



AMIGO: Advanced Model Identification using Global Optimization

USER GUIDE

Eva Balsa-Canto and Julio R. Banga
(Bio)Process Engineering Group
IIM-CSIC
SPAIN
E-mail: ebalsa@iim.csic.es
Copyright @ CSIC

March 29, 2011

Contents

1	Brief theoretical introduction	3
1.1	Parameter identification iterative procedure	3
1.2	Elements for parametric identification	4
1.2.1	The model	4
1.2.2	The experimental scheme and data	4
1.3	Ranking of parameters	6
1.4	Parameter estimation	7
1.4.1	Distance measure	7
1.4.2	Single shooting vs multiple shooting	8
1.5	Practical identifiability analysis	9
1.6	Optimal experimental design	10
1.6.1	Control vector parameterization	11
1.7	Numerical methods	12
1.7.1	Initial value problem solvers	12
1.7.2	Nonlinear programming solvers	12
2	AMIGO toolbox description	16
2.1	Toolbox download and License	16
2.2	Toolbox requirements and installation guide	16
2.3	General structure	17
2.4	Summary of features	19
2.5	How to input problems in AMIGO	23
2.5.1	Defining the model	24
2.5.2	Defining the experimental scheme	25
2.5.3	Defining the experimental data and the corresponding error information	28
2.5.4	Inputs for LRank, GRank, ContourP, RIdent and PE	29
2.5.5	Inputs for OED	31
2.5.6	Defining the numerical methods	35
2.6	How to run AMIGO tasks	37
2.6.1	AMIGO_Startup	37
2.6.2	AMIGO_Prep	38
2.6.3	AMIGO_SModel	39
2.6.4	AMIGO_SObs	40
2.6.5	AMIGO_SData	41
2.6.6	AMIGO_LRank	42
2.6.7	AMIGO_GRank	43
2.6.8	AMIGO_PE	44

2.6.9	AMIGO_ContourP	45
2.6.10	AMIGO_RIdent	46
2.6.11	AMIGO_OED	47
A	Illustrative examples	48
A.1	The Hodgking and Huxley model	49
A.1.1	Introduction	49
A.1.2	Input the model to automatically generate FORTRAN or MATLAB	50
A.1.3	Input the model in FORTRAN, MATLAB or SBML	51
A.1.4	Input the model as a blackbox model	55
A.2	A model of the circadian clock in Arabidopsis thaliana	57
A.2.1	Preprocessing the example: AMIGO_Prep('circadian_grank')	61
A.2.2	Solving system dynamics: AMIGO_Smodel('circadian_grank')	62
A.2.3	Simulating the observables: AMIGO_SObs('circadian_grank')	65
A.2.4	Performing the local rank of parameters: AMIGO_LRank('circadian_grank') . .	68
A.2.5	Performing the global rank of parameters: AMIGO_GRank('circadian_grank') . .	73
A.3	A model of the NF κ B module	77
A.3.1	Introduction	77
A.3.2	Generating pseudo-experimental data: AMIGO_SData('nfkb_psdata')	81
A.3.3	Solving the parameter estimation problem: AMIGO_PE('nfkb_pe')	85
A.3.4	Performing the identifiability analysis: AMIGO_ContourP('nfkb_pe')	92
A.3.5	Robust identifiability analysis: AMIGO_RIdent('nfkb_pe')	93
A.4	The model of a three step pathway by Mendes	98
A.4.1	Introduction	98
A.4.2	Parameter estimation under sustained stimulation: AMIGO_PE('mendes_pe') . . .	101
A.4.3	Sensitivity analysis under dynamic stimulation: AMIGO_LRank('mendes_uvar') .	103
A.4.4	Solving the optimal experimental design problem: AMIGO_OED('mendes_oed') . .	105

1

Brief theoretical introduction

1.1 Parameter identification iterative procedure

Mathematical modelling is the art of quantitatively describing from observations particular aspects of the structure and function of a particular process or system. Conferring a predictive character on a given mathematical formulation often relies on determining a number of non-measurable parameters that largely condition the model's response. These parameters can be usually estimated by fitting the model to experimental data: parameter estimation or identification.

However parametric identification of nonlinear dynamic models has revealed as very challenging problem, due mainly to lack of or poor practical identifiability, rooted on the presence of several suboptimal solutions or the presence of multiple equivalent solutions. This has lead to the development of iterative approaches for parametric identification [14, 17, 44, 3, 2], which incorporate some or all of the following steps, identifiability and sensitivity analyses, experimental design and parameter estimation.

AMIGO [4, 5] implements the numerical steps incorporated in the iterative model identification procedure described in [3, 2] (Figure A.4.4).

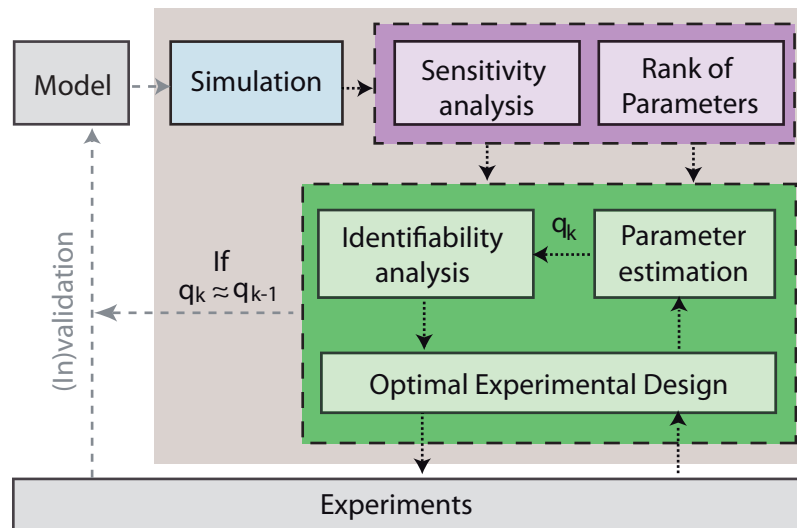


Figure 1.1: AMIGO iterative identification procedure.

It basically covers:

Simulation. To solve the system dynamics for different parameter values under different experimental schemes. This is useful to analyse model tendencies *a priori* and for the (in-)validation step *a posteriori*.

Global ranking of parameters. This step helps to decide which parameters are the most relevant to model output for a given experimental scheme. In the case of lack of structural identifiability, global ranking may be used to make decisions as to reformulate the model or which parameters to estimate.

Parameter estimation. Formulated as a non-linear optimization problem whose objective is to find model unknown parameters (kinetic constants, initial conditions, etc..) so as to minimize a measure of the distance among the model predictions and the experimental data. Unfortunately, since it is usually the case that several sub-optimal solutions are possible, the use of global optimization methods is necessary to somehow guarantee that the best possible solution is located.

Practical identifiability analysis. Enables an evaluation of the possibility of assigning unique values to the parameters from a given set of experimental data or experimental scheme, subject to experimental noise.

Optimal experimental design via dynamic optimization. The purpose of this step is to design dynamic experiments with the aim of maximizing data quality and quantity (as measured by the Fisher information matrix) for the purpose of model calibration.

1.2 Elements for parametric identification

1.2.1 The model

The mathematical model will consist on two essential elements: in one hand the set of differential equations describing the system dynamics, here, we consider a general deterministic nonlinear dynamic model:

$$\mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{u}, \boldsymbol{\theta}, \mathbf{t}) = \mathbf{0} \quad (1.1)$$

and, in the other hand, the observation function, describing the relationship among the states in the model and the available measured quantities:

$$\mathbf{y}^e = \mathbf{g}^e(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}, \mathbf{t}) \quad (1.2)$$

where $\mathbf{x}$ are the state variables; $\mathbf{y}^e$ represents the vector of observables, $\mathbf{u}$ specifies the vector of inputs (i.e. all manipulable variables), $\boldsymbol{\theta}$ is the vector of model parameters being $\boldsymbol{\theta}$ the set of admissible parameters that may be fixed by physical, chemical or biological considerations.

1.2.2 The experimental scheme and data

The experimental scheme (see Figure 1.2 for an illustrative example) collects all information related to the way experimental data are obtained, i.e. the number of experiments, the observed or measured quantities (concentrations, quantities, etc.), the (time-dependent) input profile(s), the experiment(s') duration and the sampling times.

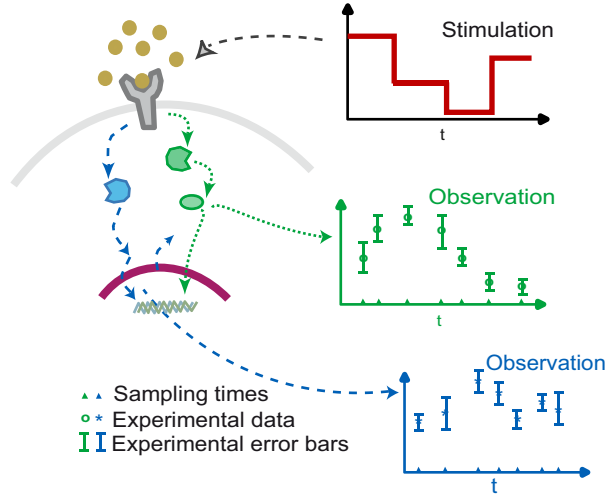


Figure 1.2: Illustrative example of the experimental scheme.

The experimental data consist on matrices of values corresponding to individual measurements obtained under the conditions specified by the experimental scheme ε . For the sake of clarity the experimental data and model predictions corresponding to an experimental scheme will be encoded in the following two vectors:

$$\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_d, \dots, \mathbf{y}_{n_d}] \quad \tilde{\mathbf{y}} = [\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2, \dots, \tilde{\mathbf{y}}_d, \dots, \tilde{\mathbf{y}}_{n_d}] \quad (1.3)$$

where d represents a certain experimental condition defined by the subindexes ε (for the experiment), o (for the observables in the experiment ε) and s (for the sampling times in the experiment ε). n_d represents the total number of such conditions, i.e. the total number of data. Note that the operators to be defined in the sequel can be then easily condensed as follows:

$$\sum_{d=1}^{n_d} (.) = \sum_{\varepsilon=1}^{n_{\varepsilon}} \left(\sum_{o=1}^{n_o^{\varepsilon}} \left(\sum_{s=1}^{n_s^{o,\varepsilon}} (.) \right) \right) \quad (1.4)$$

It is also desirable to provide information about the type and quantity of noise in the experimental data. In this concern replicates of the experiments are often required to determine the variance of the data, which may depend on what is being measured or may be different for every measurement. Output-additive experimental noise is often assumed as follows:

$$\tilde{\mathbf{y}}_d = \mathbf{y}_d + \mathbf{e}_d \quad (1.5)$$

where $\mathbf{e}_d$ belongs to a sequence of independent random variables with a given probability density $\Pi_{e_d}(e_d)$. In many practical examples the $\mathbf{e}_d$ are independent random variables, where the variance σ_d^2 of the noise is either constant or known for all d 's in the so called homoscedastic case, or unknown and dependent on d in the heteroscedastic case. Figure 1.3 illustrates the differences between homoscedastic and heteroscedastic noise.

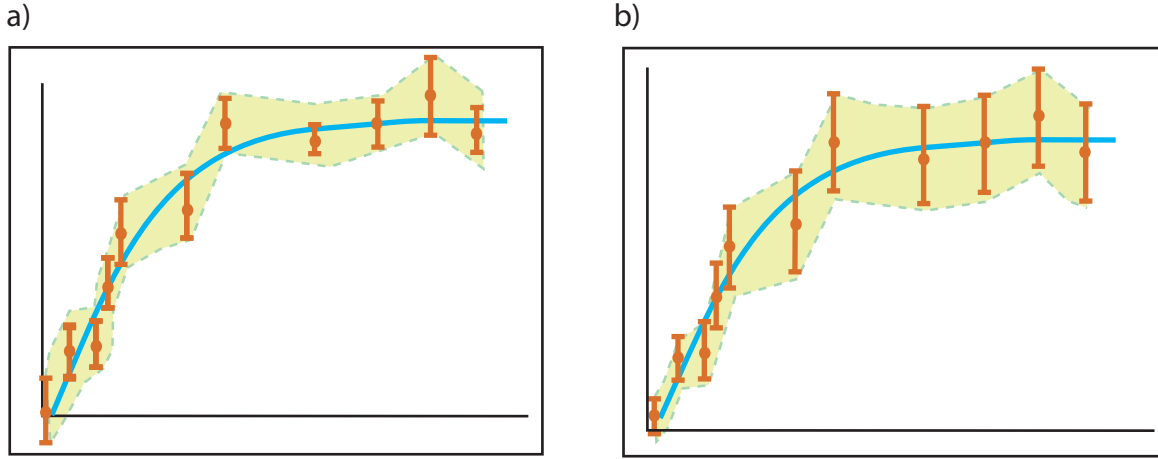


Figure 1.3: Illustrative representation of the a) homoscedastic noise with constant known variance and b) heteroscedastic noise with variance proportional to the observation.

1.3 Ranking of parameters

Observables will depend differently on different parameters and this may be used to rank the parameters in order of their relative influence on model predictions. Such influence may be quantified by the use of parametric sensitivities.

Local parametric sensitivities for a given experiment e , observable o and at a sampling time $t_s^{e,o}$ are defined as follows:

$$S_p^{e,o}(t_s^{e,o}) = \frac{\partial y^{e,o}}{\partial \theta_p}(t_s^{e,o}); \quad p = 1 \dots n_\theta \quad (1.6)$$

The corresponding relative sensitivities, $s_p^{e,o} = \frac{\Delta \theta_p}{\Delta y^{e,o}} \frac{\partial y^{e,o}}{\partial \theta_p}$, can be used to assess the individual local parameter influence or importance, that is to establish a ranking of parameters. Brun and Reichert (2001) [10] suggested several importance factors, that may be generalized for the case of having several observables and experiments [3].

Of course, the values of the parameters are not known *a priori*, and even when optimally computed, optimal values are subject to uncertainty depending on the type of experiments and the presence of experimental noise. Consequently, the ranking for a given value of the parameters may be of limited value. Alternatively, one may compute ranking for a sufficiently large number of parameter vectors in the feasible parameter space.

The simplest approach is to apply a Monte Carlo sampling. By sampling repeatedly from the assumed joint-probability density function of the parameters and by evaluating the sensitivities for each sample, the distribution of sensitivity values, along with the mean and other characteristics, can be estimated. This approach yields reasonable results if the number of samples is quite large, requiring a great computational effort.

An alternative that can yield more precise estimates is Latin hypercube sampling (LHS). This method selects n_{lhs} different values for each of the parameters, which it does by dividing the range of each parameter into n_{lhs} non-overlapping intervals on the basis of equal probability. Next, from each interval one value for the parameters is selected at random with respect to the probability density

in the interval. The n_{lhs} values thus obtained for the first parameter are then paired in a random manner (equally likely combinations) with the n_{lhs} values for the second and successive parameters. This method allows the overall parameter space to be explored without requiring an excessively large number of samples. The importance factors will then read:

$$\delta_p^{msqr} = \frac{1}{n_{lhs}n_d} \sqrt{\sum_{mc=1}^{n_{lhs}} \sum_{d=1}^{n_d} ([s_d]_{mc})^2} \quad (1.7)$$

$$\delta_p^{mabs} = \frac{1}{n_{lhs}n_d} \sum_{mc=1}^{n_{lhs}} \sum_{d=1}^{n_d} |[s_d]_{mc}| \quad (1.8)$$

$$\delta_p^{mean} = \frac{1}{n_{lhs}n_d} \sum_{mc=1}^{n_{lhs}} \sum_{d=1}^{n_d} [s_d]_{mc} \quad (1.9)$$

$$\delta_p^{max} = \sum_{mc=1}^{n_{lhs}} \left[\max_d s_d \right]_{mc} \quad (1.10)$$

$$\delta_p^{min} = \sum_{mc=1}^{n_{lhs}} \left[\min_d s_d \right]_{mc} \quad (1.11)$$

where δ^{msqr} and δ^{mabs} quantify how sensitive a model is to a given parameter considering δ^{mabs} interactions between parameters. δ^{max} and δ^{min} indicate the presence of outliers and provide information about the sign. δ^{mean} provides information about the sign of the averaged effect a change in a parameter has on the model output.

Ordering the parameters according to these criteria, preferably in decreasing order, results in a parameter importance ranking. This information may be useful to decide on reformulating the model or to fix the less relevant parameters to improve either structural or practical identifiability.

Note that the summations will, in general, hide the different effects from the different experiments and observables unless they are in the same order of magnitude. Similar analyses may be performed for experiments and observables, thus providing information on the parameters that are more relevant to a particular observable in a particular type of experiment.

1.4 Parameter estimation

The parameter estimation problem may be formulated as follows:

Find model unknown parameters (kinetic constants, initial conditions, etc.) so as to minimize a given measure of the distance among the model predictions and the experimental data.

1.4.1 Distance measure

The definition of the scalar measure of the distance among the experimental data and the model predictions will depend on the available information for a particular example.

The most well known cost function is the **generalized least squares**, given by:

$$J_{glsq}(\theta) = \sum_{d=1}^{n_d} q_d (y_d(\theta) - \tilde{y}_d)^2 \quad (1.12)$$

where the weighting coefficients $\{q_d\}_{d=1}^{n_d} \geq 0$ are fixed *a priori*. The selection of these parameters will express the relative confidence in the various experimental data and the consequent importance attached to the model performance with regards to each type of measurement, experiment and sampling time. It should be noted that non prior information is required to use the least-squares function.

When information about the nature of the experimental noise is available one may use the **maximum (log-)likelihood function** that looks for the value of the parameters that give the highest probability to the measured data.

$$J_{llk} = \ln (\Pi(\tilde{\mathbf{y}}|\theta)) \quad (1.13)$$

The probability density function (P_i) selected will condition the type of cost function. Under the assumptions of independently identically distributed (i.i.d.) measurements with normally distributed noise, the likelihood is represented by:

$$J_{llk} = \sum_{d=1}^{n_d} \left(-\frac{1}{2} \right) \left[\log(2\pi) + \log(\sigma_d^2) + \frac{(y_d(\theta) - \tilde{y}_d)^2}{\sigma_d^2} \right] \quad (1.14)$$

For the homoscedastic case, for which the variance is known or constant, the cost function results to be similar to the generalized least squares, with weights taken as the inverse of the variance of the experimental data (see details on the derivation in [46]):

$$J_{lsq}(\theta) = \sum_{d=1}^{n_d} \frac{(y_d(\theta) - \tilde{y}_d)^2}{\sigma_d^2} \quad (1.15)$$

The estimation of the constant variances depending on any characteristic of measurement d , usually requires a significant amount of prior experiments and may involve multiple identification problems. This is the reason why in many applications a constant variance is selected for all measurements $\sigma_d = \sigma, \forall d = 1, \dots, n_d$. Of course this approximation may be not realistic and thus requiring a careful analysis of the experimental error properties.

For the heteroscedastic case, for which the variance depends on what is being measured, it is possible to find a functional relationship between the variance and the model predictions. In fact, it has been shown that the power-of-the-mean variance is specially advantageous when the variances increase with the measurements since only two extra-parameters should be estimated. In this case the variance is assumed to obey the formula:

$$\sigma^2(a, b, y(\theta)) = |ay(\theta)|^b \quad (1.16)$$

with $a > 0$ and $0 \leq b \leq 2$. Note that the case of standard deviation proportional to the output corresponds to $b = 2$. The corresponding log-likelihood function reads:

$$J_{llk} = \sum_{d=1}^{n_d} b \log |y_d(\theta)| + \frac{(y_d(\theta) - \tilde{y}_d)^2}{\sigma^2(a, b, y_d(\theta))} \quad (1.17)$$

1.4.2 Single shooting vs multiple shooting

The parameter estimation problem is thus formulated as a non linear optimization problem where the objective is to find the set of model unknowns to minimize a given cost function subject to the system dynamics and possibly bounds on the unknown values. Therefore its numerical solution involves an outer iterative procedure to generate values for the unknown parameters and initial conditions, the nonlinear programming method (NLP) and an iterative procedure to solve the differential equations, the initial value problem (IVP) solver.

In the so called **single shooting approach** the initial value problem is solved from the initial conditions till the final time for all the iterates generated by the NLP solver (see Figure 1.4-a). Alternatively, in the **multiple shooting approach** [9, 36], the duration of the process is partitioned into a number of shooting intervals, in such a way that at least one experimental data may be found in each shooting, and the several initial value problems are to be solved (see Figure 1.4-b). It should be noted that in the multiple shooting the initial conditions for the different intervals are also to be computed during optimization. Therefore the addition of further constraints to the parameter estimation problem is required so as to guarantee that at the optimum the solution is smooth. This leads to a constrained non-linear optimisation problem.

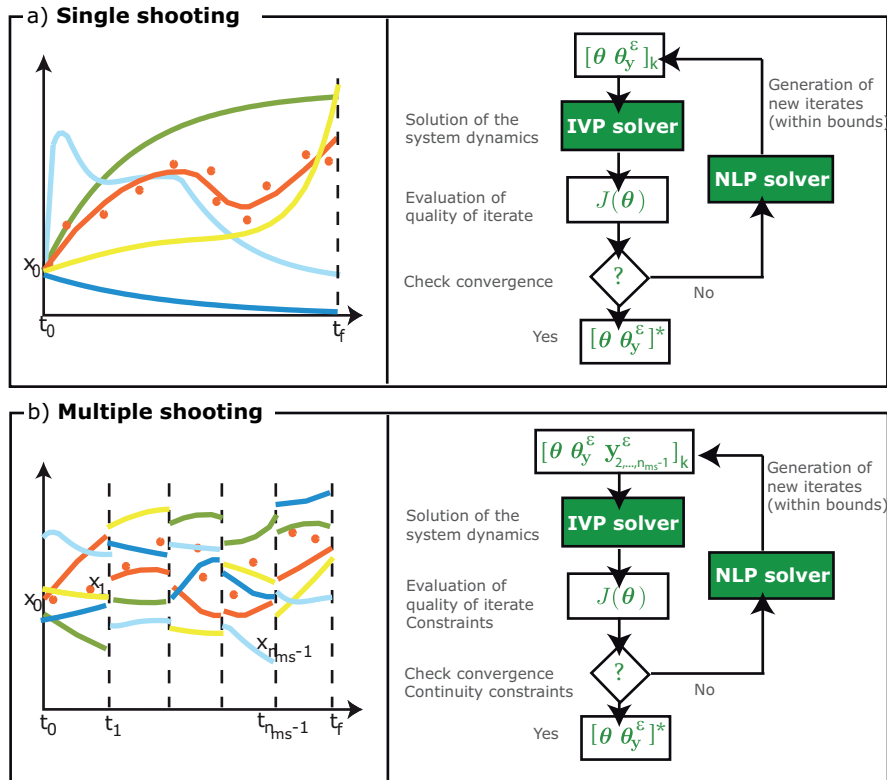


Figure 1.4: Single shooting versus multiple shooting approaches.

