PFIM 4.0

PFIM Group
IAME UMR1137, INSERM and Université Paris Diderot, Paris, France

August 2014

www.pfim.biostat.fr

User guide

Written by Cyrielle Dumont and Thu Thuy Nguyen

****

Contact: pfim@inserm.fr

Members of the PFIM Group

Disclaimer

We inform users that the PFIM 4.0 is a tool developed by the Laboratory "Biostatistics-Investigation-Pharmacometrics" - UMR 1137 INSERM and University Paris Diderot, under R and GCC.

PFIM 4.0 is a library of functions. The functions are published after a scientific validation.

However, it may be that only extracts are published.

By using this library of functions, the user accepts all the conditions of use set forth hereinafter.

Licence

CONTENTS

1 Introduction

Model based optimal design approaches are increasingly performed in population pharmacokinetic/pharmacodynamics (PKPD) [1], which consist in determining a balance between the number of subjects and the number of samples per subject, as well as the allocation of times and doses, according to experimental conditions. A good choice of design is crucial for an efficient estimation of model parameters, especially when the studies are conducted in patients where only a few samples can be taken per subject. These approaches rely on the Fisher information matrix (FIM) for nonlinear mixed effect models (NLMEM), available in several software tools [2] and are a good alternative to clinical trial simulation. They require *a priori* knowledge of the model and its parameters, which can usually be obtained from previous experiments.

PFIM (www.pfim.biostat.fr), developed in our group, is the first tool for design evaluation and optimisation that has been developed in R. It is available since 2001 [3] and was extended in version 3 to multi-response models, inter-occasion variability, discrete covariates with prediction of power of Wald test [4,5]. The current version 4, released in Spring 2014, added several new features.

In this new version, for population designs, optimisation can be performed with fixed parameters or fixed sampling times. The Fisher information matrix obtained after evaluation or optimisation can be saved in a file. Previous information already obtained can be assumed and loaded through a predicted or an observed Fisher information matrix, which is important in the perspective of performing adaptive designs [6]. Additional features for Bayesian designs are now available. The Bayesian Fisher information matrix has been implemented. Design for maximum *a posteriori* estimation of individual parameters can be evaluated or optimised and the predicted shrinkage is also reported [7]. A new way has been added to specify user-defined models through an R function. It is now possible to visualise the graphs of the model and the sensitivity functions without performing evaluation or optimisation.

This documentation describes the methodology implemented in PFIM 4.0 in Section 2. Section 3 explains how to install and use PFIM. Section 4 describes how to specify models, either by using the PKPD library or the user-defined model option. Lastly sections 5 and 6 present the input and output of PFIM.

2 Methodology

2.1 Design

The elementary design ξ_i of individual i ($i = 1, \ldots, N$) is defined by the number n_i of samples and their allocation in time $(t_{i1}, \ldots, t_{ini})$.
For N individuals, the population design is composed of the N elementary designs such as $\Xi = \{\xi_1, \ldots, \xi_N\}$. Usually, population designs are composed of a limited number Q of groups of individuals with identical design ξ_q within each group, performed in a number N_q of individuals. The population design can thus be written as $\Xi = \{[\xi_1, N_1]; \ldots; [\xi_Q, N_Q]\}$.
Individual and Bayesian designs include only one elementary design.

2.2 Nonlinear mixed effects models

A nonlinear mixed effects model, or a population model, is defined as follows. The vector of observations Y_i for the individual i ($i = 1, \ldots, N$) is defined as

$$Y_i = f(\theta_i, \xi_i) + \varepsilon_i,$$

where the function f defines the nonlinear structural model, θ_i is the vector of the p-individual parameters for individual i, ξ_i is the elementary design of individual i and ε_i is the vector of residual error.
The vector of individual parameters θ_i depends on μ, the p-vector of the fixed effects parameters and on b_i, the p-vector of the random effects for individual i. The relation between θ_i and (μ, b_i) can be additive for a normal distribution of parameters, that is

$$\theta_i = \mu + b_i,$$

or exponential for a lognormal distribution of parameters so that

$$\theta_i = \mu \times \exp(b_i).$$

It is assumed that $b_i \sim N(0, \Omega)$ with Ω defined as a $p \times p$ diagonal variance-covariance matrix, for which, each diagonal element ω_j, $j = 1, \ldots, p$, represents the inter-individual variability of the j^{th} component of the vector b_i.
It is also supposed that $\varepsilon_i \sim N(0, \Sigma_i)$, where Σ_i is a $n_i \times n_i$-diagonal matrix such that

$$\Sigma_i(\mu, b_i, \sigma_{inter}, \sigma_{slope}, \xi_i) = diag(\sigma_{inter} + \sigma_{slope} \times f(\theta_i, \xi_i))^2.$$

The terms σ_{inter} and σ_{slope} are the additive and proportional parts of the error model, respectively. Conditionnally on the value of b_i, it is assumed that the ε_i errors are independently distributed.

In the case of K multiple responses, the vector of observations Y_i can then be composed of K vectors for the different responses:

$$Y_i = [y_{i1}^T, y_{i2}^T, \ldots, y_{iK}^T]^T,$$

where y_{ik}, $k = 1, \dots, K$, is the vector of n_{ik} observations for the k^{th} response. Each of these responses is associated with a known function f_k, which can be grouped in a vector of multiple response model F, such as

$$F(\theta_i, \xi_i) = [f_1(\theta_i, \xi_{i1})^T, f_2(\theta_i, \xi_{i2})^T, \dots, f_K(\theta_i, \xi_{iK})^T]^T,$$

where ξ_i is composed of K sub-designs such that $\xi_i = (\xi_{i1}, \xi_{i2}, \dots, \xi_{iK})$. The sub-design ξ_{ik} is then defined by $(t_{ik1}, t_{ik2}, \dots, t_{ikn_{ik}})$, with n_{ik} sampling times for the observations of the k^{th} response, so that $n_i = \sum_{k=1}^{K} n_{ik}$.
Each response can have its error model and ε_i is then the vector composed of the K vectors of residual errors ε_{ik}, $k = 1, \dots, K$, associated with the K responses.

Inter-occasion variability specification

The expression of the nonlinear mixed effects model has been extended for model including additional random effects for inter-occasion variability (or within subject variability) [5].
The individual parameters of an individual i at occasion h are thus expressed by the following relation, which can be additive as

$$\theta_{ih} = \mu + b_i + \kappa_{ih}$$

or exponential as

$$\theta_{ih} = \mu \times \exp(b_i + \kappa_{ih})$$

where μ is, as previously, the p-vector of fixed effects, b_i the vector of random effects associated to the individual i and κ_{ih} the vector of random effects associated to the individual i for the occasion h ($h = 1, \dots, H$ with H the number of occasions). b_i and κ_{ih} are supposed independent. It is assumed that $b_i \sim N(0, \Omega)$ and $\kappa_{ih} \sim N(0, \Gamma)$ with Ω and Γ defined as diagonal matrices of size $p \times p$. Each element ω_j of Ω and γ_j of Γ represent the inter-individual variability of the j^{th} component of b_i and the inter-occasion variability of the j^{th} component of κ_{ih}, respectively.

This new development was performed for any number of occasions H. It is implemented in PFIM for the case where same elementary designs are used at each occasion.

Discrete covariate specification

The present expression of nonlinear mixed effects models accommodates models with parameters quantifying the influence of discrete covariates. Two or more categories can be included. In PFIM 3.2, it can be assumed either that covariates are additive on parameters if the random effect model is additive, or that covariates are additive on log parameters if the random effect model is exponential.
For instance, the individual parameter θ_i is described as the function of a discrete covariate C_i, which takes D values defining D categories, with additive effect model:

$$\theta_i = \mu + \sum_{d=2}^{D} \beta_d \, 1_{C_{i=d}} + b_i$$

where here $d = 1$ is defined as the reference group and $\beta_1 = 0$.
For each covariate, the user has to specify β, i.e. the vector of covariate effect coefficients and the proportions of subjects associated to the *D* categories.
It can be specified if covariates change or not through the different occasions. In the latter case, additional objects are needed: the vector of sequences of values of each covariate at each occasion and the vector of proportions of the elementary designs corresponding to each sequence of covariate values (see Section 5 for input specification).
The number of covariates, the number of parameters associated to each covariate as well as the number of categories for each covariate, are not limited. In PFIM, the distributions of the covariates are supposed independent.

2.3 Fisher information matrix

2.3.1 Population Fisher information matrix

2.3.1.1 Expression

The population Fisher information matrix $M_F(\Psi,\xi)$ for multiple response models, for an individual with an elementary design ξ, with the vector of population parameters Ψ, is given as:

$$M_F(\Psi,\xi) \cong \frac{1}{2}\begin{pmatrix} A(E,V) & C(E,V) \\ C^T(E,V) & B(E,V) \end{pmatrix}$$

with *E* and *V* the approximated marginal expectation and the variance of the observations of the individual. The vector of population parameter Ψ is defined by $\Psi^T = (\mu^T, \lambda^T)$ with μ the p-vector of the fixed effects and λ the vector of the variance terms. M_F is given as a block matrix (more details are given in [8–10]) with:

$$(A(E,V))_{ml} = 2\frac{\partial E^T}{\partial \mu_m}V^{-1}\frac{\partial E}{\partial \mu_l} + tr(\frac{\partial V}{\partial \mu_l}V^{-1}\frac{\partial V}{\partial \mu_m}V^{-1}) \text{ with } m \text{ and } l = 1,\ldots,p$$

$$(B(E,V))_{ml} = tr(\frac{\partial V}{\partial \lambda_m}V^{-1}\frac{\partial V}{\partial \lambda_l}V^{-1}) \quad \text{with } m \text{ and } l = 1,\ldots,\dim(\lambda)$$

$$(C(E,V))_{ml} = tr(\frac{\partial V}{\partial \lambda_l}V^{-1}\frac{\partial V}{\partial \mu_m}V^{-1}) \text{ with } l = 1,\ldots,\dim(\lambda) \text{ and } m = 1,\ldots,p$$

If the dependence of V in μ is neglected so that $\frac{\partial V}{\partial \mu} = 0$, the population Fisher information matrix is a block diagonal matrix that is to say the block C of the matrix is supposed to be 0. Also, the block A is simplified and expressed as:

$$(A(E,V))_{ml} = 2\frac{\partial E^T}{\partial \mu_m}V^{-1}\frac{\partial E}{\partial \mu_l} \quad \text{with } m \text{ and } l = 1,\ldots,p$$

Based on publications showing the better performance of the block diagonal expression compared to the full one with linearisation [2], the default option in PFIM is the block diagonal information matrix. However, since PFIM 3.2, the user can choose to compute either a full or a block diagonal matrix for models without covariate and inter-occasion variability. The size of the block C and the block B of the expression of the Fisher information matrix are thus modified to incorporate the within subject variabilities Γ.

