_APL                                    !
!COM                                                                  !
!COM   CALLS                                                         !
!COM        -                                                         !
!COM                                                                  !
!COM   F95 INTERFACE                                                  !
!COM        REAL(8) PURE FUNCTION FOMEGA(THETA)                       !
!COM                                                                  !
!COM        USE MOPARAMS                                              !
!COM        IMPLICIT NONE                                             !
!COM                                                                  !
!COM        REAL(8),INTENT(IN) :: THETA(:)                            !
!COM                                                                  !
!COM   ARGUMENTS                                                     !
!COM        THETA (input)                                             !
!COM             Fixed-effects vector - vector dimension (NTHETA).    !
!COM                                                                  !
!COM-----------------------------------------------------------------!
```

Table A.34: Interface of subroutine FPHI

```
!-----------------------------------------------------FUNCTION FPHI-----!
!
!COM---------------------------------------------------------------------!
!COM  FUNCTION FPHI: INDIVIDUAL PARAMETERS VECTOR                       !
!COM---------------------------------------------------------------------!
!COM                                                                    !
!COM   NAME                                                            !
!COM        'FPHI' - Pure function                                      !
!COM                                                                    !
!COM   PURPOSE                                                         !
!COM        To define the individual parameters vector PHI, which is    !
!COM        expressed in the second-stage model:                        !
!COM                                                                    !
!COM           #1: PHI = h(THETA,Z)*EXP(ETA), ETA~N(0,OMEGA)            !
!COM                                                                    !
!COM        Where:                                                      !
!COM                                                                    !
!COM             OMEGA = OMEGA(THETA)                                   !
!COM                                                                    !
!COM   REFERENCE                                                        !
!COM        Eqn. (4), [Overgaard et. al., 'Non-linear mixed-effects     !
!COM        models with stochastic differential equations', 2005]       !
!COM                                                                    !
!COM   INCLUDES                                                         !
!COM        '/USER/PHIVEC'                                              !
!COM                                                                    !
!COM   CALLED BY                                                        !
!COM        Subroutine LLDLL                                            !
!COM        Subroutine HESSIAN_AIAPLL                                   !
!COM        Subroutine CNTDIFF_AIAPLL                                   !
!COM                                                                    !
!COM   CALLS                                                            !
!COM        -                                                           !
!COM                                                                    !
!COM   F95 INTERFACE                                                    !
!COM        REAL(8) PURE FUNCTION FPHI(THETA,ETA,Z)                     !
!COM                                                                    !
!COM        USE MOPARAMS                                                !
!COM        IMPLICIT NONE                                               !
!COM                                                                    !
!COM        REAL(8),INTENT(IN) :: THETA(:),ETA(:)                       !
!COM        REAL(8),INTENT(IN),OPTIONAL :: Z(:)                         !
!COM                                                                    !
!COM   ARGUMENTS                                                        !
!COM        ETA (input)                                                 !
!COM             Random-effects vector - vector dimension (NETA).       !
!COM                                                                    !
!COM        THETA (input)                                               !
!COM             Fixed-effects vector - vector dimension (NTHETA).      !
!COM                                                                    !
!COM        FPHI (returned)                                             !
!COM             Individual parameters vector - vector dimension        !
!COM             (NPHI).                                                !
!COM                                                                    !
!COM        Z (input, optional)                                         !
!COM             Covariates vector - vector dimension (NZ).             !
!COM                                                                    !
!COM---------------------------------------------------------------------!
```

Table A.35: Interface of subroutine FPI

```
!-----------------------------------------------------FUNCTION FPI-----!
!
!COM---------------------------------------------------------------------!
!COM  FUNCTION FPI: DEFINES MAGNITUDE OF SYSTEM VARIABILITY MATRIX    !
!COM---------------------------------------------------------------------!
!COM                                                                    !
!COM   NAME                                                            !
!COM        'FPI' - Pure function                                       !
!COM                                                                    !
!COM   PURPOSE                                                         !
!COM        Defines magnitude of system variability matrix 'PI'         !
!COM                                                                    !
!COM   REFERENCE                                                       !
!COM        -                                                           !
!COM                                                                    !
!COM   INCLUDES                                                        !
!COM        '/USER/PIMAT'                                               !
!COM                                                                    !
!COM   CALLED BY                                                       !
!COM        Subroutine LTI_KALMAN                                       !
!COM        Subroutine LTV_KALMAN                                       !
!COM                                                                    !
!COM   CALLS                                                           !
!COM        -                                                           !
!COM                                                                    !
!COM   F95 INTERFACE                                                   !
!COM        REAL(8) PURE FUNCTION FPI(PHI,T,U)                          !
!COM                                                                    !
!COM        USE MOPARAMS                                                !
!COM        IMPLICIT NONE                                               !
!COM                                                                    !
!COM        REAL(8),INTENT(IN) :: PHI(:)                                !
!COM                                                                    !
!COM   ARGUMENTS                                                       !
!COM        PHI (input)                                                 !
!COM             Individual parameters vector - vector dimension        !
!COM             (NPHI). Unchanged on exit.                             !
!COM                                                                    !
!COM        T (input, optional)                                         !
!COM             Time vector - dimension (NT). Optional, only appli-    !
!COM             cable for time-varying models.                         !
!COM                                                                    !
!COM        U (input, optional)                                         !
!COM             Input variable vector - dimension (NU). Optional,      !
!COM             only applicable for time-varying models.               !
!COM                                                                    !
!COM---------------------------------------------------------------------!
```

Table A.36: Interface of subroutine `FSIGMA`