1.5 Practical identifiability analysis

As already mentioned before, practical identifiability analysis enables an evaluation of the possibility of assigning unique values to parameters from a given set of experimental data or experimental scheme subject to experimental noise. We distinguish between practical identifiability *a priori*, which anticipates the quality of the selected experimental scheme in terms of what we will call the expected uncertainty of the parameters, and practical identifiability *a posteriori*, which assesses the quality of the parameter estimates after model calibration in terms of the confidence region.

It is important to note that the major difference between the two analyses is that, *a priori*, we have to assume a maximum experimental error, whereas, *a posteriori*, since the experimental data are already available, the experimental error may be estimated either through experimental data manipulation

(when replicates of the experiments are available) or after model calibration using the residuals (i.e. the differences between model predictions and the experimental data) [46].

Possibly the simplest approach to perform such analyses given a set of simulated (*a priori*) or real (*a posteriori*) experimental data is to draw contours of the cost J_{lsq} or J_{llk} by pairs of parameters. This will help detect typical practical identifiability problems, such as strong correlation between parameters, the lack of identifiability for some parameters when the contours extend to infinity, or the presence of sub-optimal solutions.

A second possibility relies on the Cramm r-Rao inequality [28] which establishes a relationship between the so called Fisher Information Matrix ($\mathcal{F}$) and the covariance matrix ($\mathbf{C}$) for the case that the estimator is asymptotically unbiased:

$$\mathbf{C} \geq \mathcal{F}(\boldsymbol{\theta}^*) \quad (1.18)$$

being $\boldsymbol{\theta}^*$ a value for the parameters considered to be closed to optimum. The confidence interval of a given parameter $\boldsymbol{\theta}_i^*$ is then given by:

$$t_{\alpha/2}^\gamma \sqrt{\mathbf{C}_{ii}} \quad (1.19)$$

where $t_{\alpha/2}^\gamma$ is given by Student's t-distribution, γ corresponds to the number of degrees of freedom and α is the $(1-\alpha)$ 100% confidence interval selected by the user.

To robustly quantify the expected uncertainty of the parameters and/or the confidence region, we rely on a Monte Carlo-based sampling method [7, 22, 1]. The underlying idea is to simulate the possibility of performing hundreds of replicates of the same experimental scheme for a given experimental error. The model calibration problem is solved for each replicate and the cloud of solutions is recorded in a matrix. Note that, in order to avoid convergence to local solutions, an efficient global optimization method is required.

The cloud of solutions is assumed to correspond to, or to be fully contained in, a hyper-ellipsoid. Principal component analysis applied to the 0.95 – 0.05 interquartile range of the cloud or matrix of solutions then provides information on hyper-ellipsoid eccentricity (correlation between parameters) and pseudo-volume (accuracy of the parameters). The analysis of the histograms of the parameter solutions provides the mean value of the parameters ($\boldsymbol{\mu}$) and either maximum expected uncertainty (*a priori*) or the confidence intervals (*a posteriori*) for the parameters ($\mathbf{C}_\theta$). See details in [1].

The obtained expected uncertainty of the parameters will allow the different experimental designs to be compared *a priori*, i.e. without performing any experiment. The richest experiment, in terms of the quantity and quality of information, will be the one with the best compromise between pseudo-volume and eccentricity.

The confidence intervals obtained for the parameters will enable a decision to be made on the need to perform further experiments to improve the quality of the parameter estimates and, thus, the predictive capabilities of the model.

1.6 Optimal experimental design

A crucial aspect of experimental data is data quantity and quality. As mentioned in the previous section, a given set of data may result in practical identifiability problems. This is why data generation and modeling have to be implemented as parallel and interactive processes, thereby avoiding the generation of data that may eventually turn out to be unsuited for modeling.

In addition, the use of model-based (*in silico*) experimentation can greatly reduce the effort and cost of biological experiments, and simultaneously facilitate the understanding of complex biological systems [44, 23, 8, 24].

The aim of optimal experimental design is to calculate the best scheme of measurements in order to maximize the richness (quantity and quality) of the information provided by the experiments while minimizing, or at least, reducing, the experimental burden [7, 1].

The richness of the experimental information may be quantified by the use of the Fisher Information Matrix ($\mathcal{F}$) [46, 28], which can be defined as follows:

$$\mathcal{F} = E_{\mathbf{y}_m | \boldsymbol{\mu}} \left\{ \left[\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right] \left[\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right]^T \right\} \quad (1.20)$$

where E represents the expectation for a given value of the parameters $\boldsymbol{\mu}$ presumably close to the optimal solution $\boldsymbol{\theta}^*$.

It is important to remark here that the Fisher will, by its definition, depend on the type of experimental noise. The different formulations for the case of homoscedastic and heteroscedastic may be found in [18].

The optimal experimental design is then formulated and solved as a general dynamic optimization problem, see details in [1], that computes the time-varying stimuli profile, sampling times, experiments duration and (possibly) initial conditions so as to maximize a scalar measure of the Fisher Information Matrix subject to the system dynamics (Eqn. 1.1 and 1.2) and to other algebraic constraints associated with experimental limitations.

Regarding the selection of the scalar measure of the $\mathcal{F}$, several alternatives exist all of them related to the eigenvalues of the $\mathcal{F}$ and thus related to the shape and size of the associated hyper-ellipsoid. The most popular are probably the D-optimality and E-optimality criteria, the former corresponding to the maximization of the determinant of the $\mathcal{F}$ and the latter corresponding to the maximization of the minimum eigenvalue. From previous studies [1] it may be concluded that the E-optimality criterion offers the best quantity-quality compromise for the information, particularly for cases where the parameters are highly correlated or the sensitivities with respect to the parameters are highly uneven; otherwise D-optimality may be more successful.

1.6.1 Control vector parameterization

The control vector parameterization approach as described in [45] is extended here to the solution of the optimal experimental design problem. The CVP proceeds by dividing the duration of the experiment $[t_o, t_f^{iexp}]$ into a number ρ_{iexp} of intervals and approximating the stimuli (u_j^{iexp}) using low order (O_j^{iexp}) Lagrange polynomials within each interval (i^{iexp}).

$$u_{j^{iexp}}^{(i^{iexp})}(t) = \sum_{k=1}^{O_j^{iexp}} u_{ijk}^{iexp} \ell_k^{(O_j^{iexp})}(\tau^{(i^{iexp})}) \quad (1.21)$$

$$\text{with } t \in [t_{i-1}^{iexp}, t_i^{iexp}] \quad (1.22)$$

being τ the normalised time in the $i^{iexp}th$ element:

$$\tau^{(i^{iexp})} = \frac{t - t_{i^{iexp}-1}}{t_{i^{iexp}} - t_{(i^{iexp}-1)}} \quad (1.23)$$

and the Lagrange polynomials of order O_j^{iexp} :

$$\ell_k^{(O_j^{iexp})} = 1, \quad O_j^{iexp} = 1 \quad (1.24)$$

$$\ell_k^{(O_j^{iexp})} = \prod_{\substack{k'=1 \\ k' \neq k}}^{O_j^{iexp}} \frac{\tau - \tau_{k'}}{\tau_k - \tau_{k'}}, \quad O_j^{iexp} \geq 2 \quad (1.25)$$

with $i^{iexp} = 1, \dots, \rho_{iexp}$, $j^{iexp} = 1, 2$ y $k = 1, \dots, O_j^{iexp}$. Remark that the subindex *iexp* stands for each of the experiments being simultaneously designed. To allow for maximum flexibility, the stimuli may vary and/or may be approximated differently for the various experiments. It is important to mention that the selection of the parameterization will be constrained by the experimental possibilities. For example, in the context of cell signalling, the quantitative immunoblotting techniques may allow the use of step-wise profiles.

As a result, a NLP is obtained where the vector of decision variables $\mathbf{w} \in \mathbb{R}^{n_w}$ includes the coefficients in the polynomials u_{ijk}^{iexp} , the switching points t_i^{iexp} , the sampling times, the duration and the initial conditions for each experiment.

1.7 Numerical methods

1.7.1 Initial value problem solvers

It is out of the scope of this brief introduction to parametric identification to describe in detail the different types of initial value problem solvers. Basically available numerical solvers are based on the discretization of the experiment duration into a sufficiently large number of elements (mesh) and the approximation of the states by using local interpolation.

How the mesh is selected, the number and the location of the points in the mesh, the possibility of mesh adaptation, etc. result in a large variety of methods for both non stiff and stiff systems.

Possibly the most popular are the [Runge-Kutta](#) in its explicit and implicit versions, the [Adams-Bashforth](#) and the [BDF](#) (backward differentiation formula) based methods. Visit, for example, [39] for an extensive review of methods.

The evaluation of the rank of parameters and Fisher Information Matrix requires, in addition, the computation of the observables parametric sensitivities. In this regard, several alternatives exist, for example : the use of a finite differences scheme together with a suitable IVP solver; the analytic derivation of the parametric sensitivities which may be simultaneously solved with a suitable IVP solver, by exploiting the fact that original system and the parametric sensitivities share the Jacobian or the use of BDF methods which allow the efficient computation of sensitivities.

It should be remarked that all steps in the identification procedure require the solution of either the IVP or the IVP together with the parametric sensitivities a number of times, being this the most computationally demanding task. Therefore the selection of an adequate method will be crucial to the overall computational cost.

1.7.2 Nonlinear programming solvers

Optimization methods are designed to generate, from one or several initial guesses, a sequence of solutions that eventually converges to the minimum of the cost function. The way this sequence is generated gives rise to hundreds of different nonlinear programming (NLP) solvers.

A first classification of the methods would be in those able to handle nonlinear convex problems, [local methods](#) and those able to handle nonlinear non-convex or multimodal problems, [global methods](#).

Local methods

Local methods use information about the cost function and possibly its gradient and its Hessian in the neighborhood of every iterate thus these methods are expected to converge to the closest minimum. Figure 1.5 presents a classification of local methods including some well known examples, for detailed descriptions of the methods the reader is referred to, for example, the books by [15] or [34].

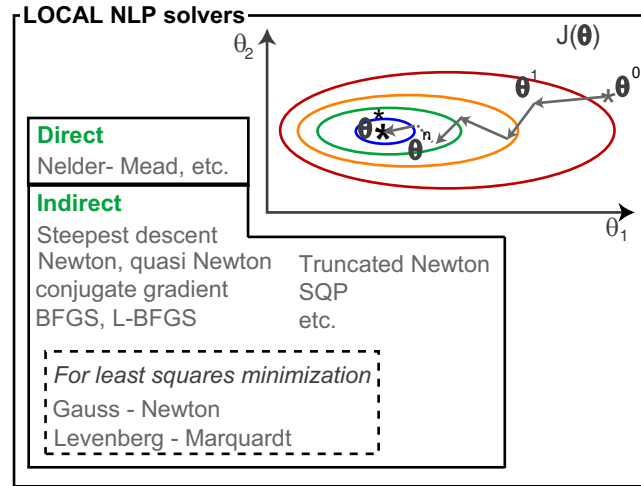


Figure 1.5: Illustrative representation of the numerical solution of a two dimensional convex optimization problem and classification of nonlinear local optimization methods.

The direct-search methods make use of the value of the cost function in several points in the vicinity of the current iterate to generate new iterates. The major disadvantage of direct methods is their slow convergence.

Alternatively indirect methods make use of gradient or gradient and Hessian information to increase convergence speed. In the context of least squares minimization the most widely used method is the Levenberg-Marquardt, a combination of the steepest descent with the Newton method for a least squares cost function. It should be noted that most of the methods for least squares problems, such as the Levenberg-Marquardt, are based on the Gauss-Newton modification of the Newton method, i.e. part of the Hessian of the objective with respect to the parameters is ignored so as to avoid computing second order derivatives [43], if this approximation is not valid the method may converge slowly or even fail. [42] describes how to combine the Gauss-Newton method with a sequential quadratic approach (SQP) for the specific case of minimizing the least squares function.

The two major advantages of the indirect local methods are:

- Convergence to a minimum is guaranteed by the fact that the gradient of the cost function evaluated at the optimum is zero and the Hessian is positive definite.
- The methods are highly efficient when started close to the solution.

Local methods have been largely used in combination with the single and the multiple shooting approaches for the purpose of parameter estimation. However the nonlinear character of the biological dynamic models leads to the presence of several suboptimal solution and thus local methods may end up in a suboptimal solution.

It has been argued that multiple shooting based approaches can circumvent some local minima by allowing for discontinuous trajectories while searching the global minimum. And even though this may be true for some cases, for example oscillatory systems, convergence to the global solution can not be guaranteed [6].

Moreover, in the presence of a bad fit, there is no way of knowing if it is due to a wrong model formulation, or if it is simply a consequence of local convergence.

Global methods

Global methods have emerged as the alternative to search the global optimum. One of the simplest global methods is a **Multistart method**. Here, a large amount of initial guesses are drawn from a distribution and subjected to a parameter estimation algorithm based on a local optimization approach. The smallest minimum is then regarded as being the global optimum. In practice, however, there is no guarantee of arriving to the global solution and the computational effort can be quite large. These difficulties are arising because it is a-priori not clear how many random initial guesses are necessary.

Over the last decade more suitable techniques for the solution of multi-modal optimization problems have been developed (see, e.g., [35], for a review). The successful methodologies combine effective mechanisms of exploration of the search space and exploitation of the previous knowledge obtained by the search. Depending on how the search is performed and the information are they exploiting the alternatives may be classified in three major groups: deterministic, stochastic and hybrid.

Global deterministic methods in general take advantage of the problem's structure and even guarantee convergence to the global minimum for some particular problems that verify specific conditions of smoothness and differentiability. Reviews of these methods can be found in [37] or [16].

Several recent works propose the application of global deterministic methods for model calibration in the context of chemical processes, biochemical processes, metabolic pathways, and signaling pathways [13, 19, 26, 38]. Although very promising and powerful, there are still limitations to their application, mainly due to rapid increase of computational cost with the size of the considered system and the number of its parameters.

As opposed to deterministic approaches, **global stochastic methods** do not require any assumptions about the problem's structure. Stochastic global optimization algorithms are making use of pseudorandom sequences to determine search directions toward the global optimum. This leads to an increasing probability of finding the global optimum during the runtime of the algorithm. The main advantage of these methods is that they rapidly arrive to the proximity of the solution. The number of stochastic methods has rapidly increased in last decades. The most successful approaches lie in one (or more) of the following groups: pure random search and adaptive sequential methods, clustering methods, population based methods or nature inspired methods [11]. Figure 1.6 presents a classification of the most widely used ones.

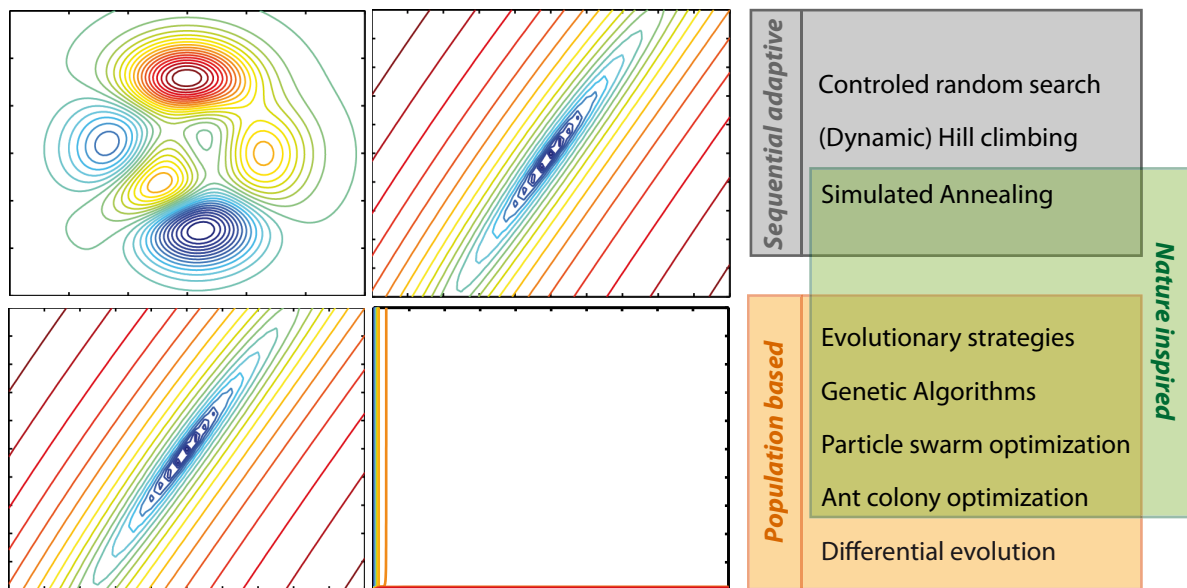


Figure 1.6: Illustrative examples of two dimensional multimodal problems and a possible classification of nonlinear global stochastic optimization methods.

Some of these strategies have been successfully applied to parameter estimation problems in the context of systems biology, see [32] for the application of simulated annealing; [33] and [40] for the application of evolutionary search algorithms or [30] for genetic programming.

Despite the fact that many stochastic methods can locate the vicinity of global solutions very rapidly, the computational cost associated to the refinement of the solution is usually very large. In order to surmount this difficulty, hybrid methods and metaheuristics have been recently presented for the solution of parameter estimation problems [41, 40, 6] that speed up these methodologies while retaining their robustness.

Similarly in the context of optimal experimental design it has been shown that the use of global solvers such as SSm main prevent the convergence to suboptimal solutions [1].

2

AMIGO toolbox description

2.1 Toolbox download and License

AMIGO toolbox and the corresponding documentation is available at:

<http://www.iim.csic.es/~amigo>

The toolbox is free of charge for academic purposes under the creative commons license. For further details on license conditions please visit:

<http://creativecommons.org/licenses/by-nc-nd/3.0/>

2.2 Toolbox requirements and installation guide

As mentioned before the most computationally demanding step in all tasks in AMIGO is the solution of the IVP (system dynamics). Therefore efficiency in simulation is of the highest importance particularly for large scale models or optimizations and global analyses when the number of simulations required is large. In this regard AMIGO offers two different usage modes:

Basic: the model will be automatically generated in MATLAB or provided by the user in MATLAB or SBML. In this case an IVP from the ones available in MATLAB will be used.

Enhanced: the model will be automatically generated in FORTRAN or provided by the user as FORTRAN code and this will be automatically mexed to one of the FORTRAN IVP solvers available in AMIGO. Note that this mexing will be performed automatically during model preprocessing and will be completely transparent to the user.

Requirements for both type of usages are:

Operating system There is an unique AMIGO version for both Windows and Linux.  	
BASIC MODE MATLAB version Matlab 6.5- or higher. AVAILABLE for 32 and 64 bits	ENHANCED MODE MATLAB & FORTRAN version Matlab 6.5- 7.1 (exclusive) requires a MATLAB compatible FORTRAN compiler Compaq Visual Studio for windows. From MATLAB 7.1, g95 will be used. It is automatically incorporated for windows users. Linux users require to install g95. AVAILABLE for 32 bits
MATLAB Toolboxes Matlab Optimization Toolbox (for using MATLAB local NLP solvers such as fmincon or fminsearch) SBML and libSBML toolboxes are required to handle SBML models.	

Table 2.1: Summary of requirements for AMIGO.

To install the toolbox:

1. Unzip the .zip archive in your computer
2. Start a Matlab session and go to the AMIGO folder
3. Type:

```
> AMIGO_Startup
```

every time you want to use AMIGO toolbox

2.3 General structure

AMIGO is organized in three main modules: the pre-processor, the numerical kernel and the post-processor (Figures 2.1 and 2.2). Given a problem definition, `AMIGO_Prep` pre-processes user input data, generates directories and necessary code. The different numerical modules (`AMIGO_SModel`, `AMIGO_Sobs`, `AMIGO_SData`, `AMIGO_LRank`, `AMIGO_GRank`, `AMIGO_ContourP`, `AMIGO_RIdent`, `AMIGO_PE` and `AMIGO_OED`) are then called by the user to perform the desired task(s).

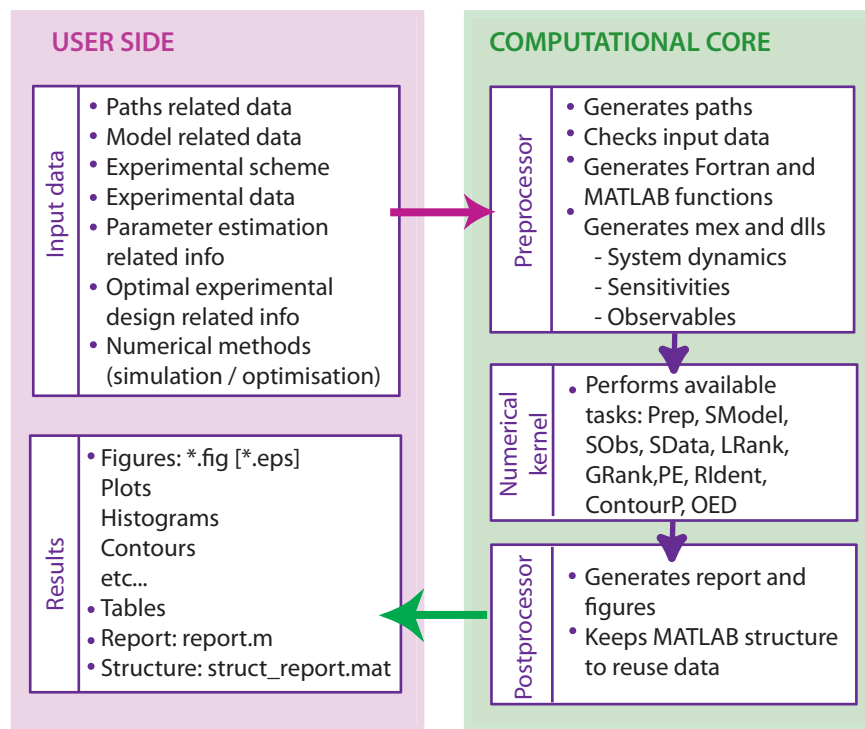


Figure 2.1: Toolbox general structure

This general structure correlates to the following folder and code organisation:

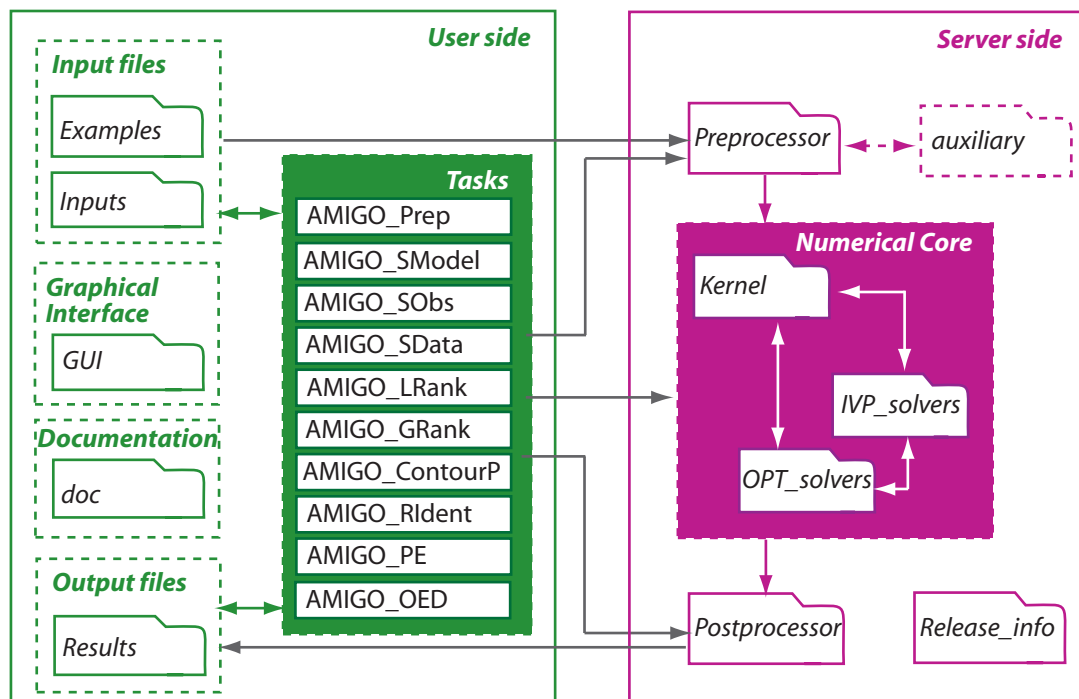


Figure 2.2: Folder and code organisation

doc folder keeps all toolbox related documentation.