Prediction of standard errors

According to the inequality of Cramer-Rao, the inverse of M_F is the lower bound of the variance-covariance matrix of any unbiased estimate of the parameters. From the square roots of the diagonal elements of the inverse of M_F, the predicted standard errors (SE) for estimated parameters can be calculated.

2.3.1.2 Computation of power and number of subjects needed

Comparison test

Computation of the expected power. The Wald test can be used to assess the difference of a covariate effect β. In PFIM, the Wald test is performed on the β of each category for each covariate; a global Wald test on the vector β (all effect coefficients) is not implemented.
For one covariate and an effect of one category β (D=2), the null hypothesis is H_0: {β=0} while the alternative hypothesis is H_1: {β≠0}. The statistic of the Wald test is defined as, $S_W = \frac{\hat{\beta}}{SE(\hat{\beta})}$ with $\hat{\beta}$ the covariate effect estimates and $SE(\hat{\beta})$ its associated standard error. Under H_1, when $\beta=\beta_1$, we then compute the power of the Wald test defined as:

$$P_{diff} = 1-\phi\left(z_{1-\alpha/2} - \frac{\beta_1}{SE(\beta_1)}\right) + \phi\left(-z_{1-\alpha/2} - \frac{\beta_1}{SE(\beta_1)}\right) \qquad (1)$$

where Φ is the cumulative distribution function of the standard normal distribution and $z_{\alpha/2}$ is such that $\phi(z_{\alpha/2}) = 1-\alpha/2$.
Using the covariate effect β_1 fixed by the user, the corresponding standard error $SE(\beta_1)$ is predicted since PFIM 3.2 for a given design and the values of population parameters.

Computation of the number of subjects needed. The number of subjects needed to achieve a power *P* to detect a covariate effect using the Wald test is also computed. First, from the equation (1), we compute the SE needed on β to obtain a power of *P*, called NSE(P), using the following relation:

$$NSE(P) = \frac{\beta_1}{z_{\alpha/2} - \phi^{-1}(1-P)} \qquad (2)$$

Last, we compute the number of subjects needed to be included to obtained a power of *P*, called NNI(P) using

$$NNI(P) = N \times \frac{SE(\beta_1)}{NSE(P)} \tag{3}$$

where *N* is the initial number of subjects in the given design and $SE(\beta_1)$ the corresponding predicted SE of *β* for the given design.

Equivalence test

Computation of the expected power. The Wald test can be used to assess the equivalence of a covariate effect *β*.
In PFIM, the Wald test is performed on the *β* of each category for each covariate, a global Wald test on the vector *β* (all effect coefficients) is not implemented.
For one covariate and an effect of one category *β* (D=2), the null hypothesis is H_0: {$\beta \leq -\Delta_L$ or $\beta \geq +\Delta_L$} while the alternative hypothesis is H_1: {$-\Delta_L \leq \beta \leq +\Delta_L$}. H_0 is composed of two unilateral hypothesis $H_{0,-\Delta_L}$:{$\beta \leq -\Delta_L$} and $H_{0,+\Delta_L}$:{$\beta \geq +\Delta_L$}. Equivalence between two covariate effects can be concluded if and only if the two hypotheses $H_{0,-\Delta_L}$ and $H_{0,+\Delta_L}$ are rejected.
The two statistics of the unilateral Wald test under the null hypothesis are defined as, $S_{W_{-\Delta_L}} = \frac{\hat{\beta} + \Delta_L}{SE(\hat{\beta})}$ and $S_{W_{+\Delta_L}} = \frac{\hat{\beta} - \Delta_L}{SE(\hat{\beta})}$ with $\hat{\beta}$ the covariate effect estimates and, its associated standard error. Under H_1, when $\beta=\beta_1$ with $\beta_1 \in [-\Delta_L, \Delta_L]$, we then compute the power of the equivalence Wald test defined as:

$$P_{equi} = 1 - \phi\left(z_{1-\alpha} - \frac{\beta_1 + \Delta_L}{SE(\beta_1)}\right) \quad if\ \beta_1 \in [-\Delta_L, 0] \tag{4}$$

$$P_{equi} = \phi\left(-z_{1-\alpha} - \frac{\beta_1 - \Delta_L}{SE(\beta_1)}\right) \quad if\ \beta_1 \in [0, +\Delta_L] \tag{5}$$

where Φ is the cumulative distribution function of the standard normal distribution and z_α is such that $\phi(z_\alpha) = 1 - \alpha$.

In equivalence test β_1 is usually chosen to be zero.

Using the covariate effect β_1 fixed by the user, the corresponding standard error $SE(\beta_1)$ is predicted since PFIM 3.2 for a given design and the values of population parameters.

Computation of the number of subjects needed. The number of subjects needed to achieve a power *P* to show equivalence between two covariate effects using the Wald test is also computed. First, from equations (4) and (5), we compute the SE needed on *β* to obtain a power of *P*, called NSE(P), using the following relation:

$$NSE(P) = \frac{(-\beta_1 - \Delta_L)}{-z_\alpha + \phi^{-1}(1-P)} \quad if\ \beta_1 \in [-\Delta_L, 0] \tag{6}$$

$$NSE(P)=\frac{(-\beta_1+\Delta_L)}{z_\alpha+\phi^{-1}(P)} \quad if\ \beta_1\in[0,+\Delta_L] \tag{7}$$

where Φ is the cumulative distribution function of the standard normal distribution and z_α is such that $\phi(z_\alpha)=1-\alpha$.

Last, we compute the number of subjects needed to be included to obtained a power of *P*, called NNI(P) using the equation (3) like for comparison test.

2.3.1.3 Previous information

New feature: An option to load previous information through a predicted or an observed Fisher information matrix is now available in PFIM 4.0. Evaluation and optimisation are then performed combining the previous information matrix with the current Fisher information matrix, following the principle of adaptive designs [6].

Taking into account previous information, the new computation of the Fisher information matrix is then:

$$\mathrm{M}_{\mathrm{F\,prev}}+\sum_{i=1}^{N}\mathrm{M}_{\mathrm{F}}(\Psi,\xi_i)$$

where $\mathrm{M}_{\mathrm{F\,prev}}$ denotes the previous Fisher information matrix.
Note that the previous Fisher information matrix should have the same dimension as the current Fisher information matrix.

It is now possible to save the Fisher information matrix corresponding to an evaluated or optimised design.

2.3.2 Bayesian Fisher information matrix

New feature: The new version 4.0 of PFIM enables design evaluation and optimization for maximum *a posteriori* estimation of individual parameters based on the Bayesian Fisher information matrix [7].

We are interested in the precision estimation of individual parameters for a subject *i*, associated to the vector of observation *y* (index *i* being omitted). These individual parameters can be estimated by maximum *a posteriori* (MAP). As μ is known, estimating θ is similar to estimating η. More precisely, the MAP estimate of η is given by

$$\hat{\eta} = \mathrm{argmax}\ (p(\eta \mid y)) = \mathrm{argmax}\left(\frac{p(y \mid \eta)\ p(\eta)}{p(y)}\right) = \mathrm{argmax}\ (\log\ (p(y \mid \eta)) + \log\ (p(\eta)))$$

where *p* is the probability density. The Bayesian Fisher information matrix, taking into account the *a priori* distribution of the random effects, is expressed as

$$M_{BF}(\xi) = -E_\eta\left(\frac{\partial^2 \log(p(\eta \mid y))}{\partial\eta\partial\eta^T}\right) = -E_\eta\left(E_{y\mid\eta}\left(\frac{\partial^2 \log(p(y \mid \eta))}{\partial\eta\partial\eta^T}\right)\right) - E_\eta\left(\frac{\partial^2 \log(p(\eta))}{\partial\eta\partial\eta^T}\right)$$

$$= E_\eta(M_{IF}(g(\mu,\eta),\xi)) - \Omega^{-1}$$

where $M_{IF}(\theta, \xi) = -E_y\left(\frac{\partial^2 \log(p(y \mid \theta))}{\partial\theta\partial\theta^T}\right)$, expression of the individual Fisher information matrix in classical nonlinear regression models. The expectation $E_\eta\left(M_{IF}(g(\mu, \eta), \xi)\right)$ can be obtained by first order approximation of the model around the expectation of random effects (*i.e.,* 0).
The shrinkage (Sh) is quantified from the ratio of the estimation variance predicted by M_{BF}^{-1} and the *a priori* variance, and can be calculated as the diagonal elements of the matrix $I - W(\xi) = M_{BF}(\xi)^{-1}\ \Omega^{-1}$ (see [7] for more details).

When a parameter has an *a priori* variance equal to 0, it will be considered as fixed to the mean value and no predicted shrinkage will be computed.

2.4 Design evaluation

Population, individual and Bayesian design evaluation is based on the computation of the population, individual and Bayesian Fisher information matrix, respectively. During this process, the expected standard errors on the population or individual parameters with the design are evaluated. The user can choose to fix one or several parameters in the model that will not be computed in the Fisher information matrix.
Eigenvalues and conditional number are given by default. When considering design for Bayesian estimation of individual parameters, the shrinkages are also reported.
The computed Fisher information matrix can be saved in a file if requested.

2.5 Design optimisation

Algorithms are required to optimise exact or statistical designs. In the case of an exact optimisation, the group structure of the design is fixed: the number of elementary designs, the number of samples per elementary design and the number of subjects per elementary design are given and the design variables to optimise are only the sampling times. In the case of statistical optimisation, the sampling times (number and allocation) and the proportions of subjects in each elementary design are optimised.
PFIM optimises population design using the D-optimal criterion, *i.e.* maximising the determinant of the population Fisher information matrix, or, similarly, minimising its inverse.
The Fedorov-Wynn algorithm has been implemented since PFIM 3.0 in addition to the Simplex algorithm. Compared to the Simplex algorithm, the Fedorov-Wynn algorithm better affords high design variables optimisation. Moreover, it considers only pre-specified sampling times, avoiding, clinically unfeasible sampling times. The drawback is the huge number of elementary designs to be created (with corresponding huge number of Fisher information matrices to compute) when the set of allowed sampling times is very large.