```
!-------------------------------------------------FUNCTION FSIGMA-----!
!
!COM-------------------------------------------------------------------!
!COM  FUNCTION FSIGMA: MEASUREMENT NOISE VARIANCE-COVARIANCE MATRIX   !
!COM-------------------------------------------------------------------!
!COM                                                                  !
!COM   NAME                                                          !
!COM        'FSIGMA' - Pure function                                  !
!COM                                                                  !
!COM   PURPOSE                                                       !
!COM        Defines variance-covariance matrix 'SIGMA' of measurement !
!COM        noise term 'e', where e~N(0,SIGMA).                       !
!COM                                                                  !
!COM        The linear models (LTI/LTV) are defined by:               !
!COM                                                                  !
!COM           #1: dX = [A*X + B*U]dt + PI*dW                         !
!COM                                                                  !
!COM           #2:  Y =  C*X + D*U + e, e~N(0,SIGMA)                  !
!COM                                                                  !
!COM        For linear time-invariant (LTI) model:                    !
!COM                                                                  !
!COM                A = A(PHI)         B = B(PHI)                    !
!COM                C = A(PHI)         D = D(PHI)                    !
!COM                PI = PI(PHI)       SIGMA = SIGMA(PHI)            !
!COM                                                                  !
!COM        For linear time-varying (LTV) model:                      !
!COM                                                                  !
!COM                A = A(X,U,t,PHI)   B = B(X,U,t,PHI)              !
!COM                C = A(X,U,t,PHI)   D = D(X,U,t,PHI)              !
!COM                PI= PI(U,t,PHI)    SIGMA = SIGMA(U,t,PHI)        !
!COM                                                                  !
!COM   REFERENCE                                                      !
!COM        -                                                         !
!COM                                                                  !
!COM   INCLUDES                                                       !
!COM        '/USER/SIGMAMAT'                                          !
!COM                                                                  !
!COM   CALLED BY                                                      !
!COM        Subroutine LTI_KALMAN                                     !
!COM        Subroutine LTV_KALMAN                                     !
!COM                                                                  !
!COM   CALLS                                                          !
!COM        -                                                         !
!COM                                                                  !
!COM   F95 INTERFACE                                                  !
!COM        REAL(8) PURE FUNCTION FSIGMA(PHI,T,U)                     !
!COM                                                                  !
!COM        USE MOPARAMS                                              !
!COM        IMPLICIT NONE                                             !
!COM                                                                  !
!COM        REAL(8),INTENT(IN) :: PHI(:)                              !
!COM                                                                  !
```

Continued on next page...

Table A.36: Interface of subroutine `FSIGMA` – Continued.

```
!COM   ARGUMENTS                                                        !
!COM        PHI (input)                                                 !
!COM             Individual parameters vector - vector dimension        !
!COM             (NPHI). Unchanged on exit.                             !
!COM                                                                    !
!COM        T (input, optional)                                         !
!COM             Time vector - dimension (NT). Optional, only appli-    !
!COM             cable for time-varying models.                         !
!COM                                                                    !
!COM        U (input, optional)                                         !
!COM             Input variable vector - dimension (NU). Optional,      !
!COM             only applicable for time-varying models.               !
!COM                                                                    !
!COM------------------------------------------------------------------!
```

Table A.37: Interface of subroutine FTHETA

```
!-------------------------------------------------FUNCTION FTHETA-----!
!
!COM-----------------------------------------------------------------!
!COM  FUNCTION FTHETA: DEFINES INITIAL GUESS OF FIXED-EFFECTS        !
!COM-----------------------------------------------------------------!
!COM                                                                 !
!COM   NAME                                                         !
!COM        'FTHETA' - Pure function                                 !
!COM                                                                 !
!COM   PURPOSE                                                      !
!COM        Defines initial fixed-effects parameters 'THETA'         !
!COM                                                                 !
!COM   REFERENCE                                                    !
!COM        -                                                        !
!COM                                                                 !
!COM   INCLUDES                                                     !
!COM        '/USER/THETAVEC'                                         !
!COM                                                                 !
!COM   CALLED BY                                                    !
!COM        Subroutine INIT_THETAOBJECT                              !
!COM                                                                 !
!COM   CALLS                                                        !
!COM        -                                                        !
!COM                                                                 !
!COM   F95 INTERFACE                                                !
!COM        REAL(8) PURE FUNCTION FTHETA()                           !
!COM                                                                 !
!COM        USE MOPARAMS                                             !
!COM        IMPLICIT NONE                                            !
!COM                                                                 !
!COM   ARGUMENTS                                                    !
!COM        -                                                        !
!COM                                                                 !
!COM-----------------------------------------------------------------!
```

Table A.38: Interface of subroutine `HESSIAN AIAPLL`

```
!---------------------------------------SUBROUTINE HESSIAN_AIAPLL-----!
!
!COM-----------------------------------------------------------------!
!COM  HESSIAN_AIAPLL: COMPUTES HESSIAN MATRIX OF APPROXIMATE         !
!COM  INDIVIDUAL A POSTERIORI LOG-LIKELIHOOD 'AIAPLL'                !
!COM-----------------------------------------------------------------!
!COM                                                                 !
!COM   NAME                                                         !
!COM        'HESSIAN_AIAPLL' - Subroutine                            !
!COM                                                                 !
!COM   PURPOSE                                                      !
!COM        To compute the Hessian matrix of the approximate indivi- !
!COM        individual log-likelihood 'AIAPLL', which is used to find !
!COM        the approximate population likelihood 'APL'.             !
!COM                                                                 !
!COM   REFERENCE                                                    !
!COM        Eqn. (19), [Overgaard et. al., 'Non-linear mixed-effects  !
!COM        models with stochastic differential equations', 2005]    !
!COM                                                                 !
!COM   INCLUDES                                                     !
!COM        -                                                        !
!COM                                                                 !
!COM   CALLED BY                                                    !
!COM        Subroutine APLDAPL                                       !
!COM        Subroutine CNTDIFF_APL                                   !
!COM                                                                 !
!COM   CALLS                                                        !
!COM        Function FPHI                                            !
!COM        Module procedure MODEL (LTI_KALMAN/LTV_KALMAN/NL_KALMAN)  !
!COM                                                                 !
!COM   F95 INTERFACE                                                !
!COM        SUBROUTINE HESSIAN_AIAPLL(MODEL,HESSIAN,THETAOBJ,ETAOBJ,  !
!COM                                                  OMEGA,POBJ,INFO) !
!COM                                                                 !
!COM        USE MOTYPES                                              !
!COM        USE MOPARAMS                                             !
!COM        USE MODATA, ONLY: NID                                    !
!COM        IMPLICIT NONE                                            !
!COM                                                                 !
!COM        INTERFACE                                                !
!COM          SUBROUTINE MODEL(LL,KOBJ,PHI,ETA,OMEGA,POBJ,INFO)      !
!COM          USE SUNPERF                                            !
!COM          USE MOTYPES,ONLY: DOSE,PATIENT,KALOBJ                  !
!COM          USE MOIOS                                              !
!COM          IMPLICIT NONE                                          !
!COM          INTEGER,INTENT(OUT) :: INFO                            !
!COM          REAL(8),INTENT(IN) :: PHI(:),ETA(:),OMEGA(:,:)         !
!COM          TYPE(PATIENT),INTENT(IN) :: POBJ                       !
!COM          TYPE(KALOBJ),INTENT(INOUT) :: KOBJ                     !
!COM          REAL(8),INTENT(OUT) :: LL                              !
!COM          END SUBROUTINE MODEL                                   !
!COM        END INTERFACE                                            !
!COM                                                                 !
```

Continued on next page...

Table A.38: Interface of subroutine `HESSIAN AIAPLL` – Continued.