Examples folder keeps a number of implemented examples that user may consider as templates to implement new problems.

Inputs folder, originally empty, is devoted to keep new inputs created by users.

Kernel folder, keeps all numerical functions, NLP solvers, IVP solvers and auxiliary code (FORTRAN compilation required files).

Postprocessor folder, keeps all matlab functions to generate reports, structures and figures.

Preprocessor folder, keeps all matlab functions to generate matlab or fortran code, to mex files when required and to generate necessary paths. This folder keeps also the defaults for all inputs, user may modify public defaults in: `AMIGO_public_defaults.m`

Release_info folder contains the `AMIGO_release_info.m` with all details about previous and current releases.

Results folder, originally empty, is devoted, by default, to keep all results. User may create other results folders.

2.4 Summary of features

AMIGO has been designed to offer maximum flexibility to the user, not only in the number of tasks that may accomplish within the parametric identification loop but also in the types of models and experimental schemes that may be considered, and in the availability of a large variety of numerical IVP and NLP solvers, enabling the solution of a broad range of problems:

Model types: AMIGO supports general nonlinear dynamic models using a simple syntax, FORTRAN or MATLAB. Allows to import sbml and black-box user defined models.

Experimental scheme: It allows for flexible experimental schemes –one or more experiments, input profiles, initial conditions, experiment durations, and sampling times – that are to be performed in silico.

Experimental data: Allows to introduce or load real experimental data with different types of experimental noise, homoscedastic or heteroscedastic. In addition it is possible to generate pseudo-experimental data for a given experimental scheme.

Parameter estimation: Allows multi-experiment fitting with local (experiment dependent) and global unknowns (parameters and initial conditions). Several types of cost functions, weighted least squares or log-likelihood, may be used depending on the available information about the experimental noise.

Practical identifiability analysis: Computes local and global sensitivities, the correlation matrix from the Fisher information matrix depending on the experimental noise conditions, cost function contour plots by pairs of unknowns and the robust Monte-Carlo based approach.

Optimal experimental design: Solves the D-, E-, Modified E or A- optimal experimental design problem as a general open loop optimal control problem allowing for sequential and parallel designs. It is possible to optimize sampling times, input conditions, experiment durations and initial conditions

for one or more simultaneous experiments. Several Fisher matrix formulations are available depending on the experimental noise.

Numerical methods: It incorporates several state of the art initial value problem (IVP) and non-linear optimization (NLP) methods to deal with both the parameter estimation and the experimental design problems. Regarding IVP solvers, explicit and implicit Runge-Kutta, Adams and BDF methods have been incorporated together with methods to compute sensitivities. Concerning NLP solvers, several direct and indirect local, multistart of local methods, global stochastic and sequential and parallel hybrid optimization methods are available. Computational demanding tasks are automatically interfaced to FORTRAN compiled code in the enhanced mode.

Reporting: Generates reports and plots according to user specifications for the different tasks. The complete working session is saved in a Matlab structure and may be reloaded any time.

Following tables summarise the current features:

Models	
> Deterministic Dynamic models	Any non-linear general form: ODEs and DAES with constant mass matrix can be handled directly; more general DAES, DDEs and PDEs may be handled through black-box models.
> Format	May be provided in MATLAB, FORTRAN, SBML, strings to generate a MATLAB or FORTRAN model and black-box models
> Observation functions	Any linear / non-linear function of the states
> Notation	Customized names for states, parameters, stimuli & observables are allowed. IMPORTANT: n, t, u, y, ydot, par, tlast, told, pend and v (in their lower and upper case versions) are reserved words

Figure 2.3: Summary of features: types of models allowed in AMIGO.

Experimental data	
> Definition of experimental scheme	Flexible number of experiments, observables, initial conditions, sampling times, type of noise, stimuli. Flexibility over experiments.
> Pseudo-experimental data	"Numerical" data under experimental scheme conditions.
> Experimental data	Experimental time-series data plus error bars (if available)
> Stimuli	Theoretical or measured stimuli (if available).

Figure 2.4: Summary of features: types of experimental schemes and data allowed in AMIGO.

IVP Solvers			
> Non-Stiff and mildly stiff	RKF45	FORTTRAN	Runge-kutta-fehlberg (4,5) method. E. Fehlberg , Low-order classical Runge-Kutta formulas with stepsize control , NASA tr r-315
	ode45	MATLAB	Runge-kutta-fehlberg (4,5) method
	ode113	MATLAB	Adams-Bashforth-Moulton (1,12). The MATLAB ODE Suite, L. F. Shampine & M. W. Reichelt, SIAM Journal on Scientific Computing, 18-1, (1997)
> Stiff	Radau5	FORTTRAN	Implicit Runge-Kutta Method. E. Hairer & G. Wanner, Solving ordinary differential equations II. Stiff and Differential-algebraic problems. Springer Series in Computational Mathematics 14, Springer-Verlag, 1996.
	LSODA	FORTTRAN	ADAMS with authomatic switch to BDF. A. C. Hindmarsh, ODEPACK, A systematized collection of ODE solvers, Scientific Computing, R. S. Stepleman et al. (eds.), Amsterdam, pp. 55-64 (1983) L.R. petzold, Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations, SIAM J. Sci. Stat. Comput. 4: 136-148.(1983)
	ode15s	MATLAB	Klopfenstein-Shampine BDF. The MATLAB ODE Suite, L. F. Shampine & M. W. Reichelt, SIAM Journal on Scientific Computing, 18-1, (1997)
> Sparse, Stiff	LSODES	FORTTRAN	ADAMS/BDF. A. C. Hindmarsh, ODEPACK, A systematized collection of ODE solvers, Scientific Computing, R. S. Stepleman et al. (eds.), Amsterdam, pp. 55-64 (1983); S.C. Eisenstat et al. Yale Sparse Matrix package. I & II. Int. J. Num. Meth. Eng., 18 (1982)
> To compute sensitivities	ODESSA	FORTTRAN	BDF: Leis JR, Kramer MA. Sensitivity Analysis of Systems of Differential and Algebraic Equations. Comp & Chem Eng 1985, 9(3):93-96.
	SENSMAT	MATLAB	Modification of the code by V.M. García Mollá & R. Gómez Padilla to compute parametric sensitivities (2002). www.mathworks.com/matlabcentral/fileexchange/1480-sensitivity-analysis-for-odes-and-daes
	Finite Differences		

Figure 2.5: Summary of features: IVP solvers available in AMIGO.

NLP Solvers		
> Local methods	Direct methods	
	NOMAD	Pattern search method. M. A. Abramson. Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems. PhD , Rice University, 2002.
	DHC	Dynamic hill climbing method. de la Maza & D. Yuret. Dynamic hill climbing. AI Expert, 9(3):26-31, 1994.
	fminsearch	Nelder-Mead. J.C. Lagarias, J.A. Reeds, M. Wright, P.E. Wright, Convergence properties of the Nelder-Mead Simplex Method in Low Dimensions, SIAM J Opt, 9(1):.112-147, 1998.
	Indirect methods	
	fmincon	SQP (Sequential Quadratic Programming), MATLAB optimization Toolbox
	solnp	Interior point SQP. Y. Ye. Interior algorithms for linear, quadratic and linearly constrained non-linear programming. PhD , Stanford University, 1987.
	ipopt	Large scale interior point. A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math Prog., 06(1):25-57,2006.
	misqp	Trust region SQP. O. Exler and K. Schittkowski. A trust region SQP algorithm for mixed-integer nonlinear programming. Opt. Lett., 1(3):269-280, 2007.
	n2fb*	Least-squares method. J.E. Dennis, D. M. Gay, and R. E. Welsch. An adaptive non-linear least-squares algorithm. ACM Trans Math Soft, 7(3):348-368, 1981.
> Multistart		
N starts of any of the available local solvers: to analyse multimodality		
> Global stochastic	DE	Population based differential evolution. Storn R, Price K. Differential Evolution – a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. J Global Optim, 11:341-359, (1997)
	SRES	Evolutionary search method. Runarsson T, Yao X. Stochastic ranking for constrained evolutionary optimization. IEEE Trans Evol Comp, 564:284-294, (2000)
> Sequential Hybrid methods		
All possible combinations of Global stochastic methods with the above mentioned local solvers		
Balsa-Canto E, Vassiliadis V, Banga J: Dynamic Optimization of Single- and Multi-Stage Systems Using a Hybrid Stochastic-Deterministic Method. Ind Eng Chem Res, 44(5):1514-1523, 2005.		
Rodriguez-Fernandez M, Mendes P, Banga JR. A hybrid approach for efficient and robust parameter estimation in biochemical pathways. Biosyst, 83:248-265, (2006)		
Balsa-Canto, E., Peifer, M., Banga, J., Timmer, J., and Fleck, C. Hybrid optimization method with general switching strategy for parameter estimation. BMC Systems Biology, 2:26, 2008.		
> Metaheuristics	SSm; fSSm; eSS*	Different Scatter Search based approaches. Egea JA, Rodriguez-Fernandez M, Banga JR, Martí R. Scatter Search for Chemical and Bio-Process Optimization. J Glob Opt, 37(3):481-503, (2007)
	GLOBALm	Clustering method. Csendes, T., L. Pal, J.O.H. Sendin, J.R. Banga. The GLOBAL Optimization Method Revisited. Optimization Letters, 2(4):445-454, 2008.

* Only for parameter estimation

Figure 2.6: Summary of features: NLP solvers available in AMIGO.

2.5 How to input problems in AMIGO

AMIGO is programmed making use of the so called Matlab structures. Structures are multidimensional Matlab arrays with elements called fields. These fields may be of any data type (arrays, matrices, strings of characters, etc.) and may be easily classified in subsets, therefore being quite comfortable for managing all input and output information. Inputs and results will be kept in two structures: **inputs** and **results** organised as follows:

inputs.	[Structure that keeps all inputs related information: model, experiments, parameters, experimental design]
Fields	[Brief description]
.model.	[Keeps all model related information: model type, equations, states, stimuli (inputs), parameters...]
.exps.	[Keeps the experimental scheme: number of exps, observation function, input conditions, sampling times, experimental data and error, ...]
.ivpsol.	[Keeps all IVP solver related info: solver, tolerances...]
.rank.	[Keeps global rank info: number of samples]
.nlpsol.	[Keeps all NLP solver related info: method, starts for multistart]
.PEsol.	[Keeps all parameter estimation related info: unknowns to be estimated (local & global), bounds, cost function]
.rid.	[Keeps number of trials for robust identifiability]
.OEDsol.	[Keeps all OED related info: exps to be designed, conditions to be designed and bounds, cost function,...]

results.	[Structure that keeps all results related information: paths, plots and reports files, and task related results]
Fields	[Brief description]
.pathd.	[Keeps all paths related information]
.plotd.	[Keeps all plots related information]
.sim.	[Keeps all results related to model simulation]
.lrnk.	[Keeps all results related to local rank: sensitivities, rank measures per experiment, overall rank, ...]
.grnk.	[Keeps all results related to global rank: rank measures per experiment, overall rank, ...]
.fit.	[Keeps all results related to the fit to experimental data: residuals, best unknowns, confidence...]
.nlpsol.	[Keeps all results related to the optimization: best solution, solver statistics...]
.rid.	[Keeps all results related to the robust identifiability analysis: cloud of solutions, eccentricity, pseudo-volume, confidence regions...]
.oed.	[Keeps all results related to the optimal experimental design: best experiments, best alphabetic criterion...]

Figure 2.7: Overview of inputs and results structures.

As explained in the Brief Theoretical Introduction, the application of the iterative identification procedure requires the definition of several elements: the model, the experimental scheme and data, the model unknowns, ranges for the unknowns, parameter estimation cost function, optimal experimental design cost function, ranges for the experimental scheme and numerical solvers.

The user may introduce the necessary inputs either through an input file or through the Graphical User Interface which will generate the corresponding input file. Several input file templates have been incorporated in the folder **Examples** in the toolbox.

The input files (and the Graphical User Interface) are organised attending to the information required for each task in AMIGO. It should be noted however that any task, but **AMIGO_OED**, may be performed when all inputs for parameter estimation are introduced.

Following figure summarizes the requirements for all tasks:

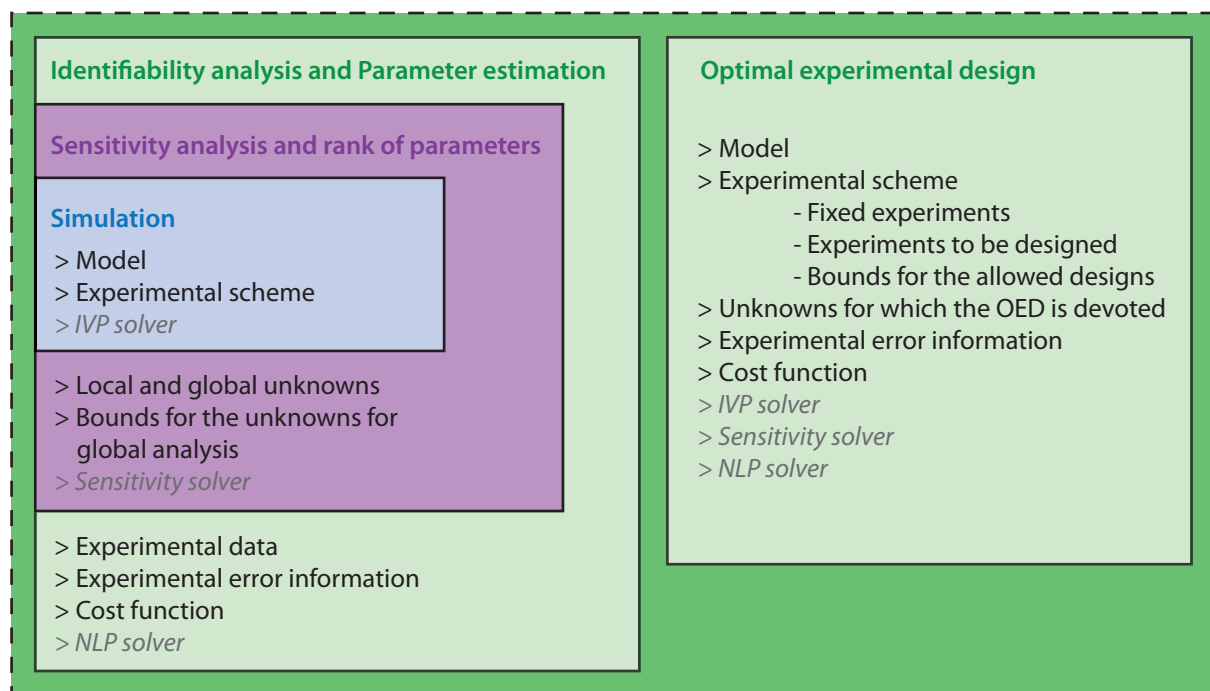


Figure 2.8: Overview of inputs for the different tasks. Note that the information is nested. All tasks but optimal experimental design may be performed when all inputs for parameter estimation are introduced.

2.5.1 Defining the model

All inputs related to model definition will be kept in the structure: `inputs.model`, whose fields are:

<code>inputs.model.input_model_type</code>	Defines how the model will be introduced. > 'charmodelF' or 'charmodelM': The user must input the model as strings and the toolbox will automatically generate FORTRAN or MATLAB code respectively. > 'fortranmodel' or 'matlabmodel': The user must provide FORTRAN or MATLAB files including the system dynamics. > 'sbmlmodel': the user will provide a .xml file and the toolbox will translate it into MATLAB. > 'blackboxmodel': the user will provide a MATLAB file which solves the system dynamics. This possibility is specially comfortable to handle PDEs, DDEs or complex DAEs or to call a MATLAB external package or software tool. > 'blackboxcost': the user must provide a MATLAB file which computes the cost function to be minimized (parameter estimation)
<code>inputs.model.n_st</code>	Number of states in the model.
<code>inputs.model.n_par</code>	Total number of parameters in the model. This includes all the constant parameters in the model, even if they are not to be estimated.
<code>inputs.model.n_stimulus</code>	Total number of inputs, controls or stimuli.

<code>inputs.model.names_type</code>	Defines how the states, parameters and stimuli will be introduced. > 'standard' (x1,x2,p1,p2...,u1, u2,...) > 'custom'(default)
--------------------------------------	---

Necessary inputs for custom names	
<code>inputs.model.st_names</code>	Names for states: char('stname1','stname2',....)
<code>inputs.model.par_names</code>	Names for parameters: char('parname1','parname2',....)
<code>inputs.model.stimulus_names</code>	Names for stimuli: char('stimulusname1','stimulusname2',....)
IMPORTANT: User may select any customised name but: n, t, u, y, ydot, par, tlast, told, pend and v which are reserved words	

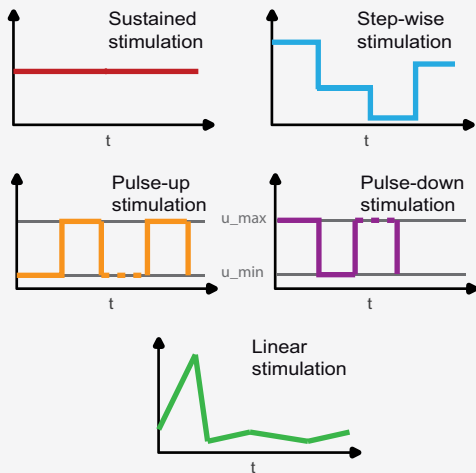
2.5.2 Defining the experimental scheme

The experimental scheme will be fixed to perform simulations, sensitivity analysis, rank of parameters, identifiability analysis and parameter estimation, whereas for the case of optimal experimental design some experiments may be fixed and some experiments or particular experimental conditions are to be designed.

All inputs related to the experimental scheme definition will be kept in the structure: `inputs.exps`, even if they are to be designed. Note that those which become decision variables in the OED problem will be saved by the toolbox in the structure, `inputs.OEDsol`.

Following tables describe the fields of the structure `inputs.exps` that correspond to the experimental scheme.

<code>inputs.exps.n_exp</code>	Number of experiments to be considered
<code>inputs.exps.obs{iexp}</code>	Observation function for the experiment iexp > 'states': when all states in the model are observed > char('obsname1= function of states 1','obsname2=function of states 2',....) when an observation function is to be defined IMPORTANT: Experiment dependent inputs should defined for every experiment: <code>iexp=1,...,inputs.exps.n_exp</code>
<code>inputs.exps.obs_names{iexp}</code>	Names given to the observables (when an observation function is defined)
<code>inputs.exps.exp_y0{iexp}</code>	Initial conditions for the experiment iexp Note that initial conditions may be estimated from the experimental data or may be designed in OED. In these cases, nominal values for the initial conditions should be introduced here. (These will be updated during the optimization)
<code>inputs.exps.t_f{iexp}</code>	Experiment duration for experiment iexp

<code>inputs.exps.n_s{iexp}</code>	Number of sampling times for experiment <code>iexp</code>
<code>inputs.exps.t_s{iexp}</code>	[OPTIONAL input] Sampling times. Default: equidistant sampling times within the interval: <code>[inputs_def.exps.ts_0{iexp} inputs.exps.t_f{iexp}]</code>. <code>inputs_def.exps.ts_0{iexp}</code> is also an OPTIONAL input, by default its value is 0.
<code>inputs.exps.u_interp{iexp}</code>	[OPTIONAL] Stimuli, input or control interpolation for <code>iexp</code> Several possibilities are available: > 'sustained': constant input > 'step': step-wise input profile > 'pulse-up': pulse-wise input profile > 'pulse-down': pulse-wise input profile > 'linear'(default): linear interpolation, particularly useful for measured inputs or to implement any type of input profile Illustrative examples: 

Necessary inputs for sustained stimulation	
<code>inputs.exps.u{iexp}</code>	Column vector of control values, with as many rows as controls. Ex.: <code>[u_1; u_2; ...]</code>
<code>inputs.exps.t_con{iexp}</code>	Row vector of initial and final times for stimulation

2.5.3 Defining the experimental data and the corresponding error information

In addition to the experimental scheme, the structure `inputs.exps` also collects the experimental data related information which is necessary for the identifiability analysis and parameter estimation. The toolbox offers the possibility of generating pseudo-experimental data by means of simulation. This possibility may be useful for numerical tests. Inputs will be, in general, different for the case when using pseudo- and real data.

Defining pseudo-experimental data

<code>inputs.exps.data_type</code>	Indicates type of data > 'pseudo': to generate simulated experimental data > 'pseudo_pos': to generate positive definite pseudo-data
<code>inputs.exps.noise_type</code>	Indicates type of experimental noise There are the following possibilities: > 'homo': homoscedastic noise with known constant variance > 'homo_var': homocedastic noise with known varying variance > 'hetero': heteroscedastic noise, standard deviation proportional to the observable is assumed
<code>inputs.exps.std_dev</code>	Indicates the standard deviation value in tant per one to be used to generate pseudo-experimental data

Defining real experimental data

<code>inputs.exps.data_type</code>	Indicates type of data. > 'real': real data introduced through matrices
<code>inputs.exps.exp_data{iexp}</code>	Matrix of real data with as many rows as sampling times and as many columns as observables Note that data may be read from .m, .mat, .txt, .xls files
<code>inputs.exps.noise_type</code>	Indicates type of experimental noise There are the following possibilities: > 'homo': homoscedastic noise with known constant variance > 'homo_var': homocedastic noise with known varying variance > 'hetero': heteroscedastic noise, standard deviation proportional to the observable is assumed
<code>inputs.exps.std_dev</code>	[OPTIONAL] Indicates the standard deviation value in tant per one to be used as a measure of the experimental error when no error bars are available
<code>inputs.exps.error_data{iexp}</code>	[OPTIONAL] Matrix of real experimental error data (standard deviation) with as many rows as sampling times and as many columns as observables

With the above mentioned information the user may already perform `AMIGO_SData` to either generate pseudo-experimental data or to plot model predictions vs real experimental data with error bars when available. Simulation will be performed with the initial value problem solver by default.