2.5.1 Simplex algorithm

The Simplex algorithm optimises statistical or exact designs in constrained intervals, given a total number of samples.
An initial population design needs to be supplied to start the optimisation. The maximum number of elementary designs and the number of sampling times per elementary design are fixed, the sampling times and the proportions of subjects in each elementary design are then optimised. From this initial design, initial vertices for the simplex algorithm are derived, reducing successively each component by 20% (a default value which can be changed) from the original component.
PFIM uses the Splus function "fun.amoeba" from Daniel Heitjan (revised 12/94), which is a translation from the Numerical Recipes for Nelder and Mead Simplex function [11].
Note that it is now possible to take into account previous information through a predicted or an observed Fisher information matrix to optimise designs with this algorithm.

2.5.2 Fedorov-Wynn algorithm

The Fedorov-Wynn algorithm is specifically dedicated to design optimisation problems and has the property to converge towards the D-optimal design [12-14]. It optimises statistical designs for a given total number of samples. The sampling times are chosen among a given finite set of times. Minimum and maximum numbers of samples per subject are specified.
To start the algorithm, an initial population design is then required.
The Fedorov-Wynn algorithm is programmed in a C code and is linked to PFIM through a dynamic library, called libFED.dll and libFED64.dll for R 32-bit and 64-bit respectively. Moreover, PFIM uses the function combn in the R package "combinat".

New feature: The best one group protocol, which maximises the determinant of the elementary Fisher information matrix of all elementary protocols chosen among the predefined set of samples, is given by default when running Fedorov-Wynn algorithm (before calling the dynamic library). This is the optimal protocol for individual design and Bayesian design.
Moreover, in PFIM 4.0, optimisation with Fedorov-Wynn algorithm can be performed assuming that some sampling times are fixed.

3 Use

3.1 Pre-requirement

The software R is required. To use PFIM 4.0, additional packages are needed in the R library directory:

- for differential equation system to describe the model: "deSolve" and "nlme" packages
- for the Fedorov-Wynn algorithm: "combinat" package

An additional package "numDeriv" is needed for the computation of the full Fisher information matrix and for numerical derivatives of models written as standard R functions.

The easiest way to install packages is directly from the web. To install the packages deSolve, nlme, combinat and numDeriv, start R and choose the Packages item from the menu. Choose Install package(s) from CRAN to install from the web (you will see a list of all available packages pop up -- choose deSolve, nlme, combinat and numDeriv).
To install PFIM 4.0, the user has to download the function named PFIM 4.0 available on the webpage www.pfim.biostat.fr.

3.2 Components

PFIM 4.0 includes two main folders called:

- PFIM 4.0
- Examples

The folder PFIM 4.0 is composed of 3 principal files and one folder:

- The 3 principal files are:
  - The main function (program) file (**PFIM.r**)
  - The input file (**stdin.r**)
  - The model file (**model.r**).

- The folder is called **Program** and contains the files of functions:
  - ***Pfim4.0op1.r***: To compute the block diagonal Fisher Information matrix (option 1) to evaluate a population, individual or Bayesian design using an analytical form to describe the model.
  - ***PfimOPT4.0op1.r***: To compute the block diagonal Fisher Information matrix (option 1) to optimise a population, individual or Bayesian design using an analytical form to describe the model
  - ***EQPfim4.0op1.r***: To compute the block diagonal Fisher Information matrix (option 1) to evaluate a population, individual or Bayesian design using a differential equation system to describe the model
  - ***EQPfimOPT4.0op1.r***: To compute the block diagonal Fisher Information matrix (option 1) to optimise a population, individual or Bayesian design using a differential equation system to describe the model
  - ***Pfim4.0op2.r***: To compute the full Fisher Information matrix (option 2) to evaluate a population design using an analytical form to describe the model.
  - ***PfimOPT4.0op2.r***: To compute the full Fisher Information matrix (option 2) to optimise a population design using an analytical form to describe the model

- ***EQPfim4.0op2.r***: To compute the full Fisher Information matrix (option 2) to evaluate a population design using a differential equation system to describe the model
- ***EQPfimOPT4.0op2.r***: To compute the full Fisher Information matrix (option 2) to optimise a population design using a differential equation system to describe the model
- ***algosimplex4.0.r***: To use the Simplex algorithm
- ***initfedoR.c*** and ***classfed.h***: To compile the dll
- ***libFED.dll and libFED64.dll:*** The dynamic libraries of the Fedorov-Wynn algorithm for R 32-bit and 64-bit respectively
- ***algofedorov4.0.r:*** To use the dynamic library libFED.dll or libFED64.dll
- ***LibrayPK.r***: To use the library of pharmacokinetic models
- ***LibrayPD_PDdesign.r***: To use the library of immediate response pharmacodynamic models alone
- ***LibrayPD_PKPDdesign.r***: To use the library of pharmacodynamic models linked to pharmacokinetic models both written using analytical form
- ***CreateModel_PKPDdesign.r:*** To use the libraries of pharmacokinetic and pharmacodynamic models when they are writing either with different forms or both with differential equation systems.

The files in the folder Program should not be changed.

The folder called Examples contains the examples files. The documentation which gives their description is included in PFIM 4.0 with this user guide.

To install PFIM 4.0, create a directory (for example directory "U:\\My Documents\\PFIM 4.0") and download PFIM 4.0.

3.3 Working directory

- Create a working directory, for example:

  "U:\\My Documents\\PFIM 4.0_examples\\Example1"

- Copy the files PFIM.r, stdin.r and model.r in this directory

- In the file "PFIM.r", specify your working directory:

  directory<-"U:\\My Documents\\PFIM 4.0_examples\\Example1"

- Then, specify your program directory i.e. where is the folder called **Program**

  directory.program<-"U:\\My Documents\\PFIM 4.0\\Program"

- Save the file PFIM.r

3.4 Run

Once the input file and the model file are filled in, the user can run PFIM. Load the main function PFIM() implemented in the file PFIM.r. To do that, choose the File item from the menu. Select "Source R code"; click on the right directories up to the file PFIM.r. The user can also load the file by typing the command in the Command Window:

```
source("U:\\My Documents\\PFIM 4.0_examples\\Example1\\PFIM.r")
```

Call the R function in the Commands window: **PFIM()**

4 Models

Models in PFIM can be specified either through their analytical form or as a solution of system of differential equations. PFIM provides libraries of models (see Section 4.1), and users may also define their own model analytically or using a system of differential equations (see Section 4.2).

The PFIM library implements R expressions or differential equation systems for PKPD models. The PK model library includes one, two and three compartment models with linear elimination and with Michaelis-Menten elimination. The PD model library supports immediate response models (alone or linked to a pharmacokinetic model) and the turnover response models (linked to pharmacokinetic model). These libraries have been derived from the PKPD library developed by Bertrand and Mentré for the MONOLIX software, and all analytical expressions are in that document [15]. A documentation of PKPD models for PFIM is available when downloading PFIM. Presently, there is no model with lag time in the library. To use the library of models, the user has to specify the path of the corresponding library file in the model file named by default *model.r*.

New feature: In the previous versions of PFIM, a user-defined model given in analytical form needed to be specified through an R expression. An alternative way to write the model is now available, through an R function with a specific format (see section 4.2.3).

4.1 Library of models

4.1.1 Library of pharmacokinetic models

Two types of PK models can be used in PFIM, PK models with a first order linear elimination or PK models with a Michaelis-Menten elimination. The PK models with a linear elimination are written using an analytical form through an R expression whereas the PK models with a Michaelis-Menten elimination are written using a differential equation system. These PK models are written in the file *LibraryPK.r* available in the Program folder. The user has to specify the path of this file in the model file to use this library of models:

source(file.path(directory.program,"LibraryPK.r"))

The following sections show the list of models for each type of PK model in separate tables. These tables display all the information in order to use the model function chosen. The model is described by:

- a **name**
- the type of **input**
- the type of **elimination**
- the **number of compartments**
- the parameters used (**parameterisation**)
- the type of **administration** (**sd** : single dose, **md:** multiple dose, **ss:** steady state) depending on administration type, additional variables may be required. They are specified in the arguments (**N:** number of doses, **tau:** interval between two doses, **TInf:** duration of the infusion, **doseMM:** dose)

For models with infusion, the user must specify the duration of infusion (**TInf**) as an argument. The rate of infusion is computed automatically in the function model through the expression: dose/TInf. For PK models with linear elimination, the variable **dose** has to be specified in the input file.

When a model with multiple dose administration is used, for example the first order oral absorption with one compartment model with option md (**oral1_1cpt_kaVCl_md**) from the library, the function of the model uses three parameters (**ka**, **Cl** and **V**) and two needed variables (**N**, **tau**): the number of doses (**N**) and the interval between two doses (**tau**) (see Example 1, section 4.2).

Pharmacokinetic models with a linear elimination

The library of PK models with linear elimination is composed of one, two and three compartment models for the three types of input (bolus, infusion and first order oral absorption) and the three types of administration (single dose, multiple dose, steady state).

The list of these PK models is given in Table 1.

Table 1. Pharmacokinetic models with first order linear elimination included in the library of models

Name	Input	Cpt	Elimination	Parameterisation	Administration	Arguments
bolus_1cpt_Vk	IV-bolus	1	1st order	V, k	sd	-
					md	N, tau
					ss	tau
bolus_1cpt_VCl	IV-bolus	1	1st order	V, Cl	sd	-
					md	N, tau
					ss	tau
infusion_1cpt_Vk	IV-infusion	1	1st order	V, k	sd	TInf
					md	TInf, N, tau
					ss	TInf, tau
infusion_1cpt_VCl	IV-infusion	1	1st order	V, Cl	sd	TInf
					md	TInf, N, tau
					ss	TInf, tau
oral1_1cpt_kaVk	1st order	1	1st order	ka, V, k	sd	-
					md	N, tau
					ss	tau
oral1_1cpt_kaVCl	1st order	1	1st order	ka, V, Cl	sd	-
					md	N, tau
					ss	tau
bolus_2cpt_Vkk12k21	IV-bolus	2	1st order	V, k, k12, k21	sd	-
					md	N, tau
					ss	tau
bolus_2cpt_ClV1QV2	IV-bolus	2	1st order	Cl, V1, Q, V2	sd	-
					md	N, tau
					ss	tau
infusion_2cpt_Vkk12k21	IV-infusion	2	1st order	V, k, k12, k21	sd	TInf
					md	TInf, N, tau
					ss	TInf, tau

infusion_2cpt_ClV1QV2	IV-infusion	2	1st order	Cl, V1, Q, V2	sd md ss	TInf TInf, N, tau TInf, tau
oral1_2cpt_kaVkk12k21	1st order	2	1st order	ka, V, k, k12, k21	sd md ss	- N, tau tau
oral1_2cpt_kaClV1QV2	1st order	2	1st order	ka, Cl, V1, Q, V2	sd md ss	- N, tau tau
bolus_3cpt_Vkk12k21k13k31	IV-bolus	3	1st order	V, k, k12, k21, k13, k31	sd md ss	- N, tau tau
bolus_3cpt_ClV1Q1V2Q2V3	IV-bolus	3	1st order	Cl, V1, Q1, V2, Q2, V3	sd md ss	- N, tau tau
infusion_3cpt_Vkk12k21k13k31	IV-infusion	3	1st order	V, k, k12, k21, k13, k31	sd md ss	TInf TInf, N, tau TInf, tau
infusion_3cpt_ClV1Q1V2Q2V3	IV-infusion	3	1st order	Cl, V1, Q1, V2, Q2, V3	sd md ss	TInf TInf, N, tau TInf, tau
oral1_3cpt_kaVkk12k21k13k31	1st order	3	1st order	ka, V, k, k12, k21, k13, k31	sd md ss	- N, tau tau
oral1_3cpt_kaClV1Q1V2Q2V3	1st order	3	1st order	ka, Cl, V1, Q1, V2, Q2, V3	sd md ss	- N, tau tau

Pharmacokinetic models with a Michaelis-Menten elimination

One, two and three compartment models are implemented for the three types of input. For bolus input, only single dose models are implemented. For infusion and first order absorption input, single dose and multiple dose are implemented. There is no steady-state form for PK models with Michaelis-Menten elimination (in this case the user can use a multiple dose model with enough doses to reach SS). The list of these PK models is given in Table 2.