```
!COM        INTEGER,INTENT(OUT) :: INFO                                  !
!COM        REAL(8),INTENT(OUT) :: HESSIAN(NETA,NETA)                    !
!COM        REAL(8),INTENT(IN) :: OMEGA(:,:)                             !
!COM        TYPE(THETAOBJECT),INTENT(IN) :: THETAOBJ                     !
!COM        TYPE(ETAOBJECT),INTENT(IN) :: ETAOBJ                         !
!COM        TYPE(PATIENT),INTENT(IN) :: POBJ                             !
!COM                                                                     !
!COM   ARGUMENTS                                                        !
!COM        ETAOBJ (input)                                               !
!COM             Single random-effects object. Unchanged on exit.        !
!COM                                                                     !
!COM        INFO (output)                                                !
!COM             = 0:  successful exit.                                  !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.    !
!COM                                                                     !
!COM        HESSIAN (output)                                             !
!COM             Hessian matrix for the approximate individual a         !
!COM             posteriori likelihood function 'AIAPLL' - matrix        !
!COM             dimension (NETA,NETA).                                  !
!COM                                                                     !
!COM        MODEL (module procedure)                                     !
!COM             Kalman filtering procedure, which depends on the        !
!COM             model specification.                                    !
!COM                                                                     !
!COM             The parameter IMODEL dictates, which Kalman filter-     !
!COM             ing method should be used for modelling, i.e.:          !
!COM                                                                     !
!COM                 = 0:  MODEL = LTI_KALMAN                            !
!COM                 = 1:  MODEL = LTV_KALMAN                            !
!COM                 = 2:  MODEL =  NL_KALMAN                            !
!COM                                                                     !
!COM        OMEGA (input)                                                !
!COM             Variance-covariance matrix of the random-effects ETA    !
!COM             in the second stage model defined by:                   !
!COM                                                                     !
!COM                #1: PHI = h(THETA,Z)*EXP(ETA), ETA~N(0,OMEGA)        !
!COM                                                                     !
!COM             Matrix of dimension (NETA,NETA). Unchanged on exit.     !
!COM                                                                     !
!COM        POBJ (input)                                                 !
!COM             Patient object (derived type 'PATIENT') containing      !
!COM             individual patient data. Unchanged on exit.             !
!COM                                                                     !
!COM        THETAOBJ (input)                                             !
!COM             Fixed-effects object (derived type THETAOBJECT).        !
!COM             Unchanged on exit.                                      !
!COM                                                                     !
!COM-------------------------------------------------------------------!
```

Table A.39: Interface of subroutine `INIT ETAOBJECT`

```
!---------------------------------------SUBROUTINE INIT_ETAOBJECT-----!
!
!COM---------------------------------------------------------------------!
!COM                                                                     !
!COM   NAME                                                             !
!COM        'INIT_ETAOBJECT' - Subroutine                                !
!COM                                                                     !
!COM   PURPOSE                                                          !
!COM        Allocates variables in the random-effects object (derived !
!COM        type 'ETAOBJECT') given the number of random-effects       !
!COM        (NETA).                                                      !
!COM                                                                     !
!COM   REFERENCE                                                        !
!COM        -                                                            !
!COM                                                                     !
!COM   INCLUDES                                                         !
!COM        -                                                            !
!COM                                                                     !
!COM   CALLED BY                                                        !
!COM        Program MAIN                                                 !
!COM        Subroutine CNTDIFF_APL                                       !
!COM                                                                     !
!COM   CALLS                                                            !
!COM        Subroutine ALLOC_KALMANOBJECT                                !
!COM                                                                     !
!COM   F95 INTERFACE                                                    !
!COM                                                                     !
!COM        USE MOTYPES                                                  !
!COM        IMPLICIT NONE                                                !
!COM                                                                     !
!COM        INTEGER,INTENT(OUT) :: INFO                                  !
!COM        TYPE(PATIENT),INTENT(IN) :: POBJ                             !
!COM        TYPE(ETAOBJECT),INTENT(INOUT) :: ETAOBJ                      !
!COM                                                                     !
!COM   PARAMETERS                                                       !
!COM        ETAOBJ (input/output)                                        !
!COM             Random-effects object (derived type 'ETAOBJECT')        !
!COM             containing results and optimization information.        !
!COM                                                                     !
!COM        INFO (output)                                                !
!COM             = 0:  successful exit.                                  !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.  !
!COM                                                                     !
!COM        POBJ (input)                                                 !
!COM             Patient object (derived type 'PATIENT') containing      !
!COM             individual patient data. Unchanged on exit.             !
!COM                                                                     !
!COM---------------------------------------------------------------------!
```

Table A.40: Interface of subroutine `INIT THETAOBJECT`

```
!-----------------------------------SUBROUTINE INIT_THETAOBJECT-----!
!
!COM-------------------------------------------------------------------!
!COM                                                                   !
!COM   NAME                                                           !
!COM        'INIT_THETAOBJECT' - Subroutine                            !
!COM                                                                   !
!COM   PURPOSE                                                        !
!COM        Allocates variables in the fixed-effects object (derived  !
!COM        type 'THETAOBJECT') given the number of fixed-effects      !
!COM        (NTHETA).                                                  !
!COM                                                                   !
!COM   REFERENCE                                                      !
!COM        -                                                          !
!COM                                                                   !
!COM   INCLUDES                                                       !
!COM        -                                                          !
!COM                                                                   !
!COM   CALLED BY                                                      !
!COM        Program MAIN                                               !
!COM        Subroutine CNTDIFF_APL                                     !
!COM                                                                   !
!COM   CALLS                                                          !
!COM        -                                                          !
!COM                                                                   !
!COM   F95 INTERFACE                                                  !
!COM                                                                   !
!COM        USE MOTYPES                                                !
!COM        IMPLICIT NONE                                              !
!COM                                                                   !
!COM        INTEGER,INTENT(OUT)             :: INFO                    !
!COM        TYPE(THETAOBJECT),INTENT(INOUT) :: THETAOBJ                !
!COM                                                                   !
!COM   PARAMETERS                                                     !
!COM        THETAOBJ (input/output)                                    !
!COM             Fixed-effects object (derived type 'THETAOBJECT')     !
!COM             containing results and optimization information.      !
!COM                                                                   !
!COM        INFO (output)                                              !
!COM             = 0:  successful exit.                                !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.  !
!COM                                                                   !
!COM-------------------------------------------------------------------!
```

Table A.41: Interface of subroutine `LINEAR MODEL`