2.5.4 Inputs for LRank, GRank, ContourP, RIdent and PE

To deal with these tasks the user must define which are the model unknowns to be taken into account. Note that selected unknowns may be global, i.e. with the same value for all experiments, or local, i.e. with experiment dependent values. In addition, for the identifiability analysis and parameter estimation the cost function should be specified. All these inputs are kept in the structure `inputs.PEsol*` defined below.

Defining global unknown parameters

<code>inputs.PEsol.id_global_theta</code>	Indicates which are the unknown parameters to be considered for LRank, GRank, ContourP, PE or RIdent. There are two possibilities > 'all': when all parameters are unknown > char('parname1','parname7',...): when only a subset of the model parameters are unknown
<code>inputs.PEsol.global_theta_max</code>	Row vector of maximum values allowed for the unknown parameters. Only necessary for GRank, ContourP, RIdent and PE
<code>inputs.PEsol.global_theta_min</code>	Row vector of minimum values allowed for the unknown parameters. Only necessary for GRank, ContourP, RIdent and PE
<code>inputs.PEsol.global_theta_guess</code>	[OPTIONAL] Row vector of initial guess values for the unknown parameters. By default the mean value in the range will be considered. IMPORTANT remarks: > Once <code>global_theta_max</code>, <code>global_theta_min</code> and <code>global_theta_guess</code> are defined, known parameters will take the nominal value defined in <code>inputs.model.par</code> and unknown parameters will take the value in <code>global_theta_guess</code> (either user defined or default mean within the range). - Note that these will be the values considered for SModel, SObs, SData, LRank, ContourP and RIdent and the initial guess for PE. > Update <code>global_theta_guess</code> to simulate, rank or to perform identifiability analysis around a given (optimal) value.

Defining global unknown initial conditions

<code>inputs.PEsol.id_global_theta_y0</code>	Indicates which are the unknown initial conditions to be considered for LRank, GRank, ContourP, PE or RIdent. > 'none' (default) > 'all': when all initial conditions are unknown > char('stname1','stname5',...): when only a subset of the model initial conditions are unknown
<code>inputs.PEsol.global_theta_y0_max</code>	Row vector of maximum values allowed for the unknown initial conditions Only necessary for GRank, ContourP, RIdent and PE
<code>inputs.PEsol.global_theta_y0_min</code>	Row vector of minimum values allowed for the unknown initial conditions Only necessary for GRank, ContourP, RIdent and PE
<code>inputs.PEsol.global_theta_y0_guess</code>	[OPTIONAL] Row vector of initial guess values for the unknown initial conditions Default: mean value IMPORTANT remarks: > Known initial conditions will take the nominal value defined in <code>inputs.exps.exp_y0</code> and unknown initial conditions will take the value in <code>global_theta_y0_guess</code>. > Update <code>global_theta_y0_guess</code> to simulate, rank or to perform identifiability analysis around a given (optimal) value.

Defining experiment dependent (local) unknowns

<code>inputs.PEsol.id_local_theta{iexp}; inputs.PEsol.id_local_theta_y0{iexp}</code>	Indicate which are the unknown experiment dependent parameters and initial conditions to be considered for LRank, GRank, ContourP, PE or RIdent > 'all': when all parameters are unknown > 'none' (default): no local unknowns are considered > char('parname1','parname7',...); char('stname1','stname7',...)
<code>inputs.PEsol.local_theta_max{iexp}; inputs.PEsol.local_theta_y0_max{iexp}</code>	Row vector of maximum values allowed for the unknown local parameters and initial conditions for each experiment
<code>inputs.PEsol.local_theta_min{iexp}; inputs.PEsol.local_theta_y0_min{iexp}</code>	Row vector of minimum values allowed for the unknown local parameters and initial conditions for each experiment
<code>inputs.PEsol.local_theta_guess{iexp}; inputs.PEsol.local_theta_y0_guess{iexp}</code>	[OPTIONAL] Row vector of initial guess values for the local unknown parameters and initial conditions for each experiment

Defining the cost function for parameter estimation (PE) and identifiability analysis (ContourP and RIdent)

<code>inputs.PEsol.PEcost_type</code>	Type of cost function depending on the available information about the experimental data error > 'lsq': Weighted Least Squares Function. For the cases where no information about the experimental error is available. > 'llk': Log-Likelihood function. For the cases where type of error and standard deviation are known.
---------------------------------------	---

Available options for Weighted Least Squares function

<code>inputs.exps.lsqr_type</code>	To indicate the type of weighting matrix to be used in the LSQ function > 'Q_I': No weighting. All data will be given the same importance. > 'Q_expmx': Normalizing with the maximum experimental data per observable per experiment. To take into account possible different orders of magnitude among the different observables.
------------------------------------	---

Available options for Log-Likelihood function

<code>inputs.exps.lkk_type</code>	To indicate the type of function depending on the type of experimental error > 'homo': for the case of homoscedastic noise. Similar to the LSQ function but the weighting matrix corresponds to the constant variance of the data. > 'homo_var': for the case of homoscedastic noise with known varying variance. Every data is weighted with its corresponding variance value. Data with less associated variance will be given more importance in the optimization. > 'hetero': for the case of heteroscedatic noise under the assumption that the standard deviation is proportional to the observation.
-----------------------------------	--

2.5.5 Inputs for OED

As stated before, to perform the tasks SModel, SData, SObs, LRank, GRank, ContourP, RIdent and PE, the experimental scheme will be fixed. For the case of optimal experimental design the user may select to have one or more experiments to be optimally designed. Several aspects may be designed for each experiment:

- Initial conditions for each experiment
- Experiment duration
- Number and location of sampling times
- Stimulation conditions

the user may select which of the above are to be designed within each experiment.

Regarding the overall experimental scheme, the user may select to design one or more experiments in a parallel experimental design or to take into consideration previous experiments in a sequential-parallel scheme:

- **Parallel experimental design:** Regards the design of one or more experiments in parallel. The user needs to introduce the model, the experimental scheme to be designed with the corresponding degrees of freedom and bounds according to experimental constraints and FIM based cost function and the numerical methods.
- **Sequential-parallel experimental design:** Regards the design of one or more experiments in parallel but considering previous experiments (that will remain fixed but will be considered to compute the FIM). In this case the user needs to introduce the model, the experimental scheme that includes experiments to be designed and fixed (already performed) experiments, degrees of freedom for those experiments to be designed and the corresponding bounds according to experimental constraints, FIM based cost function and the numerical methods.

Following tables summarise the specific inputs to run OED:

<code>inputs.exps.exp_type{iexp}</code>	Indicates whether the experiment should be fixed ('fixed') or optimally designed ('od')
IMPORTANT: The experiments that are to remain fixed will be defined as detailed above in the section "Defining the experimental scheme". Note that information about the experimental error is also necessary to define and compute the FIM.	
<code>inputs.exps.exp_y0_type{iexp}</code>	[Only for 'od' exps] Indicates whether the initial conditions should be fixed ('fixed') or optimally designed ('od')
<code>inputs.exps.tf_type{iexp}</code>	[Only for 'od' exps] Indicates if the experiment duration is to be 'fixed' or designed 'od'
<code>inputs.exps.ts_type{iexp}</code>	[Only for 'od' exps] Indicates whether the number and location of sampling times should be designed
<code>inputs.exps.u_type{iexp}</code>	[Only for 'od' exps] Indicates if the stimulation conditions should be designed

Inputs to design initial conditions

<code>inputs.OEDsol.id_y0{iexp}</code>	To indicate the which initial conditions should be designed > 'all' > char('stname2', 'stname8',...)
<code>inputs.OEDsol.y0_min{iexp}</code>	Minimum allowed value for the initial conditions
<code>inputs.OEDsol.y0_max{iexp}</code>	Maximum allowed value for the initial conditions
<code>inputs.OEDsol.y0_guess{iexp}</code>	Initial guess for the initial conditions

Inputs to design experiments duration

<code>inputs.OEDsol.tf_min{iexp}</code>	Minimum allowed value for the experiment duration
<code>inputs.OEDsol.tf_max{iexp}</code>	Maximum allowed value for the experiment duration
<code>inputs.OEDsol.tf_guess{iexp}</code>	Initial guess for the experiment duration

Inputs to design number and location of sampling times

<code>inputs.OEDsol.ts_min_dist{iexp}</code>	Minimum allowed distance between sampling times
--	---

Inputs to design stimulation conditions

<code>inputs.exps.u_interp{iexp}</code>	Interpolation selected for OED. The user may select which is the type of experiments that can be experimentally performed. Step-wise profiles may end up in designs that can be difficult to implement in the lab for some specific applications. In those cases 'sustained', 'pulse-up' or 'pulse-down' experiments should be selected.
---	---

Inputs to design sustained experiments

REMARK: Only the level of the stimuli will be optimized. Thus only minimum, maximum and initial guess for the optimization will be required.

<code>inputs.OEDsol.u_min{iexp}</code>	Column vectors n_stimuli x 1 of minimum, maximum
<code>inputs.OEDsol.u_max{iexp}</code>	and initial guess values for the stimuli
<code>inputs.OEDsol.u_guess{iexp}</code>	

Inputs to design pulse-wise experiments

REMARK: Only the duration and location of the pulses will be optimized. Levels and number of pulses are kept fixed. Thus only `inputs.exps.n_pulses{iexp}` and `inputs.exps.u_min{iexp}` and `inputs.exps.u_max{iexp}` are to be defined (see section "Defining the experimental scheme" above for more details)

Inputs to design step-wise experiments

REMARK: The level, duration and location of the steps is to be optimized. The number of steps will be kept fixed. Thus only `inputs.exps.n_steps{iexp}` and minimum, maximum and initial guess for the optimization will be required.

<code>inputs.exps.n_steps{iexp}</code>	Number of steps (not to be designed)
<code>inputs.OEDsol.u_min{iexp}</code> <code>inputs.OEDsol.u_max{iexp}</code> <code>inputs.OEDsol.u_guess{iexp}</code>	Matrices of <code>n_stimulus</code> x <code>n_steps{iexp}</code> of minimum, maximum and initial guess values for the stimuli

Necessary inputs to define Fisher Information Matrix ($\mathcal{F}$) and $\mathcal{F}$ based cost function

Note that, the experimental design is conceived to improve identifiability, thus the user should select which are the unknowns for which the experiment design is performed and provide a nominal value for such unknowns (obtained from parameter estimation, from the literature, etc.). The $\mathcal{F}$ will be computed only for these unknowns.

<code>inputs.PEsol.id_global_theta</code>	Parameters to be considered for OED > 'all': all parameters > char('parname1','parname7',...): only a subset of model parameters
<code>inputs.PEsol.global_theta_guess</code>	Row vector of nominal values for the parameters
<code>inputs.PEsol.id_global_theta_y0</code>	Initial conditions to be considered for OED > 'none' (default) > 'all' > char('stname1','stname5',...)
<code>inputs.PEsol.global_theta_y0_guess</code>	Row vector of nominal values for initial conditions

In addition the $\mathcal{F}$ will be dependent on the cost function used for parameter estimation, therefore type of function used for PE should be also indicated together with the experimental standard deviations as detailed above in section "Defining the cost function for parameter estimation (PE) and identifiability analysis (ContourP and RIdent)" and "Defining the experimental data and the corresponding error information".

Last the Fisher Information Matrix based alphabetic criterion for optimal experimental design must be selected:

```
inputs.OEDsol.OEDcost_type  Fisher Information Matrix based OED criterion

> 'Dopt': maximize the determinant of the Fisher Information matrix
> 'Eopt': maximize the minimum eigenvalue
> 'Aopt': maximize the trace
> 'Emod': minimize the ration between the maximum and the minimum
        eigenvalue
> 'DoverE': maximize the ratio Determinant/minimum eigenvalue

See more details in the Brief theoretical introduction.
```

2.5.6 Defining the numerical methods

Initial value problem solution

```
inputs.ivpsol.ivpsolver  Initial value problem solver

> 'radau5' (default Fortran): implicit Runge-Kutta method.
> 'rkf45' (Fortran): Runge-Kutta-Fehlberg
> 'lsoda' (Fortran): Adams with automatic switch to BDF
> 'lsodes', 'lsodesst' (Fortran): for large scale sparse systems
> Matlab ode solvers: 'ode15s', 'ode45', 'ode113'(default)

IMPORTANT: Note that any other method may be used through the option
inputs.model.input_model_type = 'blackboxmodel'.
This possibility can be easily used to handle PDEs or DDEs (when the
linear chain is not suitable).
```

```
inputs.ivpsol.senssolver  Method to compute sensitivities

> 'odessa' (default Fortran): BDF method
> 'sensmat' (Matlab): BDF method modified from ode15s
> 'fdsens' (Fortran-Matlab): finite differences approach

IMPORTANT: Note that for the case
inputs.model.input_model_type = 'blackboxmodel' Finite Differences
('fdsens') should be used.
```

```
inputs.ivpsol.rtol  [OPTIONAL] Relative tolerance for the simulation
                    (default:1e-7)
```

```
inputs.ivpsol.atol  [OPTIONAL] Absolute tolerance for the simulation
                    (default:1e-7)
```

Nonlinear programming problem solution

<code>inputs.nlpsol.nlpsolver</code>	[OPTIONAL] NLP problem solver (optimization method) <pre> >'local_fmincon' 'local_n2fb' 'local_dn2fb' 'local_dhc' 'local_ipopt' 'local_solnp' 'local_nomad' 'local_fsqp' 'local_misqp': Local indirect or direct methods >'multi_fmincon' 'multi_n2fb' 'multi_dn2fb' 'multi_dhc' 'multi_ipopt' 'multi_solnp' 'multi_nomad' 'multi_fsqp' 'multi_misqp': Multistart of local methods >'de': Differential Evolution >'sres': Stochastic Ranking Evolutionary search >'hyb_de_fmincon' 'hyb_de_n2fb' 'hyb_de_dn2fb' 'hyb_de_dhc' 'hyb_de_ipopt' 'hyb_de_solnp' 'hyb_de_nomad' 'hyb_de_fsqp' 'hyb_de_misqp' 'hyb_sres_fmincon' 'hyb_sres_n2fb' 'hyb_sres_dn2fb' 'hyb_sres_fmincon' 'hyb_sres_n2fb' 'hyb_sres_dn2fb' 'hyb_sres_dhc' 'hyb_sres_ipopt' 'hyb_sres_solnp' 'hyb_sres_nomad' 'hyb_sres_fsqp' 'hyb_sres_misqp': Sequential hybrid methods >'ssm'(default) 'fssm' 'ess' Different implementations of Scatter Search >'globalm', clustering method </pre> IMPORTANT remarks: > Solver options may be modified in the files: <code>ssm_options.m</code>, <code>fssm_options.m</code>, <code>ess_options.m</code>, <code>de_options.m</code>, <code>sres_options.m</code>, <code>globalm_options.m</code> by typing <code>edit *_options.m</code> from the AMIGO path or by editing the file inside the solver folder > Sequential hybrid options may be modified in the files: <code>de_options</code> and <code>sres_options</code> > Solver may be directly changed in the command line Examples: <pre> AMIGO_PE('inputfile','r1','local_fmincon') AMIGO_PE('inputfile','r1','de') AMIGO_PE('inputfile','r1','fssm') AMIGO_RIdent('inputfile','r1','local_fmincon') AMIGO_RIdent('inputfile','r1','de') </pre>
<code>inputs.nlpsol.multi_starts</code>	[OPTIONAL] Number of different starts for the multistart method (default:200)

Other optional inputs

<code>inputs.rid.conftrials</code>	[OPTIONAL] Number of trials for the robust identifiability analysis (default:500)
<code>inputs.rank.gr_samples</code>	[OPTIONAL] Number of samples for the global rank (default:10000)

2.6 How to run AMIGO tasks

2.6.1 AMIGO_Startup

AMIGO_Startup	
Syntax	AMIGO_Startup
Description	AMIGO_Startup is devoted to initialize AMIGO in current MATLAB session. It attempts to add paths and to generate necessary files for FORTRAN version usage. Paths will be added at any AMIGO session so as user does not need to modify the MATLAB path
Input Arguments	No arguments
Outputs	32 bits systems MATLAB version 6.5- <7.1. for windows The user will be asked for a FORTRAN compiler compatible with the MATLAB version and to perform mex -setup, in order to be able to use enhanced FORTRAN based mode. MATLAB version 7.1- for windows and linux Mex options file will be created automatically for g95 (provided with AMIGO) 64 bits systems The following message will be displayed: <pre> ----> Adding paths to current MATLAB session.... ----> IMPORTANT!!!: Please note that under WIN or Linux 64bits FORTRAN models can not be used ----> Startup finished.... </pre> to remind the user that enhanced mode is not yet available for 64 bits systems
Practical tips	When using Windows and older MATLAB versions, the Startup will ask to perform the mex -setup option anytime the Startup is performed. Note that, if the user is not changing compiler with other tools, it is not necessary to run the mex -setup anytime. To avoid repeating the mex -setup answer 'no' when asked <pre> -----> Please, type 'yes' if you have the compiler and 'no' otherwise </pre>

2.6.2 AMIGO_Prep

AMIGO_Prep	
Syntax	<code>AMIGO_Prep('input_file_name')</code>
Description	AMIGO_Prep attempts to generate FORTRAN codes and dll or mexw32 files to enable the AMIGO enhanced mode for a given problem. The user should run AMIGO_Prep when: - Before running the first AMIGO task in enhanced (FORTRAN based mode) - Whenever the inputs.model structure for a given example is modified - Whenever the user provided fortran codes are modified - Whenever changing the example in the same session Note that folders keeping problem results will be created under the Results folder (unless otherwise specified). All problem related files (inputs, outputs and intermediate files) will be kept in such folder.
Input Arguments	<code>input_file_name</code>: The input file name within single commas. The input file should be anywhere in the path. Several examples have been incorporated in the folder Examples and the folder Inputs is initially intended for the user to keep his/her own input files.
Outputs	> Warning messages may be displayed when some inputs are missing and default values are to be used. > The folder <code>results.pathd.results_folder</code> selected by the user will be created in the Results folder <code>fcn.f</code>, <code>sens.f</code> files will be generated for the simulation of the model and sensitivities respectively and will be kept in the user selected <code>results.pathd.results_folder</code> folder > <code>.dll</code> or <code>.mexw32</code> files will be generated for the corresponding <code>ivp</code> and <code>sens</code> solver for the model under consideration NOTE: Some warnings may appear during the mex generation and compilation. Most of them are compiler dependent and will not influence the results. If you are experiencing errors or warnings in <code>fcn.f</code> or <code>sens.f</code>, please revise the structure <code>inputs.model</code>; alternatively if warnings or errors appear in your own fortran files, please revise them.
Examples	From the AMIGO path type: <pre>AMIGO_Prep('HH') AMIGO_Prep('Mendes_OED') AMIGO_Prep('circadian_pe')</pre>

2.6.3 AMIGO_SModel

AMIGO_SModel	
Syntax	<code>AMIGO_SModel('input_file_name','run_ident')</code>
Description	Simulates model (all states) under a given experimental scheme and plots states evolution vs time. <code>inputs.model.par</code> and <code>inputs.exps.exp_y0{iexp}</code> will be used for simulation unless <code>inputs.PEsol.global_theta_guess</code>, <code>inputs.PEsol.global_theta_y0_guess</code>, <code>inputs.PEsol.local_theta_guess{iexp}</code> or <code>inputs.PEsol.local_theta_y0_guess{iexp}</code> are defined
Input Arguments	<code>input_file_name</code>: The input file name within single commas. <code>run_ident</code>: [OPTIONAL] Run identifier. <code>Run_ident</code> will be used as part of the results folder name, thus preventing from undesired overwriting of results. Note that user will be asked to provide a new run identifier when there is risk of overwriting previous results. [DEFAULT] <code>run1</code>
Outputs	> Warning messages may be displayed when some inputs are missing and default values are to be used. > The folder <code>SModel_[results.pathd.short_name]_[run_ident]</code> will be created within the folder <code>results.pathd.results_folder</code> with the following contents: - A copy of the input file - A .m report which keeps inputs and outputs - A .mat MATLAB structure file which keeps the inputs. and results. structures - .fig files with plots of states evolution vs time
Examples	From the AMIGO path type: <pre>AMIGO_SModel('HH') or AMIGO_SModel('HH','test1') AMIGO_SModel('Mendes') AMIGO_SModel('Mendes_uvar','uvar') AMIGO_SModel('circadian_pe')</pre>
Practical tips	SModel is useful to detect whether the model and the experiments have been correctly implemented or to analyse the evolution of all states in the model after Parameter Estimation and Optimal Experimental Design. By default stiff IVP solvers have been selected, if the system under consideration is not stiff, try a non-stiff solver to increase efficiency in all tasks.

2.6.4 AMIGO_SObs

AMIGO_SObs	
Syntax	<code>AMIGO_SObs('input_file_name','run_ident')</code>
Description	Simulates model under a given experimental scheme and plots observables evolution vs time. <code>inputs.model.par</code> and <code>inputs.exps.exp_y0{iexp}</code> will be used for simulation unless <code>inputs.PEsol.global_theta_guess</code>, <code>inputs.PEsol.global_theta_y0_guess</code>, <code>inputs.PEsol.local_theta_guess{iexp}</code> or <code>inputs.PEsol.local_theta_y0_guess{iexp}</code> are defined
Input Arguments	<code>input_file_name</code>: The input file name within single commas. <code>run_ident</code>: [OPTIONAL] Run identifier. <code>Run_ident</code> will be used as part of the results folder name, thus preventing from undesired overwriting of results. Note that user will be asked to provide a new run identifier when there is risk of overwriting previous results. [DEFAULT] <code>run1</code>
Outputs	> Warning messages may be displayed when some inputs are missing and default values are to be used. > The folder <code>SObs_[results.pathd.short_name]_[run_ident]</code> will be created within the folder <code>results.pathd.results_folder</code> including: - A copy of the input file - A .m report which keeps inputs and outputs - A .mat MATLAB structure file which keeps the inputs. and results. structures - .fig files with plots of observables evolution vs time
Examples	From the AMIGO path type: <code>AMIGO_SObs('HH')</code> or <code>AMIGO_SObs('HH','test1')</code> <code>AMIGO_SObs('Mendes')</code> <code>AMIGO_SObs('Mendes_uvar','uvar')</code> <code>AMIGO_SObs('circadian_pe')</code>
Practical tips	SObs is useful to detect whether the model, observables and the experiments have been correctly implemented. This option may be used to try different observation functions, thus allowing for qualitative experimental design.