For models with a bolus input, the dose has to be specified in the input file (*stdin.r* by default) as the initial condition of the differential equation system (see Example 7.1, section 4.2). For models with infusion or first order absorption input, dose has to be specified as an argument of the model function and NOT IN THE INITIAL CONDITION OF THE MODEL IN THE INPUT FILE (see Example 7.2, section 4.2).

As the dose is an argument, it is not possible to specify different doses per group for models with infusion or first order absorption input. All groups of the design considered have the same dose. Otherwise, the user should use the user defined model option.

Table 2. Pharmacokinetic models with Michaelis-Menten elimination included in the library of models

Name	Input	Cpt	Elimination	Parameterisation	Administration	Arguments
bolus_1cpt_VVmkm	IV-bolus	1	Michaelis-Menten	V, Vm, km	sd	-
infusion_1cpt_VVmkm	IV-infusion	1	Michaelis-Menten	V, Vm, km	sd md	doseMM,TInf doseMM,TInf, tau
oral1_1cpt_kaVVmkm	1st order	1	Michaelis-Menten	ka, V,Vm, km	sd md	doseMM doseMM,tau
bolus_2cpt_Vk12k21Vmkm	IV-bolus	2	Michaelis-Menten	V, k12, k21, Vm, km	sd	-
bolus_2cpt_V1QV2Vmkm	IV-bolus	2	Michaelis-Menten	V1, Q, V2, Vm, km	sd	-
infusion_2cpt_Vk12k21Vmkm	IV-infusion	2	Michaelis-Menten	V, k12, k21, Vm, km	sd md	doseMM,TInf doseMM,TInf, tau
infusion_2cpt_ V1QV2Vmkm	IV-infusion	2	Michaelis-Menten	V1, Q, V2, Vm, km	sd md	doseMM,TInf doseMM,TInf, tau
oral1_2cpt_kaVk12k21Vmkm	1st order	2	Michaelis-Menten	ka, V, k12, k21, Vm, km	sd md	doseMM doseMM, tau
oral1_2cpt_kaV1QV2Vmkm	1st order	2	Michaelis-Menten	ka, V1, Q, V2, Vm, km	sd md	doseMM doseMM, tau
bolus_3cpt_Vk12k21k31k13Vmkm	IV-bolus	3	Michaelis-Menten	V, k12, k21, k13, k31, Vm, km	sd	-
bolus_3cpt_ V1Q1V2Q2V3Vmkm	IV-bolus	3	Michaelis-Menten	V1, Q1, V2, Q2, V3, Vm, km	sd	-
infusion_3cpt_Vk12k21k13k31Vmkm	IV-infusion	3	Michaelis-Menten	V, k12, k21, k13, k31, Vm, km	sd md	doseMM,TInf doseMM,TInf, tau
infusion_3cpt_V1Q1V2Q2V3Vmkm	IV-infusion	3	Michaelis-Menten	V1, Q1, V2, Q2, V3, Vm, km	sd md	doseMM,TInf doseMM,TInf, tau
oral1_3cpt_kak12k21k13k31Vmkm	1st order	3	Michaelis-Menten	ka, k12, k21, k13, k31, Vm, km	sd md	doseMM doseMM,tau
oral1_3cpt_kaV1Q1V2Q2V3Vmkm	1st order	3	Michaelis-Menten	ka, V1, Q1, V2, Q2, V3, Vm, km	sd md	doseMM doseMM, tau

4.1.2 Library of pharmacodynamic models

The library of PD models supports immediate response models (either as a function of observed concentrations, or linked to a pharmacokinetic model) and turnover response models (linked to pharmacokinetic models).
The following tables present these models, giving the following elements for each drug model:

- the **name** of the model function in the library
- the parameters used (**parameterisation**)

Examples for the use of the library of pharmacodynamic models are presented in section 4.2.

Immediate response pharmacodynamic models alone

Linear, quadratic, logarithmic, Emax, sigmoid Emax, Imax, sigmoid Imax models with null or constant baseline are available. The list of these models is given in Table 3.
These models are written in closed form and can be used in the case of a model with one response (PD evaluation or optimisation). They are implemented in the file *LibraryPD_PDdesign.r*. Thus, the user has to specify the path of this file in the model file to use this library of models:

source(file.path(directory.program,"LibraryPD_PDdesign.r"))

For these models, the design variables are the concentrations or the doses instead of the sampling times.
For example, if one uses a linear drug action model without baseline (**immed_lin_null**) from the library, the model uses one parameters (**Alin**) (see Example 2, section 4.2).

Pharmacodynamic models linked to pharmacokinetic model

In this section, we consider models with two responses, with one response for the PK and the other one for the PD. We thus optimise sampling times for both responses using a PK/PD model. Using the libraries of models, we have four cases to compose the PK/PD model depending on the form for each submodel: either with an analytical form (AF) or a differential equation system (ODE).

Therefore, there are four cases of PK/PD models in PFIM library:

1. PK model with linear elimination (AF) and immediate response PD model (AF)

2. PK model with linear elimination (AF) and turnover response PD model (ODE)

3. PK model with Michaelis-Menten elimination (ODE) and immediate response PD model (AF)

4. PK model with Michaelis-Menten elimination and turnover response PD model (ODE)

Table 3. Immediate response pharmacodynamic models included in the PD library for PD alone and for PK/PD model

Drug action models	**Baseline**			
	Null baseline		**Constant baseline**	
	Name	**Parameterisation**	**Name**	**Parameterisation**
Linear	**immed_lin_null**	Alin	**immed_lin_const**	Alin, S0
Quadratic	**immed_quad_null**	Alin, Aquad	**immed_quad_const**	Alin, Aquad, S0
Logarithmic	**immed_log_null**	Alog	**immed_log_const**	Alog, S0
Emax	**immed_Emax_null**	Emax, C50	**immed_Emax_const**	Emax, C50, S0
Sigmoid Emax	**immed_gammaEmax_null**	Emax, C50, gamma	**immed_gammaEmax_const**	Emax, C50, gamma, S0
Imax	**immed_Imax_null**	Imax, C50	**immed_Imax_const**	Imax, C50, S0
Sigmoid Imax	**immed_gammaImax_null**	Imax, C50, gamma	**immed_gammaImax_const**	Imax, C50, gamma, S0

To use PFIM for design evaluation and optimisation for a PK/PD model, the two models must be in the same format.
If both models are written in closed form (case 1), the user can combine the immediate response pharmacodynamic models in closed form expression from the file *LibraryPD_PKPDdesign.r* with the pharmacokinetic models with first order linear elimination (Table 1) in closed form expression from the file *libraryPK.r*. In the PD functions, the expression of the PK model is given as an argument (see Example 3, section 4.2).
In this case, the user must fill in the *stdin.r* using analytical form options and must specify the paths of the library files in *model.r*:

```
source(file.path(directory.program,”LibraryPK.r”))
source(file.path(directory.program,”LibraryPK_PKPDdesign.r”))
```

For the three other cases, the user has to call a specific function in order to create a system of differential equations describing the corresponding PK/PD model. This function named ***Create_formED()*** is implemented in the file ***CreateModel_PKPDdesign.r*** and has to be used in the model file as follows:

```
source(file.path(directory.program,”CreateModel_PKPDdesign.r”))
create_formED(fun_pk,fun_pd,dose=NA,tau=NA,TInf=NA)
```

The arguments to this function are:

- ***fun_pk*** and ***fun_pd***: the names of the PK and PD models, respectively
- ***dose:*** value of the dose only for a PK model with infusion or oral input (by default: NA)
- ***tau***: dosing interval to specify only for multiple dose conditions (by default: NA)
- ***TInf***: time of infusion to specify only for PK model with infusion input (by default: NA)

The output of this function is a new file, named ***model_created.r***, which is created in the directory currently used. This new file contains a function implementing the differential equation system for the corresponding PK/PD model. This file can be deleted after running PFIM. It will be created/overwritten each time the function ***Create_formED()*** is called.

Because the resulting function is an ODE system, the user must fill in the section corresponding to differential equation options in the input file (see Section 5).

The list of the immediate response PD models in the PFIM library is shown in Tables 3 and 4. The list of the turnover response PD models is given in Table 5.

For the second case, where a PK model with linear elimination is associated to a turnover PD response model (defined using differential equation system), the PK model must be written with a differential equations system as well. Consequently, only some PK models from the Table 1 are implemented in ***CreateModel_PKPDdesign.r***:

- for bolus input, only single dose models
- for infusion input, single dose and multiple dose
- for first order absorption input, single dose and multiple dose

For models with a bolus input, the dose has to be specified in the input file (*stdin.r* by default) as the initial condition of the differential equation system (see Example 10, section 4.2). For models with infusion or first order absorption input, dose has to be specified as an argument of the function ***Create_formED()*** and NOT IN THE INITIAL CONDITION OF THE MODEL

IN THE INPUT FILE (see Example 11, section 4.2). Consequently, it is not possible to specify different doses per group when using models with infusion or first order absorption input from the library. All groups of the design are assumed to have the same dose. Otherwise, the user should use the user defined model option.