```
!----------------------------------------SUBROUTINE LINEAR_MODEL-----!
!
!COM-------------------------------------------------------------------!
!COM  LINEAR_MODEL                                                    !
!COM-------------------------------------------------------------------!
!COM                                                                  !
!COM   NAME                                                          !
!COM        'LINEAR_MODEL' - Subroutine                               !
!COM                                                                  !
!COM   PURPOSE                                                       !
!COM        Generates the linear model specified by user.             !
!COM                                                                  !
!COM        The linear model is expressed by:                         !
!COM                                                                  !
!COM           #1: dX = [A*X + B*U]dt + PI*dW                         !
!COM                                                                  !
!COM           #2:  Y =  C*X + D*U + e, e~N(0,SIGMA)                  !
!COM                                                                  !
!COM        For linear time-invariant (LTI) model:                    !
!COM                                                                  !
!COM                A = A(PHI)         B = B(PHI)                    !
!COM                C = A(PHI)         D = D(PHI)                    !
!COM                PI = PI(PHI)       SIGMA = SIGMA(PHI)            !
!COM                                                                  !
!COM        For linear time-varying (LTV) model:                      !
!COM                                                                  !
!COM                A = A(X,U,t,PHI)   B = B(X,U,t,PHI)              !
!COM                C = A(X,U,t,PHI)   D = D(X,U,t,PHI)              !
!COM                PI= PI(U,t,PHI)    SIGMA = SIGMA(U,t,PHI)        !
!COM                                                                  !
!COM   REFERENCE                                                      !
!COM        Linear time-varying model: Eqn. (1.3)-(1.4)               !
!COM        Linear time-invariant model: Eqn. (1.5)-(1.6)             !
!COM        [CTSM 2.3 Math Guide, Dec. 2003, Kristensen, N.R.]        !
!COM                                                                  !
!COM   INCLUDES                                                      !
!COM        '/INCLUDE/AMAT'                                           !
!COM        '/INCLUDE/BMAT'                                           !
!COM        '/INCLUDE/CMAT'                                           !
!COM        '/INCLUDE/DMAT'                                           !
!COM                                                                  !
!COM   CALLED BY                                                     !
!COM        Subroutine LTI_KALMAN                                     !
!COM        Subroutine LTV_KALMAN                                     !
!COM                                                                  !
!COM   CALLS                                                         !
!COM        -                                                         !
!COM                                                                  !
!COM   F95 INTERFACE                                                 !
!COM        SUBROUTINE LINEAR_MODEL(PHI,A,B,C,D,T,X,U)                !
!COM                                                                  !
!COM        USE MOPARAMS                                              !
!COM        USE MOIOS                                                 !
!COM        IMPLICIT NONE                                             !
!COM                                                                  !
```

Continued on next page...

Table A.41: Interface of subroutine `LINEAR MODEL` – Continued.

```
!COM        REAL(8),INTENT(IN) :: PHI(:)                                  !
!COM        REAL(8),INTENT(OUT) :: A(:,:),B(:,:),C(:,:),D(:,:)            !
!COM        REAL(8),INTENT(IN), OPTIONAL :: X(:),U(:),T                   !
!COM                                                                      !
!COM   ARGUMENTS                                                         !
!COM        A (output)                                                    !
!COM             State coefficient matrix in state eqn., see (#1)         !
!COM             - matrix dimension (NX,NX).                              !
!COM                                                                      !
!COM        B (output)                                                    !
!COM             Output coefficient matrix in state eqn., see (#1)        !
!COM             - matrix dimension (NX,NU). Only defined when NU>0.      !
!COM                                                                      !
!COM        C (output)                                                    !
!COM             State coefficient matrix in observation eqn.,            !
!COM             see (#2) - matrix dimension (NY,NX).                     !
!COM                                                                      !
!COM        D (output)                                                    !
!COM             Output coefficient matrix in observation eqn.,           !
!COM             see (#2) - matrix dimension (NY,NU). Only defined        !
!COM             when NU>0.                                               !
!COM                                                                      !
!COM        PHI (input)                                                   !
!COM             Individual parameters vector - vector dimension          !
!COM             (NPHI). Unchanged on exit.                               !
!COM                                                                      !
!COM        T (input, optional)                                           !
!COM             Time vector - dimension (NT). Optional, only appli-      !
!COM             cable for time-varying models.                           !
!COM                                                                      !
!COM        U (input, optional)                                           !
!COM             Input variable vector - dimension (NU). Optional,        !
!COM             only applicable for time-varying models.                 !
!COM                                                                      !
!COM        X (input, optional)                                           !
!COM             State variable vector - dimension (NX). Optional,        !
!COM             only applicable for time-varying models.                 !
!COM                                                                      !
!COM----------------------------------------------------------------------!
```

Table A.42: Interface of subroutine LLDLL

```
!-----------------------------------------------SUBROUTINE LLDLL-----!
!
!COM-------------------------------------------------------------------!
!COM  FUNCTION VALUE (LL) AND GRADIENT (GRAD_LL) FOR APPROXIMATE      !
!COM  INDIVIDUAL LOG-LIKELIHOOD                                       !
!COM-------------------------------------------------------------------!
!COM                                                                  !
!COM   NAME                                                          !
!COM        'LLDLL' - Subroutine                                      !
!COM                                                                  !
!COM   PURPOSE                                                       !
!COM        Given a set of random-effects parameters TMP_ETA, subrou- !
!COM        tine LLDLL computes the approximate individual log-like-  !
!COM        lihood (LL) and corresponding gradient (GRAD_LL) required !
!COM        by the parameter mimimization procedure UCMINF_AIAPLL.    !
!COM                                                                  !
!COM        The fixed-effects parameters 'THETA' are contained in the !
!COM        theta object 'THETAOBJ' (derived type 'THETAOBJECT').     !
!COM        The individual random-effects 'ETA' and the approximate   !
!COM        individual a posteriori log-likelihood are stored in the  !
!COM        eta objects 'ETAOBJ' (derived type 'ETAOBJECT').          !
!COM                                                                  !
!COM   REFERENCE                                                     !
!COM        -                                                         !
!COM                                                                  !
!COM   INCLUDES                                                      !
!COM        '/INCLUDE/FORMAT'                                         !
!COM                                                                  !
!COM   CALLED BY                                                     !
!COM        Subroutine UCMINF_AIAPLL                                  !
!COM        Subroutine CHKDFN_AIAPLL                                  !
!COM        Subroutine SLINE_AIAPLL                                   !
!COM                                                                  !
!COM   CALLS                                                         !
!COM        Function FPHI                                             !
!COM        Module procedure MODEL (LTI_KALMAN/LTV_KALMAN/NL_KALMAN)  !
!COM        Subroutine CNTDIFF_AIAPLL                                 !
!COM                                                                  !
!COM   F95 INTERFACE                                                 !
!COM        SUBROUTINE LLDLL(N,TMP_ETA,GRAD_LL,LL,MODEL,THETAOBJ,     !
!COM                                         ETAOBJ,POBJ,OMEGA,INFO) !
!COM                                                                  !
!COM        USE MOTYPES                                               !
!COM        USE MOPARAMS                                              !
!COM        USE MODATA, ONLY: NID                                     !
!COM        IMPLICIT NONE                                             !
!COM                                                                  !
!COM        INTERFACE                                                 !
!COM          SUBROUTINE MODEL(LL,KOBJ,PHI,ETA,OMEGA,POBJ,INFO)       !
!COM          USE SUNPERF                                             !
!COM          USE MOTYPES,ONLY: DOSE,PATIENT,KALOBJ                   !
!COM          USE MOIOS                                               !
!COM          IMPLICIT NONE                                           !
```

Continued on next page...

Table A.42: Interface of subroutine LLDLL – Continued.