2.6.5 AMIGO_SData

AMIGO_SData	
Syntax	<code>AMIGO_SData('input_file_name','run_ident')</code>
Description	Is intended to either generate pseudo-experimental data for the given observables, experimental scheme and experimental noise or to simulate real experimental data. <code>inputs.model.par</code> and <code>inputs.exps.exp_y0{iexp}</code> will be used for simulation unless <code>inputs.PEsol.global_theta_guess</code>, <code>inputs.PEsol.global_theta_y0_guess</code>, <code>inputs.PEsol.local_theta_guess{iexp}</code> or <code>inputs.PEsol.local_theta_y0_guess{iexp}</code> are defined Note that the observables may be very far from the experimental data if non-optimal values for the unknowns are being used.
Input Arguments	<code>input_file_name</code>: The input file name within single commas. <code>run_ident</code>: [OPTIONAL] Run identifier. <code>Run_ident</code> will be used as part of the results folder name, thus preventing from undesired overwriting of results. Note that user will be asked to provide a new run identifier when there is risk of overwriting previous results. [DEFAULT] <code>run1</code>
Outputs	> Warning messages may be displayed when some inputs are missing and default values are to be used. > The folder <code>SData_[results.pathd.short_name]_[run_ident]</code> will be created within the folder <code>results.pathd.results_folder</code> including: - A copy of the input file - A .m report which keeps inputs and outputs - A .mat MATLAB structure file which keeps the inputs. and results. structures - .fig files with plots of observables evolution vs time plus data (with error bars)
Examples	From the AMIGO path type: <code>AMIGO_SData('HH')</code> (plots data) <code>AMIGO_SData('circadian_rdata','real')</code> (plots data) <code>AMIGO_SData('circadian_pdata','pseudo')</code> (generates pseudo-data)
Practical tips	SData is useful to detect whether the experimental data have been correctly implemented. It is possible to generate pseudo-data and afterwards use them as 'real' for numerical tests.

2.6.6 AMIGO_LRank

AMIGO_LRank	
Syntax	<code>AMIGO_LRank('input_file_name','run_ident')</code>
Description	Is intended to analyse to what extent model unknowns are influencing the observables. With this aim, computes local sensitivities and rank of model unknowns per experiment per observable and provides an overall rank of unknowns. <code>inputs.model.par</code> and <code>inputs.exps.exp_y0{iexp}</code> will be used for simulation unless <code>inputs.PEsol.global_theta_guess</code>, <code>inputs.PEsol.global_theta_y0_guess</code>, <code>inputs.PEsol.local_theta_guess{iexp}</code> or <code>inputs.PEsol.local_theta_y0_guess{iexp}</code> are defined
Input Arguments	<code>input_file_name</code>: The input file name within single commas. <code>run_ident</code>: [OPTIONAL] Run identifier. <code>Run_ident</code> will be used as part of the results folder name, thus preventing from undesired overwriting of results. Note that user will be asked to provide a new run identifier when there is risk of overwriting previous results. [DEFAULT] <code>run1</code>
Outputs	> Warning messages may be displayed when some inputs are missing and default values are to be used. > Tabular results of rankings per experiment and overall ranking > The folder <code>LRank_[results.pathd.short_name]_[run_ident]</code> will be created within the folder <code>results.pathd.results_folder</code> including: - A copy of the input file - A .m report which keeps inputs and outputs - A .mat MATLAB structure file which keeps the inputs. and results. structures - .fig files with 2D and bar plots of sensitivities per experiment and a plot of overall ranking of parameters
Examples	<code>AMIGO_LRank('HH')</code> or <code>AMIGO_LRank('HH','test1')</code> <code>AMIGO_LRank('Mendes')</code> <code>AMIGO_LRank('Mendes_uvar','uvar')</code> <code>AMIGO_LRank('circadian_pe')</code>
Practical tips	LRank may be used after Parameter Estimation to detect whether the observables are insensitive to some unknowns or to analyse whether the available experimental scheme (or a different one) is being informative. This will provide with useful information for OED.

2.6.7 AMIGO_GRank

AMIGO_GRank	
Syntax	<code>AMIGO_GRank('input_file_name','run_ident')</code>
Description	Is intended to analyse to what extent model unknowns are influencing the observables within the allowed values for the unknowns. Computes global sensitivities and rank of model unknowns per experiment per observable and provides an overall rank of unknowns. Values of unknowns will be taken within the maximum and minimum defined for (global) parameters and initial conditions.
Input Arguments	<code>input_file_name</code>: The input file name within single commas. <code>Run_ident</code> will be used as part of the results folder name, thus preventing from undesired overwriting of results. Note that user will be asked to provide a new run identifier when there is risk of overwriting previous results. [DEFAULT] <code>run1</code>
Outputs	> Warning messages may be displayed when some inputs are missing and default values are to be used. > Tabular results of ranks per experiment and overall rank > The folder <code>GRank_[results.pathd.short_name]_[run_ident]</code> will be created within the folder <code>results.pathd.results_folder</code> including: - A copy of the input file - A .m report which keeps inputs and outputs - A .mat MATLAB structure file which keeps the inputs. and results. structures - Tabular results of rankings per experiment and overall ranking - .fig files with 2D and bar plots of global sensitivities per experiment and a plot of overall ranking of parameters
Examples	<code>AMIGO_GRank('HH')</code> or <code>AMIGO_GRank('HH','test1')</code> <code>AMIGO_GRank('Mendes')</code> <code>AMIGO_GRank('Mendes_uvar','uvar')</code> <code>AMIGO_GRank('circadian_pe')</code>
Practical tips	GRank may be used before Parameter Estimation to detect whether the observables are insensitive to some unknowns or to analyse if the available experimental scheme is being informative. This allows to anticipate lack or poor identifiability and provides some clues for OED. Note that the computational cost increases very rapidly with the size of the model and the number of experiments. The overall process may last from several minutes to hours.

2.6.8 AMIGO_PE

AMIGO_PE	
Syntax	<code>AMIGO_PE('input_file_name','run_ident','NLP_solver')</code>
Description	Attempts to estimate model global or local unknowns from experimental data.
Input Arguments	<code>input_file_name</code>: The input file name within single commas. <code>run_ident</code>: [OPTIONAL] Run identifier. Run_ident will be used as part of the results folder name, thus preventing from undesired overwriting of results. [DEFAULT] <code>run1</code> <code>NLP_solver</code>: [OPTIONAL] Optimization method. Several runs may be performed with the same or different NLP solvers available. [DEFAULT] <code>ssm</code>
Outputs	> Warning messages may be displayed when some inputs are missing and default values are to be used. > Best unknowns with Crammer-Rao confidence intervals > The folder <code>PE_[results.pathd.short_name]_[NLP_solver]_[run_ident]</code> will be created in the folder <code>results.pathd.results_folder</code> including: - A copy of the input file - A .m report and .mat MATLAB structure file to keep inputs. and results. - .fig files with best fits, histograms of solutions for multistarts, convergence curve for NLP solvers and a plot of the correlation matrix per experiment and overall
Examples	<code>AMIGO_PE('HH','r1')</code>, <code>AMIGO_PE('HH','r2')</code> (two runs of PE with <code>ssm</code>) <code>AMIGO_PE('HH','r1','multi_fmincon')</code> (PE with multistart of <code>fmincon</code>) <code>AMIGO_PE('Mendes_uvar','uvar','de')</code> (PE with DE) <code>AMIGO_PE('circadian_pe','local_n2fb')</code> (PE with local <code>n2fb</code>)
Practical tips	Try local methods first (e.g., <code>n2fb</code> and/or <code>fmincon</code>). A bad fit may mean a local (sub-optimal) solution. Solve the problem with a multistart. Histograms of solutions will help to detect multimodality and/or poor identifiability. Use a global or a hybrid method to solve the problem. If a good fit is obtained but confidence intervals are NaN or too large, try to fix unknowns (use <code>GRank</code> to fix the less influencing unknowns). When possible try to improve identifiability via new optimally designed experiments (OED).

2.6.9 AMIGO_ContourP

AMIGO_PE	
Syntax	AMIGO_Contour('input_file_name','run_ident')
Description	Attempts to visualise poor or lack of practical identifiability by plotting the Weighted Least Squares or the Log-Likelihood by pairs of parameters Unknowns guess (possibly obtained through PE) will be used as reference. Results will be displayed within the selected max and min values for the unknowns.
Input Arguments	input_file_name: The input file name within single commas. run_ident: [OPTIONAL] Run identifier. Run_ident will be used as part of the results folder name, thus preventing from undesired overwriting of results. [DEFAULT] run1
Outputs	> Warning messages may be displayed when some inputs are missing and default values are to be used. > Cost function contour plots by pairs of parameters > Folder ContourP_[results.pathd.short_name]_[NLP_solver]_[run_ident] will be created in the folder results.pathd.results_folder including: - A copy of the input file - A .m report and .mat MATLAB structure file to keep inputs. and results. - .fig files with contour plots by pairs of parameters
Examples	AMIGO_ContourP('HH') AMIGO_ContourP('Mendes_uvar','opt1') AMIGO_ContourP('circadian_pe')
Practical tips	Perform ContourP around the optimal solution found in Parameter Estimation when large or NaN confidence intervals are obtained. Infinite contour plots would indicate poor or lack of identifiability. Large "white" areas, will indicate a flat cost function. Try to reduce the bounds for unknowns in order to check whether results may improve.

2.6.10 AMIGO_RIdent

AMIGO_RIdent	
Syntax	AMIGO_RIdent('input_file_name','run_ident','NLP_solver')
Description	Performs robust identifiability analysis by means of a Monte-Carlo based approach. Unknowns guess (possibly obtained through PE) will be used as starting point and reference.
Input Arguments	input_file_name: The input file name within single commas. run_ident: [OPTIONAL] Run identifier. Run_ident will be used as part of the results folder name, thus preventing from undesired overwriting of results. [DEFAULT] run1 NLP_solver: [OPTIONAL] Optimization method. Several runs may be performed with the same or different NLP solvers available. [DEFAULT] ssm
Outputs	> Warning messages may be displayed when some inputs are missing and default values are to be used. > Robust confidence regions, hyper-ellipsoid volume and eccentricity, mean value and distance to the best (reference) value > Folder <code>RIdent_[results.pathd.short_name]_[NLP_solver]_[run_ident]</code> will be created in the folder <code>results.pathd.results_folder</code> including: - A copy of the input file - A .m report and .mat MATLAB structure file to keep inputs. and results. - .fig files with clouds of solutions by pairs of unknowns, robust confidence regions, eccentricity plot
Examples	AMIGO_RIdent('HH') or AMIGO_RIdent('HH','r1') (RIdent with ssm) AMIGO_RIdent('HH','r1','local_fmincon') (RIdent with local fmincon) AMIGO_RIdent('circadian_pe','r1','local_dn2fb') (RIdent with dn2fb)
Practical tips	RIdent solves the Parameter Estimation problem hundreds of times, thus an adequate selection of the NLP solver and its options is critical to minimize computational cost: - If the problem is convex use a local solver. - If the problem is multimodal ssm (default) is recommended. Please, modify ssm options by editing <code>ssm_options_conf</code>. Use <code>opts.maxtime</code> so as to guarantee that the local solver within ssm is called at least once. If plots reveal clouds of solutions with bands (like "cebra skin") it is needed either to switch from a local to a global solver or to increase the value of <code>opts.maxtime</code> for ssm to converge.

2.6.11 AMIGO_OED

AMIGO_OED	
Syntax	<code>AMIGO_OED('input_file_name','run_ident','NLP_solver')</code>
Description	Attempts to compute optimal dynamic experiments for the purpose of parameter estimation.
Input Arguments	<code>input_file_name</code>: The input file name within single commas. <code>run_ident</code>: [OPTIONAL] Run identifier. Run_ident will be used as part of the results folder name, thus preventing from undesired overwriting of results. [DEFAULT] run1 <code>NLP_solver</code>: [OPTIONAL] Optimization method. Several runs may be performed with the same or different NLP solvers available. [DEFAULT] ssm
Outputs	> Warning messages may be displayed when some inputs are missing and default values are to be used. > Optimally designed experiments and expected confidence intervals. > The folder <code>OED_[results.pathd.short_name]_[NLP_solver]_[run_ident]</code> will be created in the folder <code>results.pathd.results_folder</code> including: - A copy of the input file - A .m report and .mat MATLAB structure file to keep inputs. and results. - .fig files with the experimental scheme (fixed and optimally designed) and plots of the correlation matrix
Examples	<code>AMIGO_OED('Mendes_oed')</code> (OED with ssm) <code>AMIGO_OED('Mendes_oed','r1','hyb_de_fmincon')</code> (OED with sequential hybrid) <code>AMIGO_OED('circadian_oed','r1','local_fmincon')</code> (OED with fmincon)
Practical tips	Local solvers <code>n2fb</code> and <code>dn2fb</code> are specific for parameter estimation thus they can not be used for OED (even within ssm or fssm). OED should be focused in poorly identifiable unknowns. Introduce fixed experiments when available (sequential-parallel design). This makes the OED to result in experiments that complement existent information. Optimal 'sustained' or 'pulse'-wise experiments may be suboptimal. If possible in the lab, try 'step'-wise profiles, first. Note that, if optimal, you may end up in 'sustained' or 'pulse'-wise experiments.

Appendix A

Illustrative examples

This appendix is devoted to illustrate the different possibilities of using AMIGO with a number of practical examples. For each example several tasks will be performed and results interpreted.

- The Hodgking and Huxley model [20] will be used to illustrate:
 - How MATLAB, FORTRAN, sbml, charmodels or blackbox models may be introduced in AMIGO.
- A model of the circadian clock in *Arabidopsis thaliana* [29] will be used to illustrate:
 - How to implement an experimental scheme
 - The results obtained by performing SModel, SObs, LRank and GRank and how to interpret them
- A model of the NF κ B signalling module [27] will be used to illustrate:
 - The generation of pseudo-experimental data
 - The solution of the parameter estimation problem with different methods and how to interpret the results
 - How to perform the practical identifiability analysis via ContourP and RIdent and how to interpret the results
- The model of a three step pathway by Mendes [31] will be used to illustrate:
 - How to implement different stimulation profiles for the case of having several controls
 - How to solve the Optimal Experimental Design problem with different interpolations and solvers and how to interpret the results

A.1 The Hodgking and Huxley model

A.1.1 Introduction

The Hodgkin and Huxley model [20] describes how action potentials in neurons are initiated and propagated. It consists of a set of nonlinear ordinary differential equations that approximate the electrical characteristics of excitable cells such as neurons and cardiac myocytes. It was initially proposed to explain the ionic mechanisms underlying the initiation and propagation of action potentials in the squid giant axon.

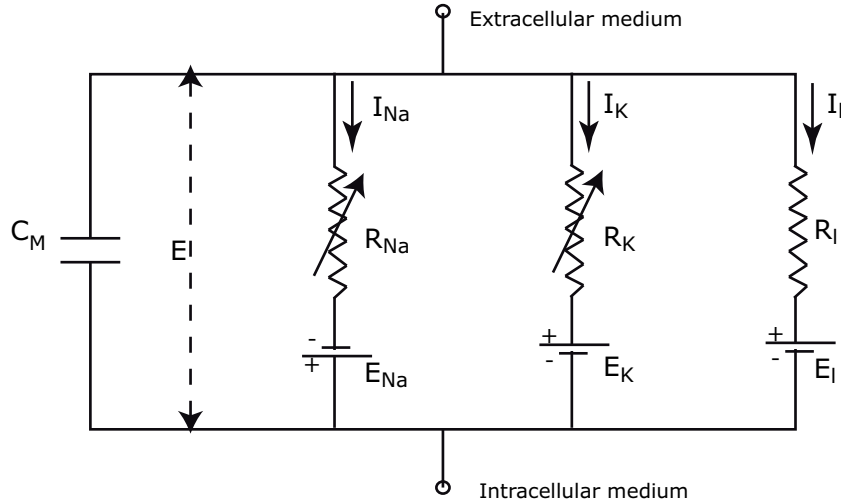


Figure A.1: Electrical circuit representing the membrane [20].

$$\begin{aligned}
 yv &= -(g_{Na}ym^3yh(yv - V_{Na}) + g_Kyn^4(yv - V_K) + g_L(yv - V_L) - TotalI)/Cm \\
 yn &= A_n(1 - yn) - B_nyn \\
 ym &= A_m(1 - ym) - B_ym \\
 yh &= A_h(1 - yh) - B_hyh
 \end{aligned}$$

with:

$$\begin{aligned}
 A_n &= 0.01 * ((10 - yv)/(exp((10 - yv) * 0.1) - 1)) \\
 B_n &= 0.125 * exp(-yv/80) \\
 A_m &= 0.1 * ((25 - yv)/(exp((25 - yv) * 0.1) - 1)) \\
 B_m &= 4 * exp(-yv/18) \\
 A_h &= 0.07 * exp(-yv/20) \\
 B_h &= 1/(1 + exp((30 - yv) * 0.1))
 \end{aligned} \tag{A.1}$$

being g_{Na} , g_K , g_L , V_{Na} , V_K , C_m the model parameters and $TotalI$ the total membrane current (the stimulus or input).

All different model implementations of the Hodgkin and Huxley example can be found in the **Examples** folder in the toolbox. Here all possibilities are described:

A.1.2 Input the model to automatically generate FORTRAN or MATLAB

`charmodelF` or `charmodelM`

```
%=====
% MODEL RELATED DATA
%=====
% Model introduction: 'charmodelF' | 'charmodelM' allows to input the model as string.
% Either FORTRAN or MATLAB code will be automatically generated.
inputs.model.input_model_type='charmodelF';
inputs.model.n_st=4;           % Number of states
inputs.model.n_par=7;          % Number of model parameters
inputs.model.n_stimulus=1;      % Number of inputs, stimuli or controls
inputs.model.names_type='custom'; % [] Names given to states/pars/inputs:
                                % 'standard' (x1,x2,...p1,p2...,u1, u2,...) |
                                % 'custom'(default).
                                % IMPORTANT: for standard names following
                                % *_names inputs are not required
inputs.model.st_names=char('yv','yn','ym','yh'); % Names of the states
inputs.model.par_names=char('gNa','gK','gL','VNa',...
                             'VK','VL','Cm'); % Names of the parameters
inputs.model.stimulus_names=char('TotalI'); % Names of the stimuli
% Equations describing system dynamics. Time derivatives are regarded 'd'st_name'
inputs.model.eqns= char('An=0.01*((10-yv)/(exp((10-yv)*0.1)-1))',...
                        'Bn=0.125*exp(-yv/80)',...
                        'Am=0.1*((25-yv)/(exp((25-yv)*0.1)-1))',...
                        'Bm=4*exp(-yv/18)',...
                        'Ah=0.07*exp(-yv/20)',...
                        'Bh=1/(1+exp((30-yv)*0.1))',...
                        'dyv=-(gNa*ym^3*yh*(yv-VNa)+ gK*yn^4*(yv-VK)+gL*(yv-VL)-TotalI) / Cm',...
                        'dyn= An*(1-yn)-Bn*yn',...
                        'dym= Am*(1-ym)-Bm*ym',...
                        'dyh= Ah*(1-yh)-Bh*yh');
```

A.1.3 Input the model in FORTRAN, MATLAB or SBML

```

fortranmodel, matlabmodel, sbmlmodel

%=====
% MODEL RELATED DATA
%=====
% Model introduction: 'matlabmodel' | 'fortranmodel' | 'sbmlmodel' allows to
% input the model as .m, .f or .xml file.
inputs.model.input_model_type='matlabmodel';
inputs.model.matlabmodel_file='HHmodel'; % File including the system dynamics
% IMPORTANT: for 'fortranmodel': inputs.model.fortranmodel_file='HHmodel';
%                               inputs.model.fortransens_file='HHmodels';
%                               for 'sbmlmodel': inputs.model.sbmlmodel_file='BIOMD0000000020';
inputs.model.n_st=4;           % Number of states
inputs.model.n_par=7;          % Number of model parameters
inputs.model.n_stimulus=1;     % Number of inputs, stimuli or controls
inputs.model.names_type='custom'; % [] Names given to states/pars/inputs:
                                % 'standard' (x1,x2,...,p1,p2...,u1, u2,...) |
                                % 'custom'(default).
                                % IMPORTANT: for standard names following
                                % *_names inputs are not required
inputs.model.st_names=char('yv','yn','ym','yh'); % Names of the states
inputs.model.par_names=char('gNa','gK','gL','VNa',...
                             'VK','VL','Cm');    % Names of the parameters
inputs.model.stimulus_names=char('TotalI');      % Names of the stimuli

```

Example of MATLAB model file HHmodel.m

```

function ydot= HHmodel(t,y,flag,par,v,pend,told)
%IMPORTANT: > Arguments should be t,y,flag,par,v,pend,told
%           > Inputs or stimuli should be defined as:
%           u(iu)=v(iu)+(t-told)*pend(iu); iu=1:inputs.model.n_stimulus
%           > Assignments such as yv=y(1); gNa=par(1); TotalI=u(1) and
%           ydot=[dyv;dyn;dym;dyh]; are required
u(1)=v(1)+(t-told)*pend(1);

yv=y(1);
yn=y(2);
ym=y(3);
yh=y(4);
gNa=par(1);
gK=par(2);
gL=par(3);
VNa=par(4);
VK=par(5);
VL=par(6);
Cm=par(7);
TotalI=u(1);

An=0.01*((10-yv)/(exp((10-yv)*0.1)-1));
Bn=0.125*exp(-yv/80);
Am=0.1*((25-yv)/(exp((25-yv)*0.1)-1));
Bm=4*exp(-yv/18);
Ah=0.07*exp(-yv/20);
Bh=1/(1+exp((30-yv)*0.1));
dyv=-(gNa*ym^3*yh*(yv-VNa)+ gK*yn^4*(yv-VK)+gL*(yv-VL)- TotalI ) / Cm;
dyn= An*(1-yn)-Bn*yn;
dym= Am*(1-ym)-Bm*ym;
dyh= Ah*(1-yh)-Bh*yh;

ydot=[dyv;dyn;dym;dyh];
return