Table 4. Immediate response pharmacodynamic models linked to a pharmacokinetic model included in the library*

Drug action models	**Baseline/disease models**					
	Linear progression		**Exponential increase**		**Exponential decrease**	
	Name	**Param.**	**Name**	**Param.**	**Name**	**Param.**
Linear	**immed_lin_lin**	Alin, S0, kprog	**immed_lin_exp**	Alin, S0, kprog	**immed_lin_dexp**	Alin, S0, kprog
Quadratic	**immed_quad_lin**	Alin, Aquad, S0, kprog	**immed_quad_exp**	Alin, Aquad, S0, kprog	**immed_quad_dexp**	Alin, Aquad, S0, kprog
Logarithmic	**immed_log_lin**	Alog, S0, kprog	**immed_log_exp**	Alog, S0, kprog	**immed_log_dexp**	Alog, S0, kprog
Emax	**immed_Emax_lin**	Emax, C50, S0, kprog	**immed_Emax_exp**	Emax, C50, S0, kprog	**immed_Emax_dexp**	Emax, C50, S0, kprog
Sigmoid Emax	**immed_gammaEmax_lin**	Emax, C50, gamma, S0, kprog	**immed_gammaEmax_exp**	Emax, C50, gamma, S0, kprog	**immed_gammaEmax_dexp**	Emax, C50, gamma, S0, kprog
Imax	**immed_Imax_lin**	Imax, C50, S0, kprog	**immed_Imax_exp**	Imax, C50, S0, kprog	**immed_Imax_dexp**	Imax, C50, S0, kprog
Sigmoid Imax	**immed_gammaImax_lin**	Imax, C50, gamma, S0, kprog	**immed_gammaImax_exp**	Imax, C50, gamma, S0, kprog	**immed_gammaImax_dexp**	Imax, C50, gamma, S0, kprog

* In addition to those in Table 3.

Table 5. Turnover response pharmacodynamic models linked to a pharmacokinetic model included in the library

Types of response	**Models with impact on the**			
	Input		**Output**	
	Name	**Parameterisation**	**Name**	**Parameterisation**
Emax	turn_input_Emax	Rin,kout,Emax,C50	turn_output_Emax	Rin,kout,Emax,C50
Sigmoid Emax	turn_input_gammaEmax	Rin,kout,Emax,C50,gamma	turn_output_gammaEmax	Rin,kout,Emax,C50,gamma
Imax	turn_input_Imax	Rin,kout,Imax,C50	turn_output_Imax	Rin,kout,Imax,C50
Sigmoid Imax	turn_input_gammaImax	Rin,kout,Imax,C50,gamma	turn_output_gammaImax	Rin,kout,Imax,C50,gamma
Full Imax[a]	turn_input_Imaxfull	Rin,kout,C50	turn_output_Imaxfull	Rin,kout,C50
Sigmoid full Imax[a]	turn_input_gammaImaxfull	Rin,kout,C50,gamma	turn_output_gammaImaxfull	Rin,kout,C50,gamma

[a] Full Imax means Imax is fixed equal to 1

4.2 Model writing

The structural model should be written in a text file (called "model.r" by default but any name can be used). It can be specified either through an analytical form (as an R expression or an R function) or as a solution of systems of differential equations.

An analytical expression model or differential equation model can be called from PFIM libraries (see section 4.1) or implemented by the users. R functions of models can only be defined by the users and are not available in the pre-implemented model libraries.

4.2.1 Models defined in analytical form through an R expression

Description

In case of analytical form, the model for each response should be written assigned in an object called 'form*i*' where *i* is the letter of the alphabet A,B,C,…. The "form*i*" for all the responses are then grouped in a vector called "form":

```
form<-c(formA,formB,formC,…)
```

If the model for a response is defined over intervals by different expressions, each response should be written as a vector of expressions. Each expression can be defined in an object 'form*I*', where I = 1, 2, 3,…. For example, if the user wants to give three expressions for the first response, he can write as follows:

```
formA<-c(form1,form2,form3)
```

formA can be a model of the PFIM libraries or an user-defined model. In the latter case, the specification of the dose can be anywhere in the analytical expression. The name **dose** should be used unchanged. In the computation of the Fisher information matrix, the dose given in each elementary design will be used. If the user gives a value to the dose directly in the model, then all elementary designs will have the same dose.

Example 1: PK model after multiple dose administration using an analytical form with the library of models

```
source(file.path(directory.program,"LibraryPK.r"))
form1<-oral1_1cpt_kaVCl_md(N=1,tau=12) [[1]]
form2<-oral1_1cpt_kaVCl_md(N=2,tau=12) [[1]]
form3<-oral1_1cpt_kaVCl_md(N=3,tau=12) [[1]]
form4<-oral1_1cpt_kaVCl_md(N=4,tau=12) [[1]]
form5<-oral1_1cpt_kaVCl_md(N=5,tau=12) [[1]]
formA<-c(form1,form2,form3,form4,form5)
form<-c(formA)
```

*Note: In this illustration, the user creates a one response model using the model function implemented in the pharmacokinetic library (**Oral1_1cpt_kaVCl**) describing a one compartment oral absorption after a multiple dose administration (**md**). N and tau are the arguments to be specified by the user in the function model. Here, there are five oral administration doses with an interval between two doses equal to twelve hours. The vector of time intervals of each expression needs to be defined in the input file:*

```
boundA<-list(c(0,12),c(0,12)+12,c(0,12)+2*12,c(0,12)+3*12,c(0,12)+4*12)
```

Example 2: PD model using an analytical form with the library of models

```
source(file.path(directory.program,dirsep,"LibraryPD_PDdesign.r”))
formA<-immed_lin_null()[[1]]
form<-c(formA)
```

*Note: In this illustration, the user creates a one response model using the model function implemented in the library (**immed_lin_null**) describing an immediate response model with a linear drug action and without baseline.*

Example 3: PK model with a linear elimination and immediate response PD model

```
source(file.path(directory.program,dirsep,"LibraryPK.r"))
source(file.path(directory.program,dirsep,"LibraryPD_PKPDdesign.r”))
formA<-bolus_1cpt_Vk()[[1]]
formB<-immed_lin_null(formA)[[1]]
form<-c(formA, formB)
```

Note: In this illustration the user creates for the PK model, a one compartment model with bolus input and first order elimination for a single dose, and for the PD model, an immediate response model with a linear drug action and no baseline is used. As shown in the example, the PK model is given as an argument of the PD model. Thus, in the PD model the drug concentration corresponds to the expression of the PK model.

Example 4: PK model using an analytical form with user-defined expression

```
formA<-expression((dose/v*ka)/(ka-ke)*(exp(-ke * t) - exp(-ka*t)))
form<-c(formA)
```

Note: In this illustration, the user creates a one response model describing a one compartment oral absorption with expression. The dose here needs to be specified in the input file.

If the dose is defined directely in the model expression as below, all elementary designs will have the same dose (100 dose unit).

```
formA<-expression((100/v*ka)/(ka-ke)*(exp(-ke * t) - exp(-ka*t)))
form<-c(formA)
```

4.2.2 Models defined in analytical form through an R function

Description

The R function for a PFIM model should take the following form:

```
formA<-function(t,p,X) {
. . .
}
```

The function has 3 arguments
- a vector of times t
- a vector of parameters p
- a scalar X which represents **the dose**

Within the function, the user can define local variables and use the parameters provided in vector p. However, **the header to the function and**

its name must remain unchanged. The order of the parameters is provided by the user through the parameter vectors in the stdin file. The function returns a vector of predictions of each time point in t, computed using the dose X and the parameters p.

Example 5: <u>PK model after single dose administration using an analytical form with user-defined R function</u>

```
formA<-function(t,p,X){
ka<-p[1]
k<-p[2]
V<-p[3]
y<-(X/V*ka/(ka-k)*(exp(-k*t)-exp(-ka*t)))
return(y)
}

form<-formA
```

Note: In this illustration, the user creates a function of a one response model describing a one compartment oral absorption.

Example 6: <u>PK model after multiple dose administration using an analytical form with user-defined R function</u>

```
form<-function(t,p,X){
ka<-p[1]
V<-p[2]
Cl<-p[3]

N<-5
tau<-12

y<-0
for (n in 1:N)
  {
  indic<-t>=(n-1)*tau
  yn<-indic*(X/V*ka/(ka-Cl/V)*(exp(-Cl/V*(t - (n - 1) * tau))-exp(-ka*(t -
(n - 1) * tau))))
  y<-y+yn
  }
return(y)
}
```

Note: In this illustration, the user creates a function of one response model describing a one compartment oral absorption after five administration doses with a between dose interval equal to twelve hours. The number of doses and the between dose interval are defined within the function. They can also be defined as fixed parameters included in the vector p (see Section 5 for more details on fixed parameters).

4.2.3 Models defined through a differential equation system

Description

Model defined as a solution of a differential equation system must be called "formED" and can be called from the PFIM libraries or defined by the users. In the latter case, the user need to write an R function in a format suitable for the solver package deSolve and using the following form:

```
formED<-function(t,y,p)
{
      ...

      ...

      ...
}
```

Again the user may modify anything within this function but **the name and header must remain unchanged.**

The function formED has 3 arguments:

- a vector of time t
- the current estimate of the variables in the ode system y
- a vector of parameters p

Within the function, the user has to define the name of the parameters in vector p and the differential equation system.

The function returns a list with 2 elements:

- the first element is a vector giving the values of the derivatives for each equation in the differential equation system, computed for each time point in t using the parameters p
- the second element is a vector of predictions computed for each time point in t using the parameters p; in PFIM, this vector contains the response(s) we are observing

The initial values of the system have to be specified in the input file **stdin.r** presented in the section 5, uder the name **condinit.**

The implementation of differential equations system requires the use of the lsoda function included in the library "deSolve" (R. Thomas Petzoldt) and of the fdHess function included in the library "nlme" developed by Jose Pinheiro and Douglas Bates.
The lsoda function uses a function of the same name written in Fortran by Linda R. Petzold and Alan C. Hindmarsh. This function solves system of differential equations using the Adams method, a predictor – corrector method for non-stiff systems; it uses the Backward Differentiation Formula (BDF) for stiff systems. The fdHess is used for numerical derivation. It evaluates an approximate gradient of a scalar function using finite differences.