```
!COM          INTEGER,INTENT(OUT) :: INFO                                  !
!COM          REAL(8),INTENT(IN) :: PHI(:),ETA(:),OMEGA(:,:)               !
!COM          TYPE(PATIENT),INTENT(IN) :: POBJ                             !
!COM          TYPE(KALOBJ),INTENT(INOUT) :: KOBJ                           !
!COM          REAL(8),INTENT(OUT) :: LL                                    !
!COM          END SUBROUTINE MODEL                                         !
!COM        END INTERFACE                                                  !
!COM                                                                       !
!COM        INTEGER,INTENT(IN) :: N                                        !
!COM        REAL(8),INTENT(IN) :: TMP_ETA(:),OMEGA(:,:)                    !
!COM        REAL(8),INTENT(OUT) :: GRAD_LL(:),LL                           !
!COM        TYPE(THETAOBJECT),INTENT(IN) :: THETAOBJ                       !
!COM        TYPE(ETAOBJECT),INTENT(INOUT) :: ETAOBJ                        !
!COM        TYPE(PATIENT),INTENT(IN) :: POBJ                               !
!COM        INTEGER,INTENT(OUT) :: INFO                                    !
!COM                                                                       !
!COM   ARGUMENTS                                                          !
!COM        ETAOBJ (input/output)                                          !
!COM             Single random-effects object. On exit, ETAOBJ has         !
!COM             been updated with filtering information stored in         !
!COM             the Kalman object 'KOBJ' (derived type KALOBJ).           !
!COM                                                                       !
!COM        GRAD_LL (output)                                               !
!COM             Function gradient required by UCMINF_AIAPL optimi-        !
!COM             zation procedure, i.e. gradient of the approximate        !
!COM             individual log-likelihood value.                          !
!COM                                                                       !
!COM        INFO (output)                                                  !
!COM             = 0:  successful exit.                                    !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.      !
!COM                                                                       !
!COM        LL (output)                                                    !
!COM             Function value required by UCMINF_AIAPLLoptimiza-         !
!COM             tion procedure, i.e. the approximate individual           !
!COM             log-likelihood value.                                     !
!COM                                                                       !
!COM        MODEL (module procedure)                                       !
!COM             Kalman filtering procedure, which depends on the          !
!COM             model specification.                                      !
!COM                                                                       !
!COM             The parameter IMODEL dictates, which Kalman filter-       !
!COM             ing method should be used for modelling, i.e.:            !
!COM                                                                       !
!COM                 = 0:  MODEL = LTI_KALMAN                              !
!COM                 = 1:  MODEL = LTV_KALMAN                              !
!COM                 = 2:  MODEL =  NL_KALMAN                              !
!COM                                                                       !
!COM        N (input)                                                      !
!COM             Number of random-effects parameters (=NETA)               !
!COM                                                                       !
!COM        OMEGA (input)                                                  !
!COM             Variance-covariance matrix of the random-effects ETA      !
!COM             in the second stage model defined by:                     !
!COM                                                                       !
!COM                #1: PHI = h(THETA,Z)*EXP(ETA), ETA~N(0,OMEGA)          !
!COM                                                                       !
!COM             Matrix of dimension (NETA,NETA). Unchanged on exit.       !
!COM                                                                       !
```

Continued on next page...

Table A.42: Interface of subroutine LLDLL – Continued.

```
!COM        POBJ (input)                                                  !
!COM             Patient object (derived type 'PATIENT') containing       !
!COM             individual patient data. Unchanged on exit.              !
!COM                                                                      !
!COM        TMP_ETA (input)                                               !
!COM             Current random-effects parameters for which the          !
!COM             approximate individual log-likelihood 'LL' and its       !
!COM             gradient GRAD_LL is computed. TMP_ETA is supplied        !
!COM             the parameter optimization procedure UCMINF_AIAPLL.      !
!COM                                                                      !
!COM        THETAOBJ (input)                                              !
!COM             Fixed-effects object (derived type THETAOBJECT).         !
!COM             Unchanged on exit.                                       !
!COM                                                                      !
!COM-------------------------------------------------------------------!
```

Table A.43: Interface of subroutine `LTI KALMAN`