```

Example of FORTRAN model file HHmodel.f

```

SUBROUTINE FCN(N,T,Y,YDOT,PAR,IPAR,V,PEND,TLAST)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DOUBLE PRECISION dyv,dyn,dym,dyh
DOUBLE PRECISION yv,yn,ym,yh
DOUBLE PRECISION gNa,gK,gL,VNa,VK,VL,Cm
DOUBLE PRECISION TotalI
DIMENSION Y(N),YDOT(N),PAR(*),IPAR(*),V(*),PEND(*)
DIMENSION U(25)

c  IMPORTANT:      > Arguments should be N,T,Y,YDOT,PAR,IPAR,V,PEND,TLAST
c                  > Inputs or stimuli should be defined as:
c                  u(iu)=v(iu)+(t-told)*pend(iu); iu=1:inputs.model.n_stimulus
c                  > All states, pars, and stimuli should be declared as double
c                  precision
c                  > Sentences:
c                  DIMENSION Y(N),YDOT(N),PAR(*),IPAR(*),V(*),PEND(*)
c                  DIMENSION U(25)
c                  are compulsory and should be written as in this example
c                  > Assignments such as yv=y(1); gNa=par(1); TotalI=u(1) and
c                  ydot(1)=dyv are required

yv=y(1)
yn=y(2)
ym=y(3)
yh=y(4)
gNa=par(1)
gK =par(2)
gL =par(3)
VNa=par(4)
VK =par(5)
VL =par(6)
Cm =par(7)
u(1)=v(1)+(t-tlast)*pend(1)
TotalI=u(1)
An=0.01*((10-yv)/(exp((10-yv)*0.1)-1))
Bn=0.125*exp(-yv/80)
Am=0.1*((25-yv)/(exp((25-yv)*0.1)-1))
Bm=4*exp(-yv/18)
Ah=0.07*exp(-yv/20)
Bh=1/(1+exp((30-yv)*0.1))
dyv=-(gNa*ym**3*yh*(yv-VNa)+ gK*yn**4*(yv-VK)+gL*(yv-VL)-TotalI)/Cm
dyn= An*(1-yn)-Bn*yn
dym= Am*(1-ym)-Bm*ym
dyh= Ah*(1-yh)-Bh*yh
ydot(1)=dyv
ydot(2)=dyn
ydot(3)=dym
ydot(4)=dyh
RETURN
END

```

Example of FORTRAN sensitivities file HHmodels.f

```

SUBROUTINE SENS(N, T, Y, PAR, YDOT)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DIMENSION Y(N),YDOT(N),PAR(*),U(25)
DOUBLE PRECISION dyv,dyn,dym,dyh
DOUBLE PRECISION yv,yn,ym,yh
DOUBLE PRECISION gNa,gK,gL,VNa,VK,VL,Cm
DOUBLE PRECISION TotalI
DIMENSION Y(N),YDOT(N),PAR(*),IPAR(*),V(*),PEND(*)
DIMENSION U(25)
COMMON /CONTROLS/ V(25), PEND(25), TLAST
c IMPORTANT:      > Arguments should be N,T,Y,YDOT,PAR,IPAR,V,PEND,TLAST
c                > Inputs or stimuli should be defined as:
c                u(iu)=v(iu)+(t-told)*pend(iu); iu=1:inputs.model.n_stimulus
c                > All states, pars, and stimuli should be declared as double
c                precision
c                > Sentences:
c                DIMENSION Y(N),YDOT(N),PAR(*),IPAR(*),V(*),PEND(*)
c                DIMENSION U(25)
c                are compulsory and should be written as in this example
c                > Assignments such as yv=y(1); gNa=par(1); TotalI=u(1) and
c                ydot(1)=dyv are required
c
yv=y(1)
yn=y(2)
ym=y(3)
yh=y(4)
gNa=par(1)
gK =par(2)
gL =par(3)
VNa=par(4)
VK =par(5)
VL =par(6)
Cm =par(7)
u(1)=v(1)+(t-tlast)*pend(1)
TotalI=u(1)
An=0.01*((10-yv)/(exp((10-yv)*0.1)-1))
Bn=0.125*exp(-yv/80)
Am=0.1*((25-yv)/(exp((25-yv)*0.1)-1))
Bm=4*exp(-yv/18)
Ah=0.07*exp(-yv/20)
Bh=1/(1+exp((30-yv)*0.1))
dyv=-(gNa*ym**3*yh*(yv-VNa)+ gK*yn**4*(yv-VK)+gL*(yv-VL)-TotalI)/Cm
dyn= An*(1-yn)-Bn*yn
dym= Am*(1-ym)-Bm*ym
dyh= Ah*(1-yh)-Bh*yh
ydot(1)=dyv
ydot(2)=dyn
ydot(3)=dym
ydot(4)=dyh
RETURN
END

```

A.1.4 Input the model as a blackbox model

blackboxmodel

```
%=====
% MODEL RELATED DATA
%=====
% Model introduction: 'blackboxmodel' allows to input a MATLAB function
% that simulates system dynamics. This function will be called to compute states
% and sensitivities for every experiment.
% REMARK: this allows to handle PDEs, DDEs, etc.
%
inputs.model.input_model_type='blackboxmodel';
inputs.model.matlabmodel_file='HHbbmodel'; % File including the simulation of the given model
inputs.model.n_st=4; % Number of states
inputs.model.n_par=7; % Number of model parameters
inputs.model.n_stimulus=1; % Number of inputs, stimuli or controls
inputs.model.names_type='custom'; % [] Names given to states/pars/inputs:
% 'standard' (x1,x2,...p1,p2...,u1, u2,...)|
% 'custom'(default).
% IMPORTANT: for standard names following
% *_names inputs are not required
inputs.model.st_names=char('yv','yn','ym','yh'); % Names of the states
inputs.model.par_names=char('gNa','gK','gL','VNa',...
    'VK','VL','Cm'); % Names of the parameters
inputs.model.stimulus_names=char('TotalI'); % Names of the stimuli
```

Example of a blackbox model file HHbbmodel.m

```

function [yteor,iflag] = HHbbmodel(t0,tf,ts,y_0,par,u,pend,tu)

% INPUT Arguments should be t0,tf,ts,y_0,par,u,pend,tu
%      t0: initial time for integration
%      tf: final time for integration
%      ts: vector of sampling times
%      y0: vector of initial conditions
%      par: vector of parameter values
%      u: vector of control values
%      pend: vector of slope values used for control linear interpolation
%      tu: vector of control switching times
%      These will be automatically introduced by AMIGO for each experiment
% OUTPUT Arguments: yteor and iflag
%      yteor: matrix of [number of sampling times x number of states] with
%             the values of all states at sampling times
%      iflag: negative if an integration error occurred
%
%
%      This example solves the HH model by means of ode15s.
%      REMARK: user may call any software from here provided it is compatible
%      with MATLAB (for example a PDE solver, DDE solvers, etc.

% Vector of times for which the simulation should stop
% includes sampling times and control switching times
vtout=union(ts,tu);

% Assign initial conditions & Initialize integration times counter i_int

if vtout(1)==t0
    yteor(1,:)=y_0; i_int=2;
else
    i_int=1;
end

% Assign solver options
options = odeset('RelTol',1e-7,'AbsTol',1e-7);
% Initialize control element counter i_con
i_con=1;
% Initialize t_old, this will be used for linear interpolated controls
t_old=tu(1);
% INTEGRATION LOOP
for i_out=1:size(vtout,2)-1
    tin=vtout(i_out);
    tout=vtout(i_out+1);
    [t,yout] = ode15s('HHmodel',[tin tout],y_0,options,par',u(:,i_con),pend(:,i_con),t_old);

    % Keep values to next integration step
    y_0=yout(size(t,1),:);

    % If t out= sampling time, keep information
    if tout==ts(i_int)
        yteor(i_int,:)=yout(size(t,1),:); i_int=i_int+1; end

    % If t out= t control, update control value
    if (size(u,2)>1)
        if (tout>=tu(i_con+1)) & ((i_con+1)<size(u,2)+1)
            i_con=i_con+1;end; end
        t_old=tu(i_con);
    end % END INTEGRATION LOOP

% Assign iflag value
iflag=1;
return

```

A.2 A model of the circadian clock in *Arabidopsis thaliana*

The model describes the first multi-gene loop identified in the *Arabidopsis* circadian clock ([29]) that comprises a negative feedback loop, in which two partially redundant genes Late Elongated Hypocotyl (LHY) and Circadian Clock Associated 1 (CCA1) repress the expression of their activator, Timing of CAB Expression 1 (TOC1):

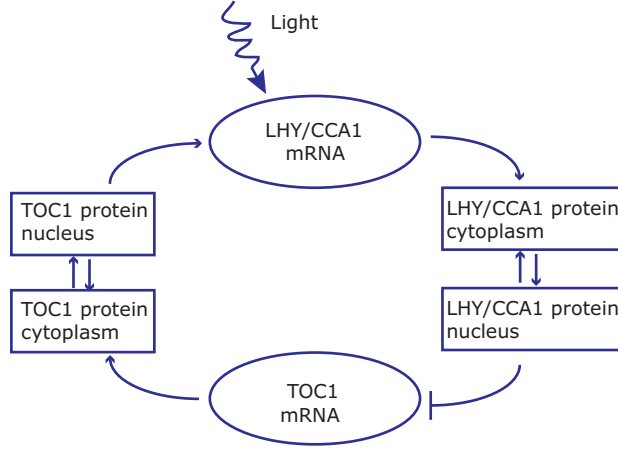


Figure A.2: Model for the central feedback loop in the *Arabidopsis* clock [29].

The resultant mathematical model consists of 7 differential equations:

$$\begin{aligned}
 C\dot{L}_m &= q1CP_n\theta_{light} + n1\frac{CT_n}{g1 + CT_n} - m1\frac{CL_m}{k1 + CL_m} \\
 C\dot{L}_c &= p1CL_m - r1CL_c + r2CL_n - m2\frac{CL_c}{k2 + CL_c} \\
 C\dot{L}_n &= r1CL_c - r2CL_n - m3\frac{CL_n}{k3 + CL_n} \\
 C\dot{T}_m &= n2\frac{g2^2}{g2 + CL_n^2} - m4\frac{CT_m}{k4 + CT_m} \\
 C\dot{T}_c &= p2CT_m - r3CT_c + r4CT_n - m5\frac{CT_c}{k5 + CT_c} \\
 C\dot{T}_n &= r3CT_c - r4CT_n - m6\frac{CT_n}{k6 + CT_n} \\
 C\dot{P}_n &= (1 - \theta_{light})p3 - m7\frac{CP_n}{k7 + CP_n} - q2lightCP_n
 \end{aligned} \tag{A.2}$$

where m, c and n denote that it is the corresponding mRNA, or protein in the cytoplasm or nucleus, respectively.

Michealis–Menten kinetics are used to describe enzyme-mediated degradation of proteins, and Hill functions are used to describe the transcriptional activation term of the mRNA for LHY and TOC1. As a result 27 parameters have to be estimated from experimental data. Note however that can be demonstrated that the model is structurally not identifiable: by taking into account the structure of the model and the observation function, it may be shown that g_2 , m_2 , m_3 , k_2 , k_3 , p_1 , r_1 and r_2 can not

be estimated independently of the experimental scheme. In addition the remaining parameters will be only locally identifiable, some of them should be fixed so as to be able to uniquely identify the others. A rank of parameters, which assesses the influence of each parameter in the observables, may help to decide which parameters to fix.

To illustrate the usage of AMIGO for this purpose we will implement the input file to perform global rank of parameters, and we will use it to simulate the model, the observables and to locally and globally rank the parameters.

The following experimental scheme will be considered:

- Two experiments may be performed under sustained and pulse-wise θ_{light} profiles.
- Between 15 and 25 equidistant measurements are performed over 120 min duration experiments.
- Only the luminiscence and the mRNA amplitude are experimentally available.

The folder *Arabidopsis circadian* within the folder *Examples* in the toolbox collects different files with the minimum inputs to perform the different tasks available in AMIGO. Here the file *circadian grank* is shown to illustrate its use for Smodel, SObs, LRank and GRank.

Regarding the selection of the bounds for the global sensitivity analysis, it has been that ranking will depend on the range of parameters selected. Thus we may perform different tests and take the average value.

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TITLE: The circadian clock in Arabidopsis thaliana
%
%
%      Type :
%          > help circadian_tutorial
%      for a more detailed description of the model.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%      INPUT FILE FOR GLOBAL RANK
%
%      This is the minimum input file for global rank.
%      Minimum required inputs:
%          > Paths related data
%          > Model:          model_type; n_st; n_par; n_stimulus;
%                           st_names; par_names; stimulus_names; eqns; par
%          > Experimental scheme: n_exp; exp_y0iexp; t_fiexp;
%                           u_interpiexp; t_coniexp; uiexp
%                           n_obsiexp; obs_namesiexp; obsiexp
%
%      (AMIGO_GRank)==>> [n_siexp]; [t_siexp];
%                           id_global_theta; [id_global_theta_y0]
%                           [id_local_thetaiexp];
%                           [id_local_theta_y0iexp]global_theta_max;
%                           global_theta_min
%                           [global_theta_y0_max];[global_theta_y0_min]
%                           [local_theta_maxiexp];[local_theta_minexp]
%                           [local_theta_y0_maxiexp];[local_theta_y0_minexp]
%                           []:optional inputs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%=====
% PATHS RELATED DATA
%=====

results.pathd.results_folder='circadian-tutorial'; % Folder to keep results (within Results)
results.pathd.short_name='circadian';              % To identify figures and reports

%=====
% MODEL RELATED DATA
%=====

inputs.model.input_model_type='charmodelF';
inputs.model.n_st=7;                               % Number of states
inputs.model.n_par=27;                              % Number of model parameters
inputs.model.n_stimulus=1;                          % Number of stimuli variables
inputs.model.st_names=char('CL_m','CL_c','CL_n','CT_m','CT_c','CT_n','CP_n'); % Names of the states
inputs.model.par_names=char('n1','n2','g1','g2','m1','m2','m3','m4','m5','m6',...
                             'm7','k1','k2','k3','k4','k5','k6','k7','p1','p2',...
                             'p3','r1','r2','r3','r4','q1','q2'); % Names of the parameters
inputs.model.stimulus_names=char('light');           % Names of the stimuli
inputs.model.eqns=... % System dynamics. Time derivatives are regarded 'd'st_name'
    char('dCL_m=q1*CP_n*light+n1*CT_n/(g1+CT_n)-m1*CL_m/(k1+CL_m)',...
          'dCL_c=p1*CL_m-r1*CL_c+r2*CL_n-m2*CL_c/(k2+CL_c)',...
          'dCL_n=r1*CL_c-r2*CL_n-m3*CL_n/(k3+CL_n)',...
          'dCT_m=n2*g2**2/(g2**2+CL_n**2)-m4*CT_m/(k4+CT_m)',...
          'dCT_c=p2*CT_m-r3*CT_c+r4*CT_n-m5*CT_c/(k5+CT_c)',...
          'dCT_n=r3*CT_c-r4*CT_n-m6*CT_n/(k6+CT_n)',...
          'dCP_n=(1-light)*p3-m7*CP_n/(k7+CP_n)-q2*light*CP_n');
inputs.model.par=[7.5038 0.6801 1.4992 3.0412 10.0982... % Nominal value for the parameters
                  1.9685 3.7511 2.3422 7.2482 1.8981 1.2 3.8045...
                  5.3087 4.1946 2.5356 1.4420 4.8600 1.2 2.1994...
                  9.4440 0.5 0.2817 0.7676 0.4364 7.3021 4.5703 1.0];

```

```

%=====
% EXPERIMENTAL SCHEME RELATED DATA
%=====

inputs.exps.n_exp=2; % Number of experiments

% EXPERIMENT 1
inputs.exps.n_obs{1}=2; % Number of observed quantities
inputs.exps.obs_names{1}=char('Lum','mRNAa'); % Name of the observed quantities
inputs.exps.obs{1}=char('Lum=CL_m','mRNAa=CT_m'); % Observation function
inputs.exps.exp_y0{1}=[0 0 0 0 0 0]; % Initial conditions
inputs.exps.t_f{1}=120; % Experiment duration
inputs.exps.u_interp{1}='sustained'; % Stimulus definition
inputs.exps.t_con{1}=[0 120]; % Switching times: Initial and final time
inputs.exps.u{1}=[1]; % Value of the stimulus
inputs.exps.n_s{1}=15; % Number of sampling times
% in this case equidistant sampling times
% will be used

% EXPERIMENT 2
inputs.exps.n_obs{2}=2; % Number of observed quantities
inputs.exps.obs_names{2}=char('Lum','mRNAa'); % Name of the observed quantities
inputs.exps.obs{2}=char('Lum=CL_m','mRNAa=CT_m'); % Observation function
inputs.exps.exp_y0{2}=[0 0 0 0 0 0]; % Initial conditions
inputs.exps.t_f{2}=120; % Experiment duration
inputs.exps.u_interp{2}='pulse-down'; % Stimulus definition
inputs.exps.n_pulses{2}=5; % Number of pulses |-|-|-|-|-|-|-|
inputs.exps.t_con{2}=[0:12:120]; % Times of switching: initial, intermediate times
% and final

inputs.exps.u_min{2}=[0]; % Minimum value for the stimulus
inputs.exps.u_max{2}=[1]; % Maximum value for the stimulus
inputs.exps.n_s{2}=25; % Number of sampling times
% in this case equidistant sampling times
% will be used

%=====
% UNKNOWNNS RELATED DATA
%=====

% GLOBAL UNKNOWNNS to be considered in the rank

inputs.PEsol.id_global_theta=char('n1','n2','g1','m1','m4','m5','m6','m7','k1','k4','k5','k6',...
    'k7','p2','p3','r3','r4','q1','q2'); % 'all'|User selected
% Maximum, minimum and guess of parameter values to compute rank. Any other values can be selected.
inputs.PEsol.global_theta_max=50.*ones(1,19);
inputs.PEsol.global_theta_min=(1e-3).*ones(1,19);
inputs.PEsol.global_theta_guess=[7.5038 0.6801 1.4992 10.0982 2.3422 7.2482 1.8981 1.2 3.8045 ...
    2.5356 1.4420 4.8600 1.2 9.4440 0.5 0.4364 7.3021 4.5703 1.0];

% PLEASE MODIFY HERE IF YOU WANT TO INCLUDE OTHER GLOBAL OR LOCAL UNKNOWNNS
% GLOBAL INITIAL CONDITIONS
% inputs.PEsol.id_global_theta_y0='none'; % [] 'all'|User selected| 'none' (default)
% inputs.PEsol.global_theta_y0_max=[]; % Maximum allowed values for initial conditions
% inputs.PEsol.global_theta_y0_min=[]; % Minimum allowed values for initial conditions
% inputs.PEsol.global_theta_y0_guess=[]; % [] Initial guess
%
% LOCAL UNKNOWNNS (DIFFERENT VALUES FOR DIFFERENT EXPERIMENTS)
% inputs.PEsol.id_local_theta1='none'; % [] 'all'|User selected| 'none' (default)
% inputs.PEsol.local_theta_maxiexp=[]; % Maximum allowed values for the paramters
% inputs.PEsol.local_theta_miniexp=[]; % Minimum allowed values for the parameters
% inputs.PEsol.local_theta_guessiexp=[]; % [] Initial guess
% inputs.PEsol.id_local_theta_y01='none'; % [] 'all'|User selected| 'none' (default)
% inputs.PEsol.local_theta_y0_maxiexp=[]; % Maximum allowed values for initial conditions
% inputs.PEsol.local_theta_y0_miniexp=[]; % Minimum allowed values for initial conditions
% inputs.PEsol.local_theta_y0_guessiexp=[]; % [] Initial guess

```

A.2.1 Preprocessing the example: AMIGO_Prep('circadian_grank')

First of all the preprocessing is performed in order to generate FORTRAN code and .dll or .mexw32 necessary for simulation. This is a typical output:

```
*****
*      AMIGO, Copyright @CSIC      *
*      AMIGO_RC2d [09 Sept 2010]  *
*****

*Date: 07-Sep-2010

*Running AMIGO for: circadian_grank

----->Pre processing...this may take a few seconds.

----->Checking inputs...

-----> WARNING message

You have selected a charmodelF model type. But you have not specified a particular ODE solver.
To generate dlls (by default) radau5 is used.
If you want to use a different solver, please update your input file.

-----> WARNING message

You have selected a charmodelF model type. But you have not specified a particular SENS solver.
To generate dlls (by default) odessa is used.

-----> Generating Fortran ...

-----> Mexing files....

In file J:\ Almacen_Eva\ AMIGO_RC2d\ Kernel\ IVP_solvers\ radau5\ cradau5g.f:86

        CALL FCNCRADAU5(NLHS,PL,NRHS,PR,A3,A5,A7,A9,A10,A13,A15,A17,
                        1
In file J:\ Almacen_Eva\ AMIGO_RC2d\ Kernel\ IVP_solvers\ radau5\ cradau5g.f:92

        SUBROUTINE FCNCRADAU5(NLHS,PL,NRHS,PR,A3,A5,A7,A9,A10,A13,A15,
                        2
Warning (155): Inconsistent types (INTEGER(4)/REAL(8)) in actual argument lists at (1) and (2)
In file J:\ Almacen_Eva\ AMIGO_RC2d\ Kernel\ IVP_solvers\ odessa\ codessag.f:82

        CALL FCNCODESSA(NLHS,PL,NRHS,PR,A3,A4,A12,A13,A15,A18,A20,
                        1
In file J:\ Almacen_Eva\ AMIGO_RC2d\ Kernel\ IVP_solvers\ odessa\ codessag.f:88

        SUBROUTINE FCNCODESSA(NLHS,PL,NRHS,PR,A3,A4,A12,A13,A15,A18,A20,
                        2
Warning (155): Inconsistent types (INTEGER(4)/REAL(8)) in actual argument lists at (1) and (2)

----->Files generated...
```

Once files are generated all tasks may be performed.

A.2.2 Solving system dynamics: `AMIGO_Smodel('circadian_grank')`

To solve the system dynamics for the above mentioned experimental scheme and for the nominal values of the model unknowns, type:

```
>> AMIGO_Smodel('circadian_grank')
```

Together with the plots of evolution of the states with time for the different experiments, typically, the following will be displayed:

```
*****
*   AMIGO, Copyright @CSIC   *
*   AMIGO_RC2d [09 Sept 2010] *
*****

*Date: 07-Sep-2010

*Running AMIGO for: circadian_grank

----->Pre processing...this may take a few seconds.

----->Checking inputs...

-----> WARNING message

You have selected a charmodelF model type. But you have not specified a particular ODE solver.
To generate dlls (by default) radau5 is used.
If you want to use a different solver, please update your input file.

-----> WARNING message

You have selected a charmodelF model type. But you have not specified a particular SENS solver.
To generate dlls (by default) odessa is used.

----->Performing simulation for the given set of parameters and initial conditions

----->Plotting results...