Example 7.1: PK model with bolus input using a differential equation form from the library of models

```
source(file.path(directory.program,dirsep,"LibraryPK.r"))
formED<-bolus_1cpt_VVmkm()
```

*Note: In this illustration, the user creates a one response model using the model function implemented in the pharmacokinetic library (**bolus_1cpt_VVmkm**) describing a one compartment bolus input with Michaelis-Menten elimination after a single dose administration (**sd**). The dose is specified in a part of the R-script file stdin.r (see section 5 for more input details):*

```
time.condinit<-0
condinit<-expression(c(100)) # dose=100
```

Example 7.2: PK model with infusion input using a differential equation form from the library of models

```
time.condinit<-0
condinit<-expression(c(0))
source(file.path(directory.program,dirsep,"LibraryPK.r"))
formED<-infusion_1cpt_VVmkm(doseMM=100, Tinf=1)
```

*Note: In this illustration, the user creates a one response model using the model function implemented in the pharmacokinetic library (**infusion_1cpt_VVmkm**) describing a one compartment infusion input with Michaelis-Menten elimination after a single dose administration (**sd**). The dose is specified as an argument of the PK function in the file model.r, not in the initial condition described in a part of the R-script file stdin.r.*

Example 8: PK model using a differential equation system created by the user

```
formED<-function(t,y,p)
             {
                   ka<-p[1]
                   km<-p[2]
                   Vm<-p[3]
                   V<-p[4]

                   yd1<--ka*y[1]
                   yd2<-+ka*y[1]- V * (Vm * y[2]/(V * km + y[2]))

                   list(c(yd1,yd2),c(y[[2]]/V))
             }
```

Note: This function formED implements a one compartment model with first order absorption and Michaelis-Menten elimination. The dose is specified as an argument of the PK function in the file model.r, not in the initial condition described in a part of the R-script file stdin.r.

The first four lines in the body of the function assign model parameters from the vector p. The next two lines describe the derivatives of the system (yd1 and yd2). More specifically, each derivative represent the drug concentration in the specific compartment at the instant t, and its elements can be either positive or negative.The notation ydX denotes the derivative of the variable in compartment X while the notation y[X] denotes the quantity in the same compartment (see documentation for the deSolve package for details).The last line defines the elements returned by the function:

- *the first item is mandatory for the deSolve package, and should always consist of a vector with the derivatives of the system (here, the two elements yd1 and yd2)*
- *the second item defines the response, here the concentration in the second (central) compartment which is defined by the quantity in this compartment (y[2]) divided by the volume of distribution V. Several responses can be given.*

Example 9: **<u>PK model after multiple dose administration using a differential equation system created by the user</u>**

```
formED<-function(t,y,p)
{
      ka<-p[1]
      V<-p[2]
      Cl <-p[3]

      tau<-12
      input_oral1<-function(ka,V,dose,n,tau,t){
            if(n==0){return(dose*ka/V*exp(-ka*t))}
            else{return(dose*ka/V*exp(-ka*(t-
      n*tau))+input_oral1(ka,V,dose,n-1,tau,t))}
      }
      n<-t%/%tau
      input<-input_oral1(ka,V,dose,n,tau,t)

      dy<--Cl/V*y[1]+input

      list(c(dy),c(y[1]))
}
```

Note: In this illustration, the user creates a function of one response model describing a one compartment oral absorption after multiple dose administration with a between dose interval between two doses equal to twelve hours. The number of doses and the between dose interval are defined within the function.

Example 10: **<u>PK model with bolus input, linear elimination and turnover response PD model</u>**

```
source(file.path(directory.program,dirsep,"CreateModel_PKPDdesign.r"))
create_formED(bolus_1cpt_Vk,turn_input_Imax)
```

Note: In this example, the user creates a PK/PD model with a one compartment bolus input for the PK and a turnover response model with an inhibition on the input for the PD, using the function create_formED. The dose is specified as initial condition of the differential equation system in the R-script file stdin.r.

Example 11: **<u>PK model with infusion input, Michaelis-Menten elimination and immediate response PD model</u>**

```
source(file.path(directory.program,dirsep,"CreateModel_PKPDdesign.r"))
create_formED(infusion_1cpt_VVmkm,immed_lin_null,doseMM=100,TInf=1)
```

*Note: In this illustration, the user creates a one response model using the model function implemented in the pharmacokinetic library (**infusion_1cpt_VVmkm**) describing a one compartment infusion input with Michaelis-Menten elimination after a single dose administration (**sd**). The dose is specified as an argument of the PK function in the file model.r, not in the initial condition in the R-script file stdin.r.*

5 Input

This section shows the common objects required for both design evaluation and optimisation. One input file has to be filled called by default: stdin.r.

In the file stdin.r, the following R objects must be created:

- **project**:	character string indicating the name of the project
- **file.model**:	character string indicating where to find the structural model
- **output**:	character string indicating in which file the results should be printed
- **outputFIM**:	character string indicating in which file the Fisher information matrix should be saved

To specify evaluation or optimisation designs, the user has to complete the "run" object:

- **run**: character string for function selection:
  - -"**EVAL**" for evaluation
  - -"**OPT**" for optimisation

According to the choice, specific objects notified below must be specified.

5.1 Option for Fisher information expression

You have to choose the option for the Fisher information matrix, that is population, individual or bayesian and complete the "**FIM**" object:

- **FIM**: type of the Fisher information matrix:
  P for population
  I for individual
  B for Bayesian

In the case of population design (*i.e.*, population Fisher matrix), you can complete the "**previous.FIM**" object if previous information is available. If it is not the case, leave it as the default.

- **previous.FIM**: character string indicating the name of the file containing the previous information

To compute Fisher information matrix, you have to complete the "option" object:

- **option**: type of option:
  1 for block diagonal Fisher information matrix
  2 for complete Fisher information matrix

Then, you have to complete the "nr" object:

- **nr**: value indicating the number of responses in the model

According to the choice, specific objects notified below must be specified.

5.2 Structural model option

- **modelform**: character string indicating either a model given under differential equation systems (“**DE**”) or analytical form (“**AF**”)

For analytical model specification only

- **dose.identical**: logical value: ‘T’ if the dose is the same for all elementary designs, ‘F’ if not.
- **dose**: value of the dose if dose.identical==T; if not, vector
  of the q doses for each elementary design.
  If one uses infusion models implemented in the library of models, the dose has to be specified here. The rate of infusion is computed in the function model by the expression: dose/TInf.
- **bound*i***: vector of bounds specific to each response *i* with i=A,B,C,…. If the model is defined over intervals by different expressions, give in that vector, the values of each time interval for the response *i*. The first expression will be use for the first time interval, the second expression for the second time interval …
- **NUM**: logical value: ‘T’if using numerical derivatives,then the model must be written by the user using R function in model.r file, ‘F’ if not, then the model is specified via the object "form" which is a vector of expressions

For differential equations system specification only

- **time.condinit**: initial time at which initial conditions are given.
- **condinit.identical**: logical value: ‘T’ if the initial conditions are the same whatever the elementary design, ‘F’if not.
- **condinit**: initial values of the system at the initial time, given into an expression. If condinit.identical==T, enter once the expression of the initial values of the system at the initial time; else, enter the vectors of the initial conditions for each elementary design. If initial values depend on parameters to be estimated, enter this parameter into the expression without any quotation marks
- **RtolEQ**: relative error tolerance, either a scalar or an array as long as ‘y’. See details in help for lsoda function. Default value is 1e-06.
- **AtolEQ**: absolute error tolerance, either a scalar or an array as long as 'y'. See details in help for lsoda function. Default value is 1e-.06

- **Hmax**: an optional value for the maximum integration stepsize. Default value is Inf.

*From help for lsoda: "The input parameters 'RtolEQ', and 'AtolEQ' determine the error control performed by the solver. The solver will control the vector *e* of estimated local errors in *y*, according to an inequality of the form max-norm of (*e*/*ewt*) <= 1, where *ewt* is a vector of positive error*
*weights. The values of 'RtolEQ' and 'AtolEQ' should all be non-negative. The form of *ewt* is:*

RtolEQ* * abs(*y*) + *AtolEQ

where multiplication of two vectors is element-by-element.

If the request for precision exceeds the capabilities of the machine, the Fortran subroutine lsoda will return an error code; under some circumstances, the R function 'lsoda' will attempt a reasonable reduction of precision in order to get an answer. It will write a warning if it does so."

5.3 Statistical model option

- **parameters**: vector of p character strings for the names of the fixed effects parameters
- **beta**: vector of the p fixed effects parameters values
- **beta.fixed**: p-vector indicating if the parameter is estimated or not
- **omega**: vector of the p variances of the random effects should be given
- **n_occ**: integer indicating the number of occasions. Example: **n_occ=2**
- **gamma**: vector of the *p* variances of the random effects for inter-occasion variability.

- ***sig.interi*** and ***sig.slopei***: values of the parameters for the residual variance error model given by $\mathrm{var}(\varepsilon_i)=(\sigma_{\mathrm{int}\,er\,i}+\sigma_{slope_i}*f_i)^2$ for each response *i*, with i = A,B,C,….

- **Trand:** type of between-subject variance (or random effects) model:
  - 1 for additive between-subject variance model
  - 2 for exponential model

If the user wants to deal with covariates which do not change with occasion, he has to specify the following object.

- **covariate.model**: logical value; if T, covariates are added to the model

If the user has filled in by T the previous object, he has to specify the following objects:

- **covariate.name**: list of character indicating the name of the covariate(s) Example:
  covariate.name<-list(c(“Gender”))

- **covariate.category**: list of vectors of categories. Each vector is associated to one covariate and defines its corresponding categories. They can be written as character or integer. Example:
  covariate.category<-list(Gender=c(“F”,”M”))

- **covariate.proportions**: list of vectors of proportions. Each vector is associated to one covariate and defines the corresponding proportions of subjects involved in each corresponding categories. Example:
  covariate.proportions<-list(Gender=c(0.5,0.5))

- **parameter.associated**: list of vectors of parameter(s) associated with each covariate. Each vector is associated to one covariate and is defined by the corresponding parameters on which is added the covariate. Example:
  parameter.associated<-list(Gender=c(Cl, V))

  Name of the parameter(s) has to be identical to those entered in the object parameters.

- **beta.covariate**: list of the values of parameters for all other categories than the reference category (for which beta=0. Example:
  beta.covariate<-list(Gender=list(c(0.5,0.6)))

If the user wants to deal with covariates which change with occasion, he has to specify the following object.

- **covariate_occ.model**: logical value; if T, covariates changing with occasion are added to the model

If the user has filled in by T the previous object, he has to specify the following objects:

- **covariate_occ.name**: list of character indicating the name of the covariate(s) Example:
  covariate_occ.name<-list(c(“Treat”))

- **covariate_occ.category**: list of vectors of categories. Each vector is associated to one covariate and defines its corresponding categories. They can be written as character or integer. Example:
  covariate_occ.category<-list(Treat=c(“A”,”B”))

- **covariate_occ.sequence**: list of vectors of sequences. Each vector is associated to one sequence of values of covariates at each occasion. The size of each sequence has to be equal to the number of occasions (**n_occ**) for each covariate. Example:
covariate_occ.sequence<-list(Treat=list(c("A","B"),c("B","A"))

- **covariate_occ.proportions**: list of vectors of proportions. Each vector is associated to one covariate and defines the proportions of elementary designs corresponding to each sequence of covariate values. The size of each vector has to be equal to the number of sequences. Example:
covariate_occ.proportions<-list(Treat=list(0.5,0.5))

- **parameter_occ.associated**: list of vectors of parameter(s) associated with each covariate. Each vector is associated to one covariate and is defined by the corresponding parameters on which is added the covariate. Example:
parameter_occ.associated<-list(Treat=c(Cl))

Name of the parameter(s) has to be identical to those entered in the object parameters.