```
!-----------------------------------------SUBROUTINE LTI_KALMAN-----!
!
!COM-----------------------------------------------------------------!
!COM  KALMAN FILTER FOR LINEAR TIME-INVARIANT MODEL                  !
!COM-----------------------------------------------------------------!
!COM                                                                 !
!COM   NAME                                                         !
!COM        'LTI_KALMAN' - Subroutine                                !
!COM                                                                 !
!COM   PURPOSE                                                      !
!COM        Kalman filter for the linear time-invariant (LTI) model: !
!COM                                                                 !
!COM           #1: dX = [A*X + B*U]dt + PI*dW                        !
!COM                                                                 !
!COM           #2:  Y =  C*X + D*U + e, e~N(0,SIGMA)                 !
!COM                                                                 !
!COM        Where                                                    !
!COM                                                                 !
!COM                A = A(PHI)         B = B(PHI)                   !
!COM                C = A(PHI)         D = D(PHI)                   !
!COM                PI = PI(PHI)       SIGMA = SIGMA(PHI)           !
!COM                                                                 !
!COM   REFERENCE                                                    !
!COM        Eqn. (1.5) and (1.6) [CTSM 2.3 Math Guide, Dec. 2003,    !
!COM        Kristensen, N.R.].                                       !
!COM                                                                 !
!COM   INCLUDES                                                     !
!COM        '/INCLUDE/FORMAT'                                        !
!COM                                                                 !
!COM   CALLED BY                                                    !
!COM        Subroutine CNTDIFF_AIAPLL                                !
!COM        Subroutine LLDLL                                         !
!COM                                                                 !
!COM   CALLS                                                        !
!COM        Function FPI                                             !
!COM        Function FSIGMA                                          !
!COM        Function IDMN                                            !
!COM        Subroutine DEXPM                                         !
!COM        Subroutine LINEAR_MODEL                                  !
!COM        Subroutine DTRM                                          !
!COM                                                                 !
!COM   F95 INTERFACE                                                !
!COM        SUBROUTINE LTI_KALMAN(LL,KOBJ,PHI,ETA,OMEGA,POBJ,INFO)   !
!COM                                                                 !
!COM        USE SUNPERF                                              !
!COM        USE MOTYPES, ONLY: DOSE, PATIENT, KALOBJ                 !
!COM        USE MOIOS                                                !
!COM        IMPLICIT NONE                                            !
!COM                                                                 !
!COM        INTEGER,INTENT(OUT) :: INFO                              !
!COM        REAL(8),INTENT(IN) :: PHI(:),ETA(:),OMEGA(:,:)           !
!COM        TYPE(PATIENT),INTENT(IN) :: POBJ                         !
!COM        TYPE(KALOBJ),INTENT(INOUT) :: KOBJ                       !
!COM        REAL(8),INTENT(OUT) :: LL                                !
!COM                                                                 !
```

Continued on next page...

Table A.43: Interface of subroutine `LTI KALMAN` – Continued.

```
!COM   ARGUMENTS                                                        !
!COM        ETA (input)                                                 !
!COM             Random-effects vector - vector dimension (NETA).       !
!COM                                                                    !
!COM        INFO (output)                                               !
!COM             = 0:  successful exit.                                 !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.   !
!COM                                                                    !
!COM        KOBJ (input/output)                                         !
!COM             Kalman object (derived type 'KALOBJ') containing       !
!COM             results from Kalman filtering.                         !
!COM                                                                    !
!COM        OMEGA (input)                                               !
!COM             Variance-covariance matrix of the random-effects ETA   !
!COM             in the second stage model defined by:                  !
!COM                                                                    !
!COM                #3: PHI = h(THETA,Z)*EXP(ETA), ETA~N(0,OMEGA)       !
!COM                                                                    !
!COM             Matrix of dimension (NETA,NETA).                       !
!COM                                                                    !
!COM        PHI (input)                                                 !
!COM             Individual parameters vector - vector dimension        !
!COM             (NPHI). Unchanged on exit.                             !
!COM                                                                    !
!COM        POBJ (input)                                                !
!COM             Patient object (derived type 'PATIENT') containing     !
!COM             individual patient data.                               !
!COM                                                                    !
!COM        LL (output)                                                 !
!COM             Minus log-likelihood value. Scalar.                    !
!COM                                                                    !
!COM-----------------------------------------------------------------------!
```

Table A.44: Interface of subroutine `LTV_KALMAN`

```
!-----------------------------------------SUBROUTINE LTV_KALMAN-----!
!
!COM-----------------------------------------------------------------!
!COM  KALMAN FILTER FOR LINEAR TIME-VARYING MODEL                    !
!COM-----------------------------------------------------------------!
!COM                                                                 !
!COM   NAME                                                         !
!COM        'LTV_KALMAN' - Subroutine                                !
!COM                                                                 !
!COM   PURPOSE                                                      !
!COM        Kalman filter for the linear time-varying (LTV) model:   !
!COM                                                                 !
!COM           #1: dX = [A*X + B*U]dt + PI*dW                        !
!COM                                                                 !
!COM           #2:  Y =  C*X + D*U + e, e~N(0,SIGMA)                 !
!COM                                                                 !
!COM        Where                                                    !
!COM                                                                 !
!COM                A = A(X,U,t,PHI)   B = B(X,U,t,PHI)              !
!COM                C = A(X,U,t,PHI)   D = D(X,U,t,PHI)              !
!COM                PI= PI(U,t,PHI)    SIGMA = SIGMA(U,t,PHI)        !
!COM                                                                 !
!COM   REFERENCE                                                    !
!COM        Eqn. (1.3) and (1.4) [CTSM 2.3 Math Guide, Dec. 2003,    !
!COM        Kristensen, N.R.].                                       !
!COM                                                                 !
!COM   INCLUDES                                                     !
!COM        '/INCLUDE/FORMAT'                                        !
!COM                                                                 !
!COM   CALLED BY                                                    !
!COM        Subroutine CNTDIFF_AIAPLL                                !
!COM        Subroutine LLDLL                                         !
!COM                                                                 !
!COM   CALLS                                                        !
!COM        Function FPI                                             !
!COM        Function FSIGMA                                          !
!COM        Function IDMN                                            !
!COM        Subroutine DEXPM                                         !
!COM        Subroutine LINEAR_MODEL                                  !
!COM        Subroutine DTRM                                          !
!COM                                                                 !
!COM   F95 INTERFACE                                                !
!COM        SUBROUTINE LTV_KALMAN(LL,KOBJ,PHI,ETA,OMEGA,POBJ,INFO)   !
!COM                                                                 !
!COM        USE SUNPERF                                              !
!COM        USE MOTYPES, ONLY: DOSE, PATIENT, KALOBJ                 !
!COM        USE MOIOS                                                !
!COM        IMPLICIT NONE                                            !
!COM                                                                 !
!COM        INTEGER,INTENT(OUT) :: INFO                              !
!COM        REAL(8),INTENT(IN) :: PHI(:),ETA(:),OMEGA(:,:)           !
!COM        TYPE(PATIENT),INTENT(IN) :: POBJ                         !
!COM        TYPE(KALOBJ),INTENT(INOUT) :: KOBJ                       !
!COM        REAL(8),INTENT(OUT) :: LL                                !
!COM                                                                 !
```

Continued on next page...

Table A.44: Interface of subroutine `LTV KALMAN` – Continued.