- **beta.covariate_occ**: list of the values of parameters for all other categories than the reference category for which beta=0. Example:
beta.covariate_occ<-list(Treat=list(c(log(1.1)))

5.4 Design

proti: list of vectors of elementary designs for each response *i* with i = A,B,C,…. Each vector contains the sampling times of the corresponding elementary design for the respective response. The size of the vector for each response has to be the same.
For example, if there are two responses, the user must specify:

prot*A*<-…
prot*B*<-…

If there are several responses with several elementary designs, the user can specify by NULL if a group do not have samples for one response. Example:

prot*A*<-list(c(1,3,6,12),c(18,20,24))
prot*B*<- list(c(1,3,6,12),c(NULL))

In the present version, the Fedorov-Wynn does not work when there is a NULL design for one response.

- **subjects**: vector of the q numbers of subjects for each elementary design.

In the case of optimisation, this object is a vector of the q initial proportions (if subjects.input=2) or of the numbers of subjects in each elementary design (if subjects.input=1).

- **subjects.input**: **1** if the subjects per elementary designs are given as numbers, **2** if they are given as proportions

- **Ntot:** total number of samples. Ntot is required if the subjects per elementary design are given as proportions, i.e., only if subjects.input=2

5.5 Objects required only for computation of power and number of subjects needed

To compute the expected power to detect covariate effects as to compute the number of subjects needed to achieve a given power, the previous object **covariate.model** has to be filling in by T.

Additional R objects are required to be created.

The following object is needed for both options

- **alpha**: the value of the type one error for the Wald test. Example: **alpha<-0.05**

It is possible to compute either the expected power only or the number of subjects needed for a given power or both of them together.

- **compute.power** logical value, if T the expected power **for comparison test** is computed for each covariate. Example: **compute.power<-T**

- **compute.nni** logical value, if T the number of subjects needed for a given power **for comparison test** is computed for each covariate. Example: **compute.nni<-T**

- **interval_eq** vector of equivalence interval. Example: **interval_eq<-c(log(0.8),log(1.25))**

- **compute.power_eq** logical value, if T the expected power **for equivalence test** is computed for each covariate. Example: **compute.power_eq<-T**

- **compute.nni_eq** logical value, if T the number of subjects needed for a given power **for equivalence test** is computed for each covariate. Example: **compute.nni_eq<-T**

- **given.power** the value of the given power for comparison and/or equivalence test. Example: **given.power<-0.9**

5.6 Graph option

This list of objects allows to draw a graph with the evaluated design (evaluation step) or the optimised design (optimisation step).
Options for plots of sensitivity functions have been added in version 4.0.
If the user wants a graph, he has to specify it in the following object.
- **graph.logical**: logical value; if T, draws the graph of the predicted output(s) and the sampling times.
- **graphsensi.logical**: logical value; if T, draws the graph of the sensitivity function (first order derivation of the model) with respect to each parameter

The user can choose to display only the graphs of models and/or sensitivity functions without computing the Fisher information matrix
- **graph.only**: logical value; if T, draws the graph of the predicted output(s) and/or the sensitivity functions (no design evaluation or optimisation)

If the user has filled in by T the previous object, he has to specify the following objects.

- **names.datax**: character vector for the names of X axis for each graph that corresponds to each type of measurement (the length of this vector must be equal to the number of responses).

- **names.datay**: character vector for the names of Y axis for each graph that corresponds to each type of measurement (the length of this vector must be equal to the number of responses).

- **log.logical**: character string for controls logarithmic axes for the graphical representation of the models. Values "xy", "x" or "y" produce log-log or log-x or log-y axes. Standard graphic is given by log.logical<-F
- ***graph.infi*** and ***graph.supi***: vector of lower and upper sampling times for the graphs for each response *i* with i=A,B,C, …. For example for a single response model, representation in the interval [0-60] is specified by **graph.infA<-c(0)** and **graph.supA<-c(60). If any lower and upper sampling times for the graph are specified, by default for each response the lower sampling time is 0 and the upper sampling time is the maximum sampling time of the initial design.**
- **y.range**: vector of lower and upper values for the graphical representations. They are identical of each response. By default, the value is NULL. For example, representation in 1 interval [0-10] is specified by **y.range=c(0,10)**.

5.7 Objects required only for optimisation

The user can optimise design with identical time in each elementary design for each response. To trigger this option, the user needs to specify it in the next objects:

- **identical.times**: **T** if identical sampling times for each response within an elementary design else **F**

The choice of the algorithm is specify in the following object.

- **algo.option**: character string for algorithm selection. "**SIMP**" for Simplex algorithm selection; "**FW**" for Fedorov-Wynn algorithm selection

For each algorithm, the user need to specify particular objects described below.

For Simplex algorithm specification only

- **subjects.opt**: logical value: 'T' is the optimisation of the proportions or the number of subjects is required; 'F' if not.

- **loweri** and **upperi**: vector of lower and upper admissible sampling times for each response i with i = A,B,C, …. For example if there is a single response model, representation in 2 intervals [0-12], and [48-60] is specified by upperA<-c(0,48) and lowerA<-c(12,60)

- **delta.time**: numeric value for the minimum delay between two successive sampling times

- **iter.print**: logical value to print the iterations (T) or not (F)

- **simplex.parameter**: percent of change from initial design for the initial vertices of the simplex algorithm building (default 20%).

- **Max.iter**: a positive integer specifying the maximum number of iterations allowed (default = 5000)

- **Rctol**: a positive numeric value specifying the tolerance level for the relative convergence criterion of the Simplex algorithm (default = 1e-6)

For Fedorov-Wynn algorithm only

- **nwind*i***: numeric value for the number of sampling intervals for each response i with i=A,B,C, …

- **sampwin*i***: list of vector of the allowed sampling times for each sampling interval for each response i with i = A,B,C, …

- **nsamp*i***: list of vector of allowed numbers of points to be taken from each sampling interval for each response i with i = A,B,C, …
- **fixed.times*i***: list of times which will be in all evaluated protocols, corresponding to fixed constraints for each response i=A,B,C,…
- **nmaxpts*i***: numeric value for the maximum number of sampling times per subject for each response i with i = A,B,C, …

- **nminpts*i***: numeric value for the minimum number of sampling times per subject for each response i with i = A,B,C, …

For Fedorov-Wynn algorithm use, the initial population design specified in prot must involve elementary designs with number of samples per subject and sampling times in accordance with sampwin, nsamp, nmaxpts and nminpts for each response.

6 Output

The results are written in the output file called stdout.r by default or with the name specified in the input file. This file is different when only evaluation is performed or when optimisation is performed. It is detailed in next sections respectively for evaluation and for optimisation.

6.1 Evaluation output file and objects

Figure 1 represents the output file from the design evaluation built on the Example documentation in the sections 1.2.4 and 1.6.1.

The user can read on the Figure 1:

1 The name of the function used: PFIM 4.0.

2 The name of the project and the date.

3 A summary of the input: response(s), the evaluated population or individual/Bayesian design for each response, doses or initial conditions (in case of models defined by differential equation system) and number of subjects corresponding to those designs, between-subject variance model and residual error model for each response(s), error tolerances for the solver of differential equations system if used, name of the previous Fisher information matrix if considered.

The figure 1 shows a one response model(written in user-defined model form) with a group described by three sampling times for 200 subjects, considering a previous Fisher information matrix. The parameter k_a is fixed (assuming no variability on k_a) and the dose is equal to 100.

4 The population or individual or Bayesian Fisher information matrix, a dim*dim symmetric matrix where dim is the total number of population parameters to be estimated, the number of individual parametres + the number of error model parameters or only the number of individual parameters respectively. The name of the file where is possibly saved the Fisher information matrix is given.

5 The value of each parameter with the expected standard error (StdError) and relative standard error (RSE). In case of Bayesian design, the associated shrinkages values are also reported.

6 The value of the determinant of the Fisher information matrix and the value of the criterion (determinant$^{(1/dim)}$) where dim is defined in 4

7 The eigenvalues of the Fisher information matrix and the correlation matrix.

```
PFIM 4.0                                                                        1

Project:  Doc_example                                                           2

Date:  Fri Mar 21 13:02:23 2014


**************************** INPUT SUMMARY ********************************

Analytical function models :

function(t,p,X){
ka<-p[1]
k<-p[2]
V<-p[3]
y<-(X/V*ka/(ka-k)*(exp(-k*t)-exp(-ka*t)))
return(y)
}


Design:
Sample times for response: A
       times subjects doses
1 c(1, 3, 8)      200   100                                                     3


Random effect model: Trand =  2

Variance error model response A : ( 0.5 + 0.15 *f)^2




Computation of the Population Fisher information matrix: option =  1

Previous FIM from file Previous_FIM.txt

FIM saved in FIM.txt


****************** FISHER INFORMATION MATRIX *****************

             V1        V2        V3        V4        V5         V6
[1,] 17523.1601 113.42921    0.0000    0.0000    0.0000     0.0000
[2,]   113.4292  11.91056    0.0000    0.0000    0.0000     0.0000
[3,]     0.0000   0.00000 1499.7312  228.5511  420.7733   594.9001             4
[4,]     0.0000   0.00000  228.5511 8988.9175  701.5138  2780.3955
[5,]     0.0000   0.00000  420.7733  701.5138 2114.1728  4049.3932
[6,]     0.0000   0.00000  594.9001 2780.3955 4049.3932 12315.5527
```

```
************************* EXPECTED STANDARD ERRORS ************************

------------------------ Fixed Effects Parameters -------------------------

   Beta    StdError      RSE
k  0.25 0.007798489 3.119395 %
V 15.00 0.299123566 1.994157 %


------------------------- Variance of Inter-Subject Random Effects --------

  omega²   StdError      RSE                                                   5
k   0.25 0.02669459 10.67783 %
V   0.10 0.01098554 10.98554 %


------------------------ Standard deviation of residual error -------------

           Sigma   StdError       RSE
sig.interA  0.50 0.03670921  7.341841 %
sig.slopeA  0.15 0.01527153 10.181022 %

******************************* DETERMINANT ******************************

2.199791e+19
                                                                               6
******************************** CRITERION *******************************

1673.903


******************* EIGENVALUES OF THE FISHER INFORMATION MATRIX **********

        FixedEffects VarianceComponents
min     15109.231479           11.17585
max     17523.894830         1509.84067
max/min     1.159814          135.09852


******************* CORRELATION MATRIX *****************
                                                                               7
          [,1]       [,2]        [,3]        [,4]        [,5]        [,6]
V1  1.0000000 -0.2482863  0.00000000  0.00000000  0.00000000  0.00000000
V2 -0.2482863  1.0000000  0.00000000  0.00000000  0.00000000  0.00000000
V3  0.0000000  0.0000000  1.00000000 -0.04548621 -0.21289290  0.09121384
V4  0.0000000  0.0000000 -0.04548621  1.00000000  0.09083326 -0.23035289
V5  0.0000000  0.0000000 -0.21289290  0.09083326  1.00000000 -0.78714187
V6  0.0000000  0.0000000  0.09121384 -0.23035289 -0.78714187  1.00000000
```

Figure 1. Example of design evaluation output file

6.2 Optimisation output file and objects

Figure 2 represents the output file corresponding to the optimal Bayesian design described in the Example documentation in the section 1.3.2.