```
!COM   ARGUMENTS                                                        !
!COM        ETA (input)                                                 !
!COM             Random-effects vector - vector dimension (NETA).       !
!COM                                                                    !
!COM        INFO (output)                                               !
!COM             = 0:  successful exit.                                 !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.   !
!COM                                                                    !
!COM        KOBJ (input/output)                                         !
!COM             Kalman object (derived type 'KALOBJ') containing       !
!COM             results from Kalman filtering.                         !
!COM                                                                    !
!COM        OMEGA (input)                                               !
!COM             Variance-covariance matrix of the random-effects ETA   !
!COM             in the second stage model defined by:                  !
!COM                                                                    !
!COM                #3: PHI = h(THETA,Z)*EXP(ETA), ETA~N(0,OMEGA)       !
!COM                                                                    !
!COM             Matrix of dimension (NETA,NETA).                       !
!COM                                                                    !
!COM        PHI (input)                                                 !
!COM             Individual parameters vector - vector dimension        !
!COM             (NPHI). Unchanged on exit.                             !
!COM                                                                    !
!COM        POBJ (input)                                                !
!COM             Patient object (derived type 'PATIENT') containing     !
!COM             individual patient data.                               !
!COM                                                                    !
!COM        LL (output)                                                 !
!COM             Minus log-likelihood value. Scalar.                    !
!COM                                                                    !
!COM-----------------------------------------------------------------------!
```

Table A.45: Interface of subroutine `READDAT`

```
!-----------------------------------------------SUBROUTINE READDAT-----!
!
!COM-------------------------------------------------------------------!
!COM   READ INPUT DATA FROM FILE 'DATFIL1', 'DATFIL2' AND 'DATFIL3'   !
!COM-------------------------------------------------------------------!
!COM                                                                  !
!COM   NAME                                                          !
!COM        'READDAT' - Subroutine                                    !
!COM                                                                  !
!COM   PURPOSE                                                       !
!COM        Reads observation data ('datfil1'), dose administration   !
!COM        data ('datfil2') and covariates data ('datfil3'). All     !
!COM        data are allocated in patient object 'POBJ' (derived type !
!COM        'PATIENT'), which is returned to the calling procedure.   !
!COM                                                                  !
!COM   REFERENCE                                                     !
!COM        -                                                         !
!COM                                                                  !
!COM   INCLUDES                                                      !
!COM        -                                                         !
!COM                                                                  !
!COM   CALLED BY                                                     !
!COM        Program MAIN                                              !
!COM                                                                  !
!COM   CALLS                                                         !
!COM        Subroutine STRINGS                                        !
!COM                                                                  !
!COM   F95 INTERFACE                                                 !
!COM        SUBROUTINE READDAT(POBJ,DATFIL1,DATFIL2,DATFIL3,INFO)     !
!COM                                                                  !
!COM        USE SUNPERF                                               !
!COM        USE MOIOS                                                 !
!COM        USE MOPARAMS                                              !
!COM        IMPLICIT NONE                                             !
!COM                                                                  !
!COM        TYPE(PATIENT),INTENT(INOUT) :: POBJ(:)                    !
!COM        CHARACTER(LEN=20),INTENT(IN) :: DATFIL1,DATFIL2,DATFIL3   !
!COM        INTEGER,INTENT(OUT) :: INFO                               !
!COM                                                                  !
!COM   ARGUMENTS                                                     !
!COM        POBJ (input/output)                                       !
!COM             Patient object (derived type 'PATIENT') containing   !
!COM             individual patient data. On exit, it contains all    !
!COM             data supplied in the datafiles.                      !
!COM                                                                  !
!COM        'datfil1' (input)                                         !
!COM             Name of patient data file (observations).            !
!COM                                                                  !
!COM        'datfil2' (input)                                         !
!COM             Name of patient dose file (dose administration).     !
!COM                                                                  !
!COM        'datfil3' (input)                                         !
!COM             Name of covariate data file.                         !
!COM                                                                  !
```

Continued on next page...

Table A.45: Interface of subroutine `READDAT` – Continued.

```
!COM        INFO (output)                                                  !
!COM             = 0:  successful exit.                                    !
!COM             > 0:  unsuccessful exit, see subroutine 'ERRORSTAT'.      !
!COM                                                                       !
!COM-----------------------------------------------------------------------!
```

Table A.46: Interface of format file `FORMAT`