The user can read on the Figure 2:

1 The name of the function used: PFIM 4.0

2 The name of the project and the date.

3a A summary of the input: structural model, between-subject and error variance model, initial design, initial numbers or proportions of subjects and doses, total number of allowed samples, criterion associated to the initial design.

3b Sampling times specifications (according to the algorithm used) within which the optimal samples will be chosen and error tolerances for the solver of differential equations system if used.

4 The optimised design and the associated criterion.

For the simplex algorithm, the number of iterations performed and the number of function evaluations, the status of the convergence (false or achieved) are reported

For the Fedorov-Wynn algorithm for optimal population design, the optimal group structure with the proportion of subjects and the equivalence in number are then reported. The best one group protocol is also always reported with associated criterion.

When optimising a Bayesian or an individual design, the resulted design correspond to the best one group protocol.

5 The population or individual or Bayesian Fisher information matrix, a dim*dim symmetric matrix where dim is the total number of population parameters to be estimated, the number of individual parametres + the number of the error model parameters or only the number of individual parameters respectively. The name of the file where is possibly saved the Fisher information matrix is given.

6 The value of each parameter with the expected standard error (StdError) and relative standard error (RSE). In case of Bayesian design, the associated shrinkages values are also reported.

7 The value of the determinant of the Fisher information matrix and the value of the criterion (determinant^(1/dim)) where dim is defined in 5

8 The eigenvalues of the Fisher information matrix and the correlation matrix.

```
PFIM 4.0                                                                    1

Project: Example Optimisation
                                                                            2
Date: Thu Jul 31 09:22:17 2014


*************************** INPUT SUMMARY ********************************

Analytical function model:

function(t,p,X){
ka<-p[1]
k<-p[2]
V<-p[3]
y<-(X/V*ka/(ka-k)*(exp(-k*t)-exp(-ka*t)))
return(y)
}



Initial design:


Sample times for response: A                                               3a
              Protocol subjects doses
1 c=(0.33, 1.5, 5, 12)        1   100


Total number of samples: 4

Associated criterion value: 3.5272

Identical sampling times for each response: FALSE

Random effect model: Trand =  2

Variance error model response A : ( 0.5 + 0.15 *f)^2


Optimization step:

Sampling windows for the response: A
Window 1 : t= 0.33 1 1.5 3 5 8 12
    Nb of sampling points to be taken in this window, n[ 1 ]= 4            3b
Maximum total number of points in one elementary protocol : 4
Minimum total number of points in one elementary protocol : 4



BEST ONE GROUP PROTOCOL:

Sample times for response: A
               times freq Subjects doses
1 c(0.33, 1.5, 5, 8)    1        1   100                                   4


Associated criterion: 3.8066
```

```
Computation of the Bayesian Fisher information matrix

FIM saved in FIM.txt

****************** FISHER INFORMATION MATRIX ******************
                                                                                  5
          [,1]       [,2]       [,3]
[1,]  1.590507   2.096455 -0.2426030
[2,]  2.096455 354.843266  4.4964361
[3,] -0.242603   4.496436  0.2013882

************************* EXPECTED STANDARD ERRORS ***********************

----------------------- Fixed Effects Parameters ------------------------

    Beta  StdError      RSE   Shrinkage                                           6
ka  2.00 0.9638509 48.19255 %  23.22522 %
k   0.25 0.0688475 27.53900 %  30.33586 %
V  15.00 3.1862487 21.24166 %  45.12080 %


****************************** DETERMINANT *******************************

55.15913

******************************* CRITERION ********************************        7

3.806617


****************** EIGENVALUES OF THE FISHER INFORMATION MATRIX *****************

        FixedEffects VarianceComponents
min     9.552493e-02                 NA
max     3.549127e+02                 NA
max/min 3.715393e+03                 NA
                                                                                  8

****************** CORRELATION MATRIX *****************

           [,1]       [,2]       [,3]
[1,]  1.0000000 -0.4133690  0.5638373
[2,] -0.4133690  1.0000000 -0.6330761
[3,]  0.5638373 -0.6330761  1.0000000
```

Figure 2. Example of design optimisation output file

Moreover, the PFIM() function returns the following R objects:

- **mfisher:** the population or individual or Bayesian Fisher information matrix corresponding to the optimised protocole
- **determinant:** the determinant of the Fisher information matrix
- **crit:** the value of the criterion
- **se:** the vector of the expected standard errors for each parameter
- **cv:** the corresponding coefficient of variation, expressed in percent (relative standard error)
- **sh:** the shrinkage values for each parameter in case of Bayesian design
- **EigenValues**: the eigenvalues of the Fisher information matrix
- **corr.matrix**: the correlation matrix

6.3 Comments

- It is now possible to visualise the graphs of the model and the sensitivity functions without performing evaluation or optimisation (see Example Documentation)

- For an infusion model, it is not possible for design evaluation to include the time at the end of the infusion when the end of infusion parameter is a parameter to be estimated.

- If the bound are not correctly specified according to the number of responses in case of analytical form expression, by default bounds are initialized to c(0, Inf) but that does not appear on the stdin.r file. In case of analytical form defined by R function or of differential equation system, the bound is not used.

- If the between-subject variance of a parameter is assumed to be zero, enter 0 for this variance in omega: PFIM will remove the corresponding row and column in the Fisher information matrix.

- If a parameter is fixed, no variability is assumed and no shrinkage is computed for this parameter

- The number of subjects for each elementary design is the same for all the response(s)

- Optimisation with the Fedorov-Wynn algorithm can be performed with an initial design composed of several groups with different doses or initial conditions and with fixed sampling times

- The dimension of the previous Fisher information matrix is the same as the current Fisher information matrix. The previous Fisher information matrix can be taken into account in evaluation and in optimisation with Simplex algorithm or for best one group protocol

- If a design leads to very poor information with a singular population Fisher information matrix (det=0), the expected standard errors and the RSE are returned as NA.

- Standard error of derived parameters can be computed by the delta method available in the R package “car”, using the FIM stored in files or directly obtained in R console after running PFIM (see Example documentation Section 1.5 for detailed examples)

7 References

1. Mentré F, Chenel M, Comets E, Grevel J, Hooker A, et al. (2013) Current Use and Developments Needed for Optimal Design in Pharmacometrics: A Study Performed Among DDMoRe's European Federation of Pharmaceutical Industries and Associations Members. CPT Pharmacomet Syst Pharmacol 2: e46. doi:10.1038/psp.2013.19.

2. Nyberg J, Bazzoli C, Ogungbenro K, Aliev A, Leonov S, et al. (2014) Methods and software tools for design evaluation for population pharmacokinetics-pharmacodynamics studies. Br J Clin Pharmacol. doi:10.1111/bcp.12352 [Epub ahead of print].

3. Retout S, Duffull S, Mentré F (2001) Development and implementation of the population Fisher information matrix for the evaluation of population pharmacokinetic designs. Comput Methods Programs Biomed 65: 141–151.

4. Bazzoli C, Retout S, Mentré F (2010) Design evaluation and optimisation in multiple response nonlinear mixed effect models: PFIM 3.0. Comput Methods Programs Biomed 98: 55–65. doi:10.1016/j.cmpb.2009.09.012.

5. Nguyen TT, Bazzoli C, Mentré F (2012) Design evaluation and optimisation in crossover pharmacokinetic studies analysed by nonlinear mixed effects models. Stat Med 31: 1043–1058. doi:10.1002/sim.4390.

6. Dumont C, Chenel M, Mentré F (2014) Two-stage adaptive designs in nonlinear mixed effects models: application to pharmacokinetics in children. Commun Stat Simul Comput [Epub ahead of print].

7. Combes FP, Retout S, Frey N, Mentré F (2013) Prediction of shrinkage of individual parameters using the bayesian information matrix in non-linear mixed effect models with evaluation in pharmacokinetics. Pharm Res 30: 2355–2367. doi:10.1007/s11095-013-1079-3.

8. Mentré F, Mallet A, Baccar D (1997) Optimal design in random-effects regression models. Biometrika 84: 429–442. doi:10.1093/biomet/84.2.429.

9. Retout S, Mentré F (2003) Further developments of the Fisher information matrix in nonlinear mixed effects models with evaluation in population pharmacokinetics. J Biopharm Stat 13: 209–227. doi:10.1081/BIP-120019267.

10. Bazzoli C, Retout S, Mentré F (2009) Fisher information matrix for nonlinear mixed effects multiple response models: evaluation of the appropriateness of the first order linearization using a pharmacokinetic/pharmacodynamic model. Stat Med 28: 1940–1956. doi:10.1002/sim.3573.

11. Nelder JA, Mead R (1965) A Simplex Method for Function Minimization. Comput J 7: 308–313. doi:10.1093/comjnl/7.4.308.

12. Fedorov VV (1972) Theory Of Optimal Experiments. Academic Press: New York.

13. Wynn HP (1972) Results in the theory and construction of D-optimum experimental designs. J R Stat Soc Series B 34:133–147.

14. Retout S, Comets E, Samson A, Mentré F (2007) Design in nonlinear mixed effects models: optimization using the Fedorov-Wynn algorithm and power

of the Wald test for binary covariates. Stat Med 26: 5162–5179. doi:10.1002/sim.2910.

15. Bertrand J, Mentré F (2008) Mathematical expressions of the pharmacokinetic and pharmacodynamic models implemented in the MONOLIX software. MONOLIX Software Documentation. www.lixoft.eu.