```
!--------------------------------------------------------------------!
!                                                                    !
!     USER SPECIFICATION OF FORMATS                                  !
!                                                                    !
!--------------------------------------------------------------------!
!
!COM-----------------------------------------------------------------!
!COM                                                                 !
!COM   FILE NAME                                                    !
!COM        'FORMAT'                                                 !
!COM                                                                 !
!COM   FILE DIRECTORY                                               !
!COM        '/INCLUDE/'                                              !
!COM                                                                 !
!COM   INCLUDED IN                                                  !
!COM        Subroutine 'READDAT'                                     !
!COM        Subroutine 'DISPLAY_RESULTS'                             !
!COM        Subroutine 'LTI_KALMAN'                                  !
!COM        Subroutine 'LTV_KALMAN'                                  !
!COM        Subroutine 'LLDLL'                                       !
!COM                                                                 !
!COM   PURPOSE                                                      !
!COM        To define input/output formats for files in PSM.         !
!COM                                                                 !
!COM   PARAMETERS                                                   !
!COM        -                                                        !
!COM                                                                 !
!COM-----------------------------------------------------------------!
!COM                                                                 !
!COM   COMMENTS                                                     !
!COM        ::  The current format settings do not take advantage of !
!COM            Fortran's 'tab'- and 'blank-'. A more refined format- !
!COM            ting setup is recommended.                           !
!COM                                                                 !
!COM            [CHRISTENSEN, A.S.,2007-02-04]                       !
!COM                                                                 !
!COM-----------------------------------------------------------------!
```

Appendix B

Error Statements

Table B.1: Information transferred by `INFO`

Value	Description
0	The Population Stochastic Modelling programme completed without errors.
10	Illegal model assignment. The model parameter must be of the type: = 0: Linear time-invariant model. = 1: Linear time varying model.
21	Allocation error when attempting to allocate the eta-parameter object!
22	Allocation error when attempting to allocate the theta-parameter object!
23	Allocation error when attempting to allocate the kalman object!
50	Allocation error occured during data acquisition from datafiles

Continued on next page...

Table B.1: Information transferred by `INFO` – Continued.

Value	Description
51	Dellocation error occured during data acquisition from datafiles
100	Allocation error occured in Kalman filtering procedure!
150	Computation of matrix exponential failed in Kalman filtering procedure!
155	Computation of LU-factorization failed!
156	Not possible to compute matrix inversion of singular matrix!
160	Computation of inverse matrix failed!
165	Computation of singular value decomposition failed!
202	THETA optimization failed to start in subroutine APL. NTHETA ≤ 0.
204	THETA optimization failed to start in subroutine APL. Step size DX too small.
205	THETA optimization failed to start in subroutine APL. Stop criterion EPS ≤ 0.0.
206	THETA optimization failed to start in subroutine APL. Maximum number of iterations allowed MAXFUN ≤ 0.0.
207	THETA optimization failed to start in subroutine APL. Given Hessian matrix is not positive definite.
208	THETA optimization failed to start in subroutine APL. The workspace is too small.
212	ETA optimization failed to start in subroutine AIAPLL. NTHETA ≤ 0.
214	ETA optimization failed to start in subroutine AIAPLL. Step size DX too small.
215	ETA optimization failed to start in subroutine AIAPLL. Stop criterion EPS ≤ 0.0.
216	ETA optimization failed to start in subroutine AIAPLL. Maximum number of iterations allowed MAXFUN ≤ 0.0.

Continued on next page...

Table B.1: Information transferred by `INFO` – Continued.

Value	Description
`217`	ETA optimization failed to start in subroutine AIAPLL. Given Hessian matrix is not positive definite.
`218`	ETA optimization failed to start in subroutine AIAPLL. The workspace is too small.
`300`	Mapping of fixed-effects THETA failed!
`DEFAULT`	An error occured during execution.

Appendix C

Flowchart Symbol List

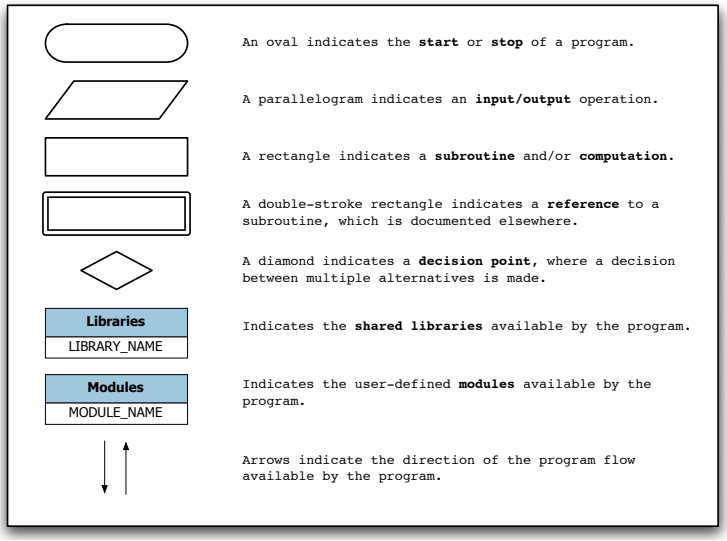

Figure C.1: Definition of the symbols used for flowcharts.

Bibliography

[1] C. Chatfield. *The Analysis of Time Series: An Introduction*. Chapman & Hall/CRC, 6th edition, 2004.

[2] M. Davidian. Nonlinear mixed effects models: An overview and update. In *Interational Biometric Conference, 2004*, 2004. URL `http://www4.stat.ncsu.edu/~davidian/nlmmtalk.pdf`.

[3] K. L. Davis, D. Charney, J. T. Coyle, and C. Nemeroff, editors. *Neuropsychopharmacology: The Fifth Generation of Progress*. American College of Neuropsychopharmacology, 2002. Chapter 38: Pharmacokinetics, Pharmacodynamics, And Drug Disposition.

[4] U.S. FDA. Guidance for Industry – Population Pharmacokinetics. URL `http://www.fda.gov/cber/gdlns/popharm.htm`, 1999.

[5] P.E. Frandsen, K. Jonasson, H. B. Nielsen, and Ole Tingleff. Unconstrained optimization. Technical report, IMM, Technical University of Denmark, 2004.

[6] J. Gabrielsson and D. Weiner. *Pharmacokinetic and Pharmacodynamic Data Analysis: Concepts and Applications*. Kristianstads Boktryckeri, 2nd edition, 1997.

[7] J.G. Hardman and L.E. Limbird. *The Pharmacological Basis of Therapeutics*. Goodman & Gilman, McGraw Hill, 10th edition, 2001.

[8] N. R. Kristensen and H. Madsen. Continuous time stochastic modelling. Technical report, IMM, Technical University of Denmark, December 2003.

[9] N. R. Kristensen, H. Madsen, and S. H. Ingwersen. Using stochastic differential equations for PK/PD model development. *Journal of Pharmacokinetics and Pharmacodynamics*, 32(1):109–141(33), February 2005.

[10] S. Mortensen and S. Klim. Stochastic PK/PD modelling. Master's thesis, Technical University of Denmark, 2006.

[11] H. B. Nielsen. Ucminf - an algorithm for unconstrained nonlinear optimization. Technical Report IMM-REP-2000-19, IMM, Technical University of Denmark, 2000.

[12] R. V. Overgaard. Non-linear mixed-effects models with stochastics differential equations: Implementation of an estimation algorithm. *Journal of Pharmacoki-*

netics and Pharmacodynamics, 32(1):85–107, February 2005.

[13] L. Sheiner and J. Wakefield. Population modelling in drug development. *Statistical Methods in Medical Research*, 8:183–93, 1999.

[14] R. B. Sidje. Expokit. A software package for computing matrix exponentials. *ACM Trans. Math. Softw.*, 24(1):130–156, 1998.

[15] Sun Studio. *Sun Studio: Sun Performance Library*. Sun®, `http://developers.sun.com/sunstudio/perflib_index.html`, 2007.

[16] C. W. Tornøe. *Population pharmacokinetic/pharmacodynamic modelling of the hypothalamic-pituitary-gonadal axis*. PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2005. URL `http://www2.imm.dtu.dk/pubdb/p.php?3926`.

[17] C W. Tornøe, R. V. Overgaard, H. Agersø, H. A. Nielsen, H. Madsen, and E. N. Jonsson. Stochastic differential equations in NONMEM®: Implementation, application, and comparison with ordinary differential equations. *Pharmaceutical Research*, 22(8):1247–1258, March 2005.

[18] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, Department of Computer Science, University of North Carolina, July 2006.

[19] The Danish Medicines Agency's working group on clinical pharmacy. Better use of medicines - perspectives in clinical pharmacy. URL `http://www.dkma.dk/1024/visUKLSArtikel.asp?artikelID=6378`, June 2005.