----->Results (report and struct_results.mat) and plots were kept in the directory:

Results\ circadian-tutorial\ SModel_circadian_run1
```

Results will be kept in the folder `Results\circadian-tutorial\SModel_circadian_run1` as indicated in the last line of the output and will be organised as follows:

AMIGO Path\Results

circadian-tutorial

SModel_circadian_run1

AMIGO_gen_obs_circadian.m
fcn.f
sens.f
Files generated during preprocessing

circadian_smodel_input_run1.m
report_circadian_run1.m
states_plot_exp1.fig
states_plot_exp2.fig
strreport_circadian_run1.mat

The folder `circadian-tutorial` keeps:

- > fcn.f and sens.f the FORTRAN code generated during preprocessing
- > A .m file to compute observation function

The folder `SModel_circadian_run1` keeps:

- > A copy of the input file
- > A .m report with inputs and results
- > Two .fig files with the plots of the evolution of states with time for experiment 1 and 2 respectively.
- > A .mat file which keeps the inputs. and results. structures.

Figure A.3: Contents of folder Results\circadian-tutorial\SModel_circadian_run1

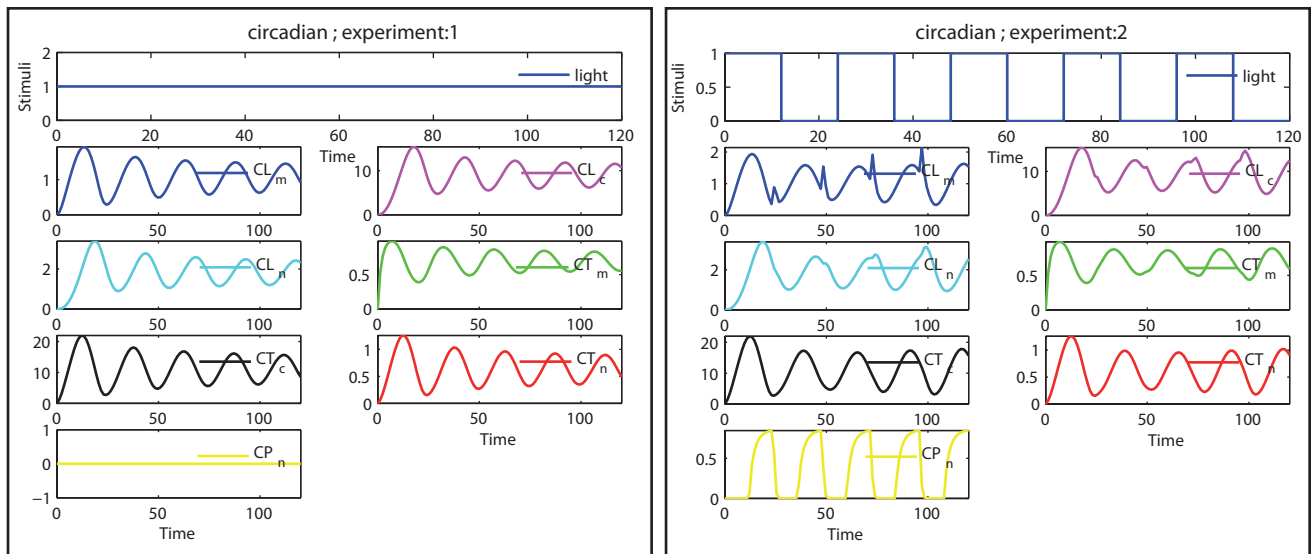


Figure A.4: The circadian clock in *Arabidopsis thaliana*: States evolution vs time. Results obtained for the nominal value of parameters and the experimental scheme described above.

The user may load inputs. and results. structures any time by typing:

```
>> load strreport_circadian_run1.mat
```

The information is organised as follows:

<pre>inputs. model.: [1x1 struct], structure that keeps all model related inputs exps.: [1x1 struct], structure that keeps experimental scheme and data ivpsol.: [1x1 struct], structure that keeps information related to IVP and sens solvers input_file.: 'circadian_grank' pathd.: [1x1 struct], structure that keeps AMIGO path</pre>	
<pre>results. pathd.: [1x1 struct], structure that keeps all paths and files names plotd.: [1x1 struct], structure that keeps information related to figures sim.: [1x1 struct], structure that keeps results of simulation</pre>	
<pre>results.sim. tsim: {[1x100 double] [1x100 double]}, states: {[100x7 double] [100x7 double]},</pre>	<pre>cell array of simulation times for experiments 1 and 2 cell array of states values vs time for experiments 1 and 2</pre>

A.2.3 Simulating the observables: `AMIGO_SObs('circadian_grank')`

To solve the system dynamics for the above mentioned experimental scheme and for the nominal values of the model unknowns, type:

```
>> AMIGO_SObs('circadian_grank')
```

Together with the plots of evolution of the observables with time for the different experiments, typically, the following will be displayed:

```
*****
*      AMIGO, Copyright @CSIC      *
*      AMIGO_RC2d [09 Sept 2010]    *
*****

*Date: 07-Sep-2010

*Running AMIGO for: circadian_grank

----->Pre processing...this may take a few seconds.

----->Checking inputs...

-----> WARNING message

You have selected a charmodelF model type. But you have not specified a particular ODE solver.
To generate dlls (by default) radau5 is used.
If you want to use a different solver, please update your input file.

-----> WARNING message

You have selected a charmodelF model type. But you have not specified a particular SENS solver.
To generate dlls (by default) odessa is used.

----->Performing simulation for the given set of parameters and initial conditions

----->Plotting results...

----->Results (report and struct_results.mat) and plots were kept in the directory:

Results\ circadian-tutorial\ SObs_circadian_run1
```

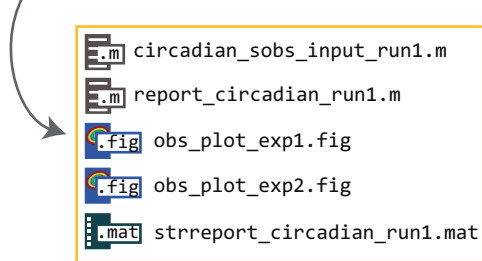
Results will be kept in the folder `Results\circadian-tutorial\SObs_circadian_run1` as indicated in the last line of the output and will be organised as follows:

AMIGO Path\Results

circadian-tutorial

SModel_circadian_run1

SObs_circadian_run1



The folder `SObs_circadian_run1` keeps:

- > A copy of the input file
- > A .m report with inputs and results
- > Two .fig files with the plots of the evolution of observables with time for experiment 1 and 2 respectively.
- > A .mat file which keeps the inputs. and results. structures.

Figure A.5: Contents of folder Results\circadian-tutorial\SObs_circadian_run1

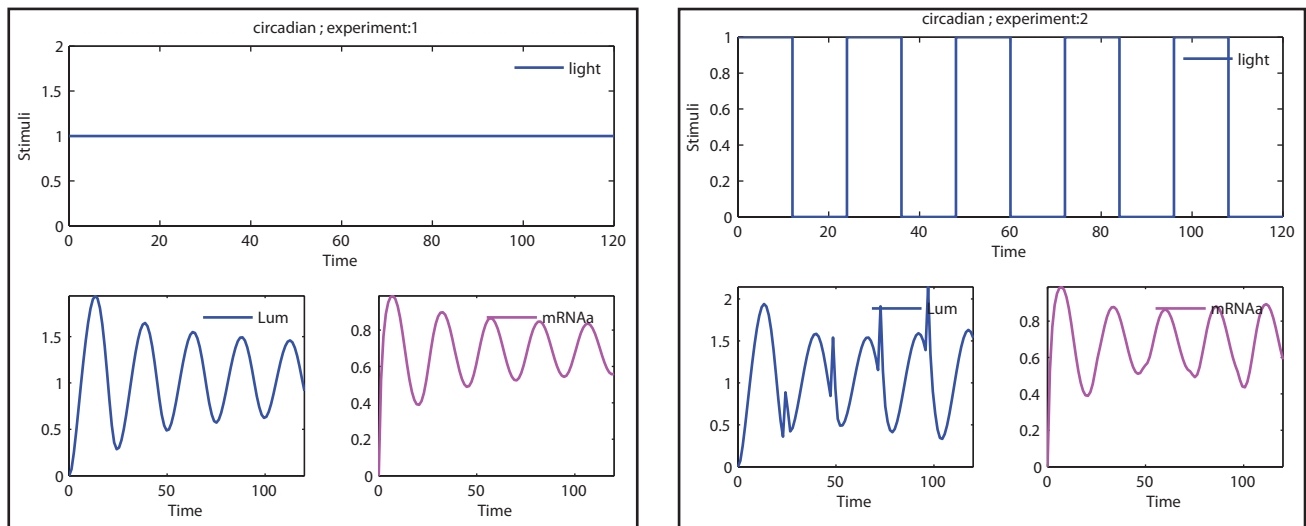


Figure A.6: The circadian clock in *Arabidopsis thaliana*: Observables evolution vs time. Results obtained for the nominal value of parameters and the experimental scheme described above.

```
>> load strreport_circadian_run1.mat
```

[illegible]

A.2.4 Performing the local rank of parameters: `AMIGO_LRank('circadian_grank')`

To perform the local sensitivity and ranking of parameters for the selected model parameters at the nominal value for the given experimental scheme type:

```
>> AMIGO_LRank('circadian_grank')
```

Several plots will be displayed with different measures of local sensitivities for the different experiments. In addition tabular results ranking the parameters will be displayed and saved in the MATLAB report.

Note that for the sake of brevity display of results is only partially shown:

```
.....

-----> RANKING for experiment: 1
-----> ABSOLUTE Ranking of model unknowns:
```

	par value	d_msqr	d_mabs	d_mean	d_max	d_min
r3	4.3640e-001	6.7880e-001	2.7111e+000	-1.2876e-001	5.9856e+000	-1.0350e+001
n2	6.8010e-001	5.6419e-001	2.3651e+000	7.3875e-001	8.1229e+000	-3.3630e+000
g1	1.4992e+000	1.9930e-001	7.9513e-001	3.6211e-002	3.0545e+000	-1.6807e+000
m4	2.3422e+000	1.7133e-001	7.5846e-001	-2.2719e-001	1.2857e+000	-2.2942e+000
k4	2.5356e+000	1.1523e-001	5.1319e-001	1.6648e-001	1.4822e+000	-8.6519e-001

```
.....

----->RELATIVE Ranking of model unknowns:
```

	par value	rd_msqr	rd_mabs	rd_mean	rd_max	rd_min
n1	7.5038e+000	5.8197e-001	2.3975e+000	-2.1391e-001	5.3477e+000	-9.0290e+000
m1	1.0098e+001	5.2957e-001	2.1709e+000	2.5133e-001	8.3699e+000	-4.5605e+000
m4	2.3422e+000	4.9860e-001	2.2389e+000	2.2913e-001	5.7671e+000	-3.7213e+000
n2	6.8010e-001	4.3181e-001	1.9627e+000	-1.5869e-001	3.8258e+000	-4.5262e+000
m5	7.2482e+000	3.7960e-001	1.7224e+000	4.2848e-001	3.4327e+000	-3.1264e+000

```
.....

-----> RANKING for experiment: 2
-----> ABSOLUTE Ranking of model unknowns:
```

	par value	d_msqr	d_mabs	d_mean	d_max	d_min
n2	6.8010e-001	4.6824e-001	2.4779e+000	8.8220e-001	7.0418e+000	-6.2723e+000
r3	4.3640e-001	4.4845e-001	2.4190e+000	9.2013e-002	7.5283e+000	-7.8309e+000
g1	1.4992e+000	1.3426e-001	7.2945e-001	-2.9488e-002	2.3824e+000	-2.2559e+000
m4	2.3422e+000	1.2826e-001	7.0475e-001	-2.4984e-001	1.5568e+000	-2.0057e+000
p3	5.0000e-001	1.2456e-001	6.0163e-001	6.5856e-004	2.6158e+000	-1.7480e+000

```
.....

----->RELATIVE Ranking of model unknowns:
```

	par value	rd_msqr	rd_mabs	rd_mean	rd_max	rd_min
n1	7.5038e+000	4.6482e-001	2.5015e+000	2.0959e-001	7.2658e+000	-8.6870e+000
m1	1.0098e+001	3.8919e-001	2.1101e+000	-1.3334e-001	7.7036e+000	-5.4960e+000
n2	6.8010e-001	3.3262e-001	1.8823e+000	1.9769e-001	4.1286e+000	-5.5183e+000
m4	2.3422e+000	3.0710e-001	1.8480e+000	-5.5889e-002	4.6986e+000	-3.1033e+000
p2	9.4440e+000	2.6963e-001	1.5128e+000	-2.8854e-001	3.1091e+000	-3.9021e+000

```
.....
```

-----> OVERALL RANKING

----->ABSOLUTE Ranking of GLOBAL model unknowns:

	par value	d_msqr	d_mabs	d_mean	d_max	d_min
r3	4.3640e-001	5.6363e-001	2.5650e+000	-1.8375e-002	7.5283e+000	-1.0350e+001
n2	6.8010e-001	5.1621e-001	2.4215e+000	8.1048e-001	8.1229e+000	-6.2723e+000
g1	1.4992e+000	1.6678e-001	7.6229e-001	3.3616e-003	3.0545e+000	-2.2559e+000
m4	2.3422e+000	1.4980e-001	7.3161e-001	-2.3852e-001	1.5568e+000	-2.2942e+000
k4	2.5356e+000	9.9699e-002	4.8915e-001	1.7256e-001	1.4822e+000	-9.2559e-001
k1	3.8045e+000	7.1623e-002	3.2490e-001	-9.2650e-004	9.2387e-001	-1.3652e+000
p3	5.0000e-001	6.2281e-002	3.0081e-001	3.2928e-004	2.6158e+000	-1.7480e+000
n1	7.5038e+000	5.9310e-002	2.7004e-001	-2.6398e-003	7.8694e-001	-1.1014e+000
m1	1.0098e+001	3.9347e-002	1.7735e-001	1.1878e-003	7.5873e-001	-4.9570e-001
m5	7.2482e+000	3.2666e-002	1.5972e-001	-1.6925e-002	3.5995e-001	-4.4488e-001
r4	7.3021e+000	3.1504e-002	1.4337e-001	9.9974e-004	5.8019e-001	-4.2204e-001
m7	1.2000e+000	3.1323e-002	1.5131e-001	-1.2908e-003	8.8149e-001	-1.2778e+000
p2	9.4440e+000	2.7763e-002	1.2989e-001	1.0929e-002	4.1393e-001	-3.1940e-001
k7	1.2000e+000	1.9575e-002	9.4613e-002	9.9477e-004	7.8443e-001	-5.5210e-001
k5	1.4420e+000	1.4941e-002	6.8333e-002	1.0401e-002	2.5470e-001	-9.5823e-002
q2	1.0000e+000	1.0028e-002	4.7955e-002	-1.7850e-003	2.8121e-001	-3.8812e-001
m6	1.8981e+000	7.1403e-003	3.4694e-002	-2.4569e-003	7.5661e-002	-1.0319e-001
q1	4.5703e+000	3.7283e-003	1.8023e-002	-4.1692e-005	1.5664e-001	-1.0475e-001
k6	4.8600e+000	2.2160e-003	1.0799e-002	8.4003e-004	3.1993e-002	-2.2381e-002

----->RELATIVE Ranking of GLOBAL model unknowns:

	par value	rd_msqr	rd_mabs	rd_mean	rd_max	rd_min
n1	7.5038e+000	5.2340e-001	2.4495e+000	-2.1617e-003	7.2658e+000	-9.0290e+000
m1	1.0098e+001	4.5938e-001	2.1405e+000	5.8992e-002	8.3699e+000	-5.4960e+000
m4	2.3422e+000	4.0285e-001	2.0434e+000	8.6618e-002	5.7671e+000	-3.7213e+000
n2	6.8010e-001	3.8222e-001	1.9225e+000	1.9500e-002	4.1286e+000	-5.5183e+000
k1	3.8045e+000	3.1838e-001	1.4892e+000	-6.6557e-003	4.0568e+000	-5.6741e+000
p2	9.4440e+000	3.0567e-001	1.5016e+000	-4.1727e-001	3.1091e+000	-3.9021e+000
m5	7.2482e+000	2.9966e-001	1.4880e+000	4.1681e-001	3.4399e+000	-3.1264e+000
g1	1.4992e+000	2.9814e-001	1.3953e+000	-3.3932e-002	5.0026e+000	-4.4081e+000
r3	4.3640e-001	2.9589e-001	1.3750e+000	5.7439e-002	4.4951e+000	-4.9341e+000
k4	2.5356e+000	2.8688e-001	1.4689e+000	-2.3383e-002	2.6028e+000	-4.0879e+000
r4	7.3021e+000	2.7571e-001	1.2830e+000	-4.4686e-002	4.6282e+000	-4.1443e+000
m7	1.2000e+000	5.4783e-002	2.4804e-001	4.1097e-002	2.6061e+000	-1.6869e+000
p3	5.0000e-001	4.5619e-002	2.0607e-001	-3.6620e-002	1.4389e+000	-2.1868e+000
k5	1.4420e+000	3.4281e-002	1.4153e-001	2.8239e-002	6.6779e-001	-2.5300e-001
k7	1.2000e+000	3.4189e-002	1.5495e-001	-2.5097e-002	1.0356e+000	-1.6231e+000
q1	4.5703e+000	2.5022e-002	1.1298e-001	-2.0619e-002	7.8761e-001	-1.2037e+000
m6	1.8981e+000	1.7405e-002	8.3969e-002	2.8142e-002	2.3163e-001	-1.3564e-001
q2	1.0000e+000	1.4310e-002	6.4809e-002	7.8099e-003	6.5561e-001	-4.5662e-001
k6	4.8600e+000	1.3695e-002	6.6611e-002	-2.1994e-002	1.0768e-001	-1.7460e-001

----->Plotting results....

----->Results (report and struct_results.mat) and plots were kept in the directory:

Results\ circadian-tutorial\ LRank_circadian_run1

Results will be kept in the folder `Results\circadian-tutorial\LRank_circadian_run1` as indicated in the last line of the output and will be organised as follows:

AMIGO Path\Results

circadian-tutorial

LRank_circadian_run1

SModel_circadian_run1

SObs_circadian_run1

 circadian_sobs_input_run1.m
 LRank_global_pars.fig
 report_circadian_run1.m
 sens_2D_lmsqr_exp1.fig
 sens_2D_lmsqr_exp2.fig
 sens_2D_rel_lmsqr_exp1.fig
 sens_2D_rel_lmsqr_exp2.fig
 sens_lmsqr_exp1.fig
 sens_lmsqr_exp2.fig
 sens_rel_lmsqr_exp1.fig
 sens_rel_lmsqr_exp2.fig
 strreport_circadian_run1.mat

The folder LRank_circadian_run1 keeps:

- > A copy of the input file
- > A .m report with inputs and results
- > Several .fig files with the plots of:
 - Local rank of parameters
 - 2D plots of the msqr and relative msqr measures of local sensitivities for experiments 1 and 2
 - Bar plots of the msqr and relative msqr measures of local sensitivities for experiments 1 and 2
- > A .mat file which keeps the inputs. and results. structures.

IMPORTANT remark: User may get more plots regarding other sensitivity measures: dmabs, dmean, dmax, and dmin and sensitivities evolution vs time, by allowing for 'full' display of results.

In any case user may access to complete results by loading the structure results.

Figure A.7: Contents of folder Results\circadian-tutorial\LRank_circadian_run1 for medium level display (results.plotd.plotlevel='medium')

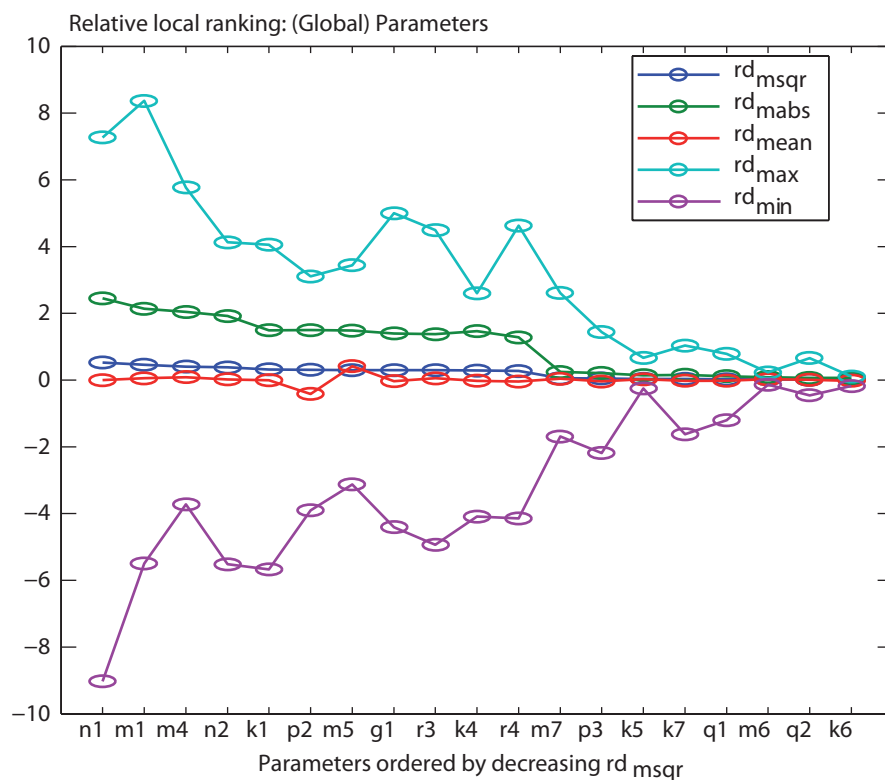


Figure A.8: Local relative rank of parameters. Results obtained for the nominal value of parameters and the experimental scheme described above.

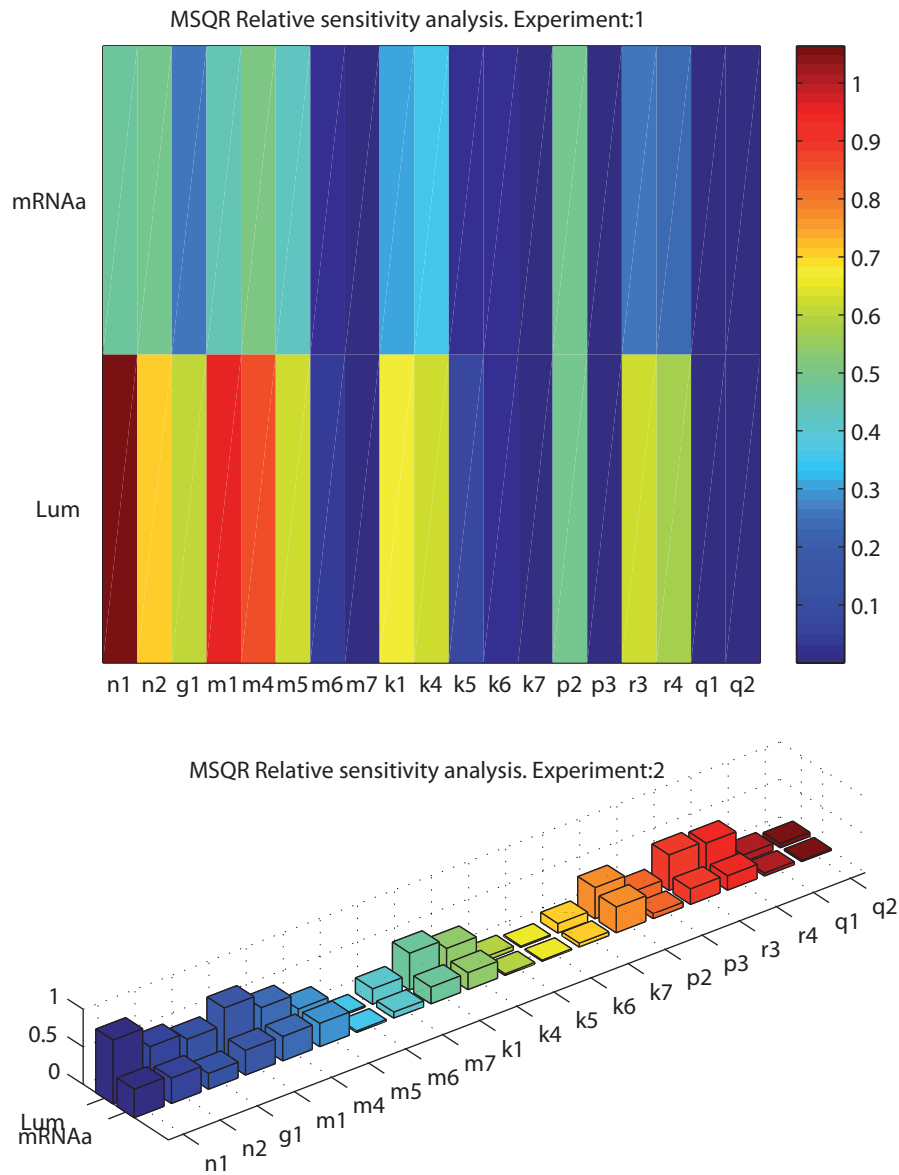


Figure A.9: Illustrative example of 2D and bar sensitivity plots for the circadian example. Results obtained for the nominal value of parameters and the experimental scheme described above. Note that plots correspond to different experiments.

Figures reveal that there are some parameters which are more clearly influencing the observables. Considering the two different observables and the two different experiments in the experimental scheme, it is clear that mRNA is less influenced by parameters than the luminiscence independently of the type of stimulation.

The user may load inputs, and results, structures any time by typing:

```
>> load strreport_circadian_run1.mat
```

The information is organised as follows:

```

inputs.
    model.: [1x1 struct], structure that keeps all model related inputs
    exps.: [1x1 struct], structure that keeps experimental scheme and data
    ivpsol.: [1x1 struct], structure that keeps information related to IVP and sens solvers
    PEsol.: [1x1 struct], structure that keeps information related to parameter estimation
            problem
    input_file.: 'circadian_grank'
    pathd.: [1x1 struct], structure that keeps AMIGO path

results.
    pathd.: [1x1 struct], structure that keeps all paths and files names
    plotd.: [1x1 struct], structure that keeps information related to figures
    rank.: [1x1 struct], structure that keeps results of sensitivity analysis and rank
           of unknowns

```

```

results.rank.

    number_int_errors: 0,                                number of integration errors

Results per observable per experiment
    sens_t: {[15x2x19 double] [25x2x19 double]} cell arrays of absolute & relative
    r_sens_t: {[15x2x19 double] [25x2x19 double]} sensitivities at sampling times

    d_obs_par_msqr: {[2x19 double] [2x19 double]} cell arrays of msqr, mabs and
    d_obs_par_mabs: {[2x19 double] [2x19 double]} mean sensitivity measures for
    d_obs_par_mean: {[2x19 double] [2x19 double]} unknown parameters and initial
    d_obs_y0_msqr: {[ ] [ ]} conditions
    d_obs_y0_mabs: {[ ] [ ]}
    d_obs_y0_mean: {[ ] [ ]}

    r_d_obs_par_msqr: {[2x19 double] [2x19 double]} cell arrays of relative msqr, mabs
    r_d_obs_par_mabs: {[2x19 double] [2x19 double]} and mean sensitivity measures
    r_d_obs_par_mean: {[2x19 double] [2x19 double]} for unknown parameters and initial
    r_d_obs_y0_msqr: {[ ] [ ]} conditions
    r_d_obs_y0_mabs: {[ ] [ ]}
    r_d_obs_y0_mean: {[ ] [ ]}

    d_obs_msqr: {[2x19 double] [2x19 double]} cell arrays of absolute
    d_obs_mabs: {[2x19 double] [2x19 double]} and relative msqr, mabs and mean
    d_obs_mean: {[2x19 double] [2x19 double]} sensitivity measures for all
    r_d_obs_msqr: {[ ] [ ]} unknowns
    r_d_obs_mabs: {[ ] [ ]}
    r_d_obs_mean: {[ ] [ ]}

Results per experiment
    rank_mat: {[19x5 double] [19x5 double]} cell arrays of absolute & relative
    r_rank_mat: {[19x5 double] [19x5 double]} msqr, mabs, mean, min and max
                                           sensitivity measures for all unknowns

    sorted_par_rank_mat: {[19x5 double] [19x5 double]} cell arrays of absolute & relative
    r_sorted_par_rank_mat: {[19x5 double] [19x5 double]} sorted by msqr sensitivity measures
    sorted_y0_rank_mat: {[0x5 double] [0x5 double]} for unknown parameters and
    r_sorted_y0_rank_mat: {[0x5 double] [0x5 double]} initial conditions

    par_rank_index: {[19x1 double] [19x1 double]} cell arrays of absolute and relative
    y0_rank_mat: {[0x1 double] [0x1 double]} rank of unknown parameters and
    r_par_rank_index: {[19x1 double] [19x1 double]} initial conditions
    r_y0_rank_index: {[0x1 double] [0x1 double]}

Overall results (combining all experiments and observables)
    sorted_over_par_rank_mat: [19x5 double] Matrices of absolute and relative
    r_sorted_over_par_rank_mat: [19x5 double] sorted by msqr sensitivity measures

    over_par_rank_index: [19x1 double] Overall absolute and relative
    r_over_par_rank_index: [19x1 double] rank of parameters

```

A.2.5 Performing the global rank of parameters: `AMIGO_GRank('circadian_grank')`

To perform the global sensitivity and ranking of the selected model parameters within the allowed bounds for the given experimental scheme type:

```
>> AMIGO_GRank('circadian_grank')
```

Several plots will be displayed with different measures of global sensitivities for the different experiments. In addition tabular results ranking the parameters will be displayed and saved in the MATLAB report.

Note that for the sake of brevity display of results is only partially shown:

```
.....

-----> GLOBAL RANKING

----->ABSOLUTE Ranking of GLOBAL unknown PARAMETERS:

      d_msqr      d_mabs      d_mean      d_max      d_min
-----
q2  3.4036e+003  1.0013e+003 -1.0013e+003  4.7835e-002  -5.2251e+003
m7  3.7911e+001  1.1880e+001 -1.1852e+001  7.3298e-002  -5.2929e+001
p3  3.6617e+001  1.1646e+001  1.1558e+001  4.9781e+001  -2.2257e-001
q1  2.5696e+001  8.2379e+000  8.1696e+000  3.4429e+001  -1.7272e-001
m1  1.7070e+001  5.6245e+000 -5.5169e+000  4.0300e-001  -2.4123e+001
p2  1.5037e+001  4.1310e+000  6.1998e-001  1.3970e+001  -1.1532e+001
n1  1.4737e+001  4.9459e+000  3.2732e+000  1.6670e+001  -4.7578e+000
m4  1.1385e+001  2.9734e+000 -2.9704e+000  2.8518e-002  -1.8951e+001
n2  1.0087e+001  2.7197e+000  2.7099e+000  1.6371e+001  -8.5225e-002
k4  5.8764e+000  1.5618e+000  1.5581e+000  9.5142e+000  -2.9879e-002
k7  5.6826e+000  1.8105e+000  1.7742e+000  7.6511e+000  -8.7071e-002
r3  4.8461e+000  1.6152e+000  1.0495e+000  5.5879e+000  -2.3977e+000
...

-----

----->RELATIVE Ranking of GLOBAL unknown PARAMETERS:

      rd_msqr      rd_mabs      rd_mean      rd_max      rd_min
-----
m4  3.5524e-001  1.1272e+000 -6.5660e-001  8.8769e-001  -1.1070e+001
n2  3.0343e-001  9.1493e-001  4.0635e-001  9.9604e+000  -9.4159e-001
m1  3.0148e-001  1.3479e+000  4.2024e-002  2.7333e+000  -6.5537e+000
k4  2.3236e-001  7.7168e-001  5.4317e-001  6.9486e+000  -4.6253e-001
n1  2.0933e-001  1.0422e+000 -3.8276e-001  2.2895e+000  -3.5819e+000
p3  2.0225e-001  9.1183e-001 -2.3555e-002  4.1427e+000  -1.4150e+000
p2  1.7297e-001  5.1537e-001  4.2948e-002  5.6742e+000  -8.6763e-001
m7  1.2433e-001  5.6639e-001  3.6093e-002  9.0616e-001  -2.4527e+000
q1  1.1350e-001  5.0802e-001 -4.5095e-003  2.3631e+000  -7.8046e-001
q2  1.1050e-001  4.9328e-001  2.1119e-003  7.5630e-001  -2.3128e+000
m6  1.0751e-001  2.9656e-001 -5.4841e-002  4.5016e-001  -3.7034e+000
k1  1.0587e-001  4.5263e-001 -1.4041e-003  1.7584e+000  -1.4175e+000
....

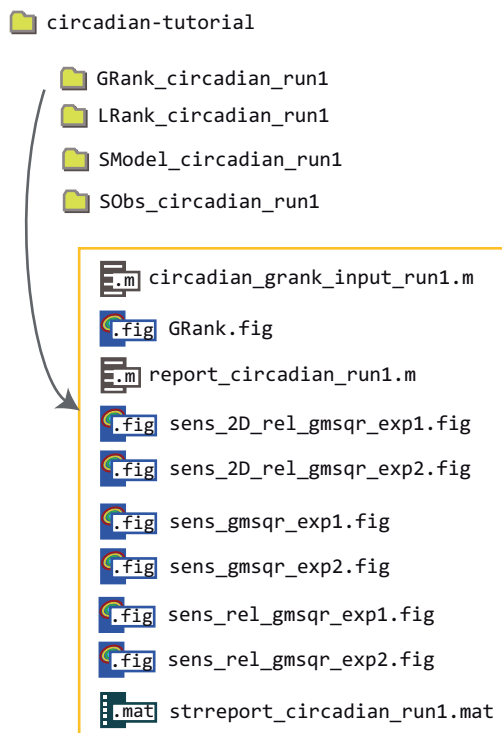
-----

----->Results (report and struct_results.mat) and plots were kept in the directory:

Results\ circadian-tutorial\ GRank_circadian_run1
```

Results will be kept in the folder `Results\circadian-tutorial\GRank_circadian_run1` as indicated in the last line of the output and will be organised as follows:

AMIGO Path\Results



The folder `GRank_circadian_run1` keeps:

- > A copy of the input file
- > A .m report with inputs and results
- > Several .fig files with the plots of:
 - Local rank of parameters
 - 2D plots of the msqr and relative msqr measures of local sensitivities for experiments 1 and 2
 - Bar plots of the msqr and relative msqr measures of local sensitivities for experiments 1 and 2
- > A .mat file which keeps the inputs. and results. structures.

IMPORTANT remark: User may get more plots regarding other sensitivity measures: dmabs, dmean, dmax, and dmin and sensitivities evolution vs time, by allowing for 'full' display of results. In any case user may access to complete results by loading the structure results.

Figure A.10: Contents of folder `Results\circadian-tutorial\GRank_circadian_run1` for medium level display (`results.plotd.plotlevel='medium'`)

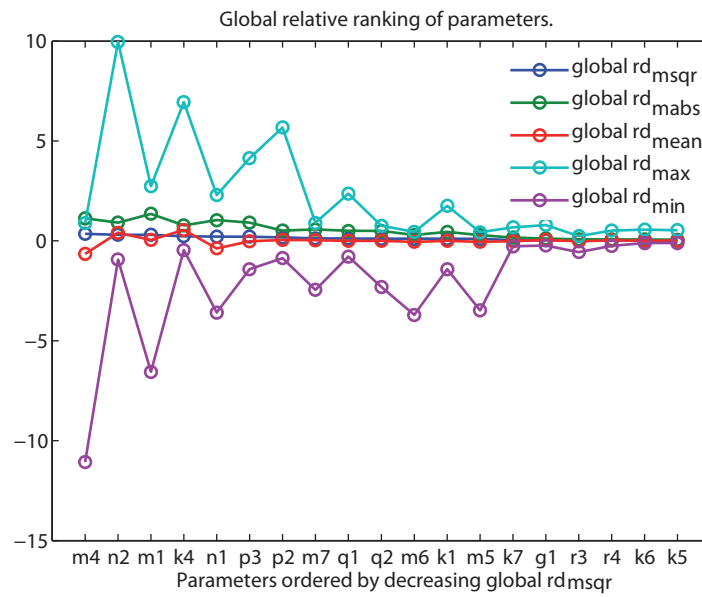


Figure A.11: Global relative rank of parameters. Results obtained for bounds $[1e-3, 100]$ for the parameters and the experimental scheme described above.

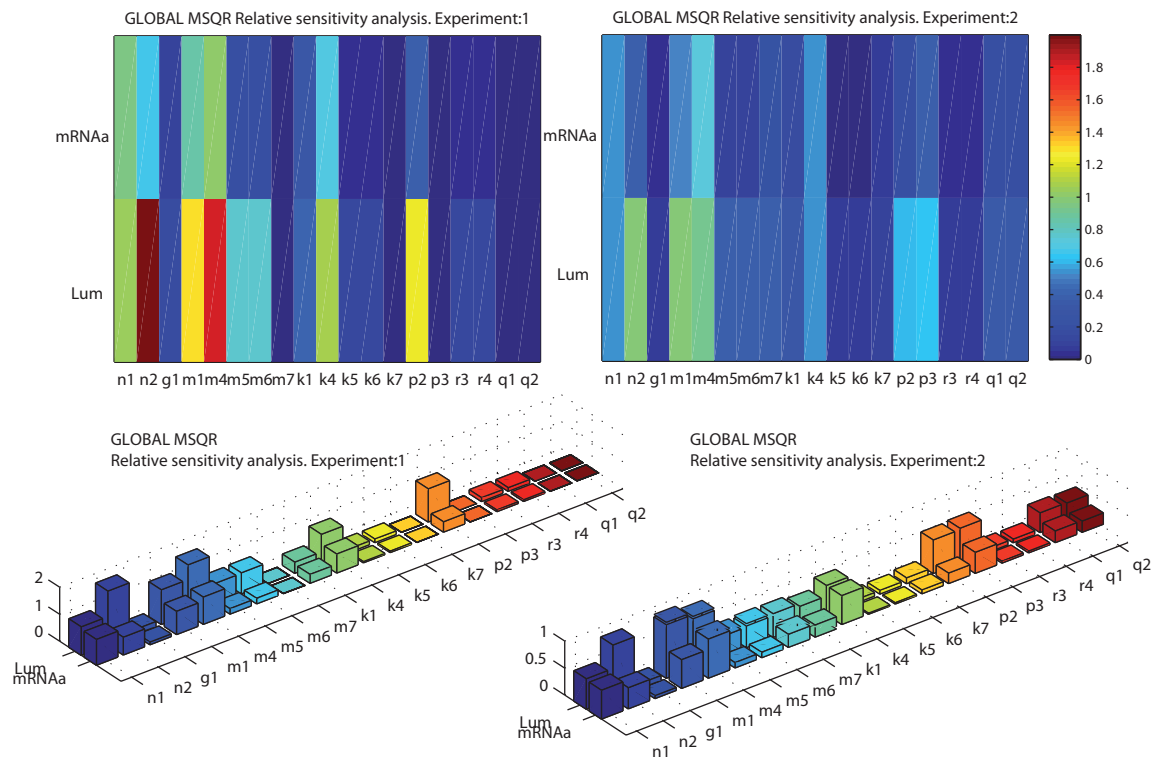


Figure A.12: Illustrative example of 2D and bar sensitivity plots for the circadian example. Results obtained for bounds $[1e-3, 100]$ for the parameters and the experimental scheme described above.

Figures reveal that there are some parameters which are more clearly influencing the observables. Considering the two different observables and the two different experiments in the experimental scheme, it is clear that measurements of mRNA are less informative for the purpose of parameter estimation than those of luminiscence independently of the type of stimulation, even though sustained experiment is more informative. From the results it is also expected poor or lack of identifiability for some of the parameters such as k_5 , k_6 , r_4 , r_3 , g_1 , k_7 , m_5 , k_1 .

The user may load inputs. and results. structures any time by typing:

```
>> load strreport_circadian_run1.mat
```

The information is organised as follows:

```
inputs.
    model.: [1x1 struct], structure that keeps all model related inputs
    exps.: [1x1 struct], structure that keeps experimental scheme and data
    ivpsol.: [1x1 struct], structure that keeps information related to IVP and sens solvers
    PEsol.: [1x1 struct], structure that keeps information related to parameter estimation
              problem
    input_file.: 'circadian_grank'
    pathd.: [1x1 struct], structure that keeps AMIGO path

results.
    pathd.: [1x1 struct], structure that keeps all paths and files names
    plotd.: [1x1 struct], structure that keeps information related to figures
    rank.: [1x1 struct], structure that keeps results of sensitivity analysis and rank
            of unknowns

results.rank.

    number_int_errors: 0,                number of integration errors
    n_global_samples: 10001,            number of samples for grank
    global_par_rank_index: [19x1 double], absolute rank of parameters
    r_global_par_rank_index: [19x1 double], relative rank of parameters

    global_par_rank_mat: [19x5 double]    matrices of absolute and relative msqr,
    r_global_par_rank_mat: [19x5 double], mabs, mean, max and min measures of
                                          rank for unknown parameters

    global_y0_rank_mat: [19x5 double]    matrices of absolute and relative msqr,
    r_global_y0_rank_mat: [19x5 double], mabs, mean, max and min measures of
                                          rank for unknown initial conditions

    global_y0_rank_mat: [0x5 double]     absolute and relative
    r_global_y0_rank_mat: [0x5 double],  rank of initial conditions

    g_d_obs_msqr_mat: {[2x19 double] [2x19 double]} cell arrays of global msqr, mabs
    g_d_obs_mabs_mat: {[2x19 double] [2x19 double]} and mean sensitivities per experiment
    g_d_obs_mean_mat: {[2x19 double] [2x19 double]}, per observable

    g_r_d_obs_msqr_mat: {[2x19 double] [2x19 double]} cell arrays of relative global
    g_r_d_obs_mabs_mat: {[2x19 double] [2x19 double]} msqr, mabs and mean sensitivities
    g_r_d_obs_mean_mat: {[2x19 double] [2x19 double]}, per experiment per observable
```

A.3 A model of the NF κ B module

A.3.1 Introduction

Mathematical models connected to experimental data have played a key role in revealing forms of regulation of NF- κ B signaling and the underlying molecular mechanisms. The model considered here was proposed by Lipniacki et al. [27]. This model involves two compartment kinetics of the activators IKK and NF- κ B, the inhibitors A20 and I κ B α and their complexes. It is assumed that IKK exists in any of these forms: neutral (IKK_n), active (IKK_a) or inactive (IKK_i). In the presence of the extracellular signal TNF, IKK is transformed into its phosphorylated form. In this form it is capable of phosphorylating I κ B α , and this leads to its degradation. In resting cells, the unphosphorylated I κ B α binds to NF- κ B and sequesters it in an inactive form in the cytoplasm. As a result, degradation of I κ B α releases the second activator, NF- κ B. The free NF- κ B enters the nucleus and upregulates transcription of the two inhibitors I κ B α and A20 and of a large number of other genes including the control gene cgen. The newly synthesized I κ B α again inhibits NF- κ B, while A20 inhibits IKK by catalyzing its transformation into another inactive form in which it is no longer capable of phosphorylating I κ B α .

The scheme of the pathway is:

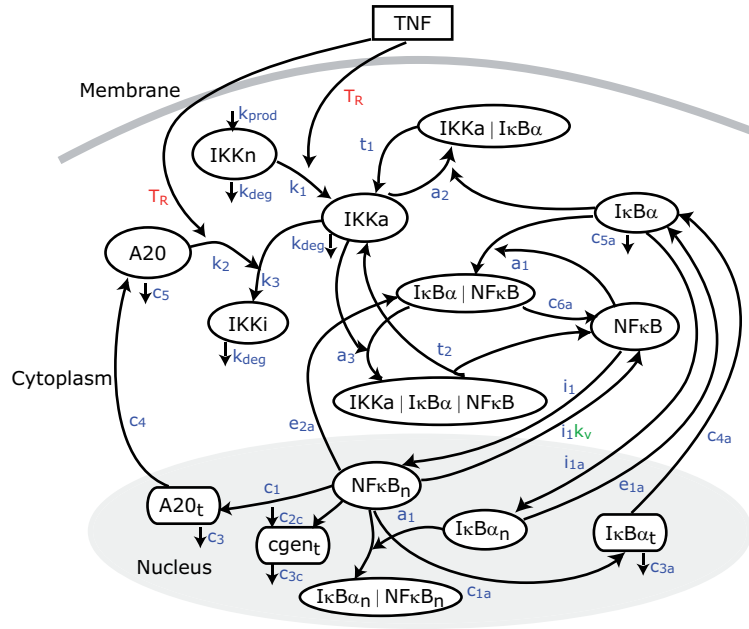


Figure A.13: Scheme of the NF κ B.

The corresponding mathematical model consists of 15 non-linear ordinary differential equations with 30 parameters as follows [27]:

$$\dot{\text{IKKn}} = k_{\text{prod}} - k_{\text{deg}}\text{IKKn} - T_R k_1 \text{IKKn}$$

$$\dot{\text{IKKa}} = T_R k_1 \text{IKKn} - k_3 \text{IKKa} - T_R k_2 \text{IKKa} \text{A20} - k_{\text{deg}} \text{IKKa} - a_2 \text{IKKa} \text{I}\kappa\text{B}\alpha + t_1 (\text{IKKa} | \text{I}\kappa\text{B}\alpha) - a_3 \text{IKKa} (\text{I}\kappa\text{B}\alpha | \text{NF}\kappa\text{B}) + t_2 (\text{IKKa} | \text{I}\kappa\text{B}\alpha | \text{NF}\kappa\text{B})$$

$$\dot{\text{IKKi}} = k_3 \text{IKKa} + T_R k_2 \text{IKKa} \text{A20} - k_{\text{deg}} \text{IKKi}$$

$$(\text{IKKa} | \text{I}\kappa\text{B}\alpha) = a_2 \text{IKKa} \text{I}\kappa\text{B}\alpha - t_1 (\text{IKKa} | \text{I}\kappa\text{B}\alpha)$$

$$\begin{aligned}
(\text{IKKa}|\text{I}\kappa\dot{\text{B}}\alpha|\text{NF}\kappa\text{B}) &= a_3\text{IKKa}(\text{IKKa}|\text{I}\kappa\text{B}\alpha) - t_2(\text{IKKa}|\text{I}\kappa\text{B}\alpha|\text{NF}\kappa\text{B}) \\
\text{NF}\kappa\text{B} &= c_{6a}(\text{I}\kappa\text{B}\alpha|\text{NF}\kappa\text{B}) - a_1\text{NF}\kappa\text{B} \text{I}\kappa\text{B}\alpha + t_2(\text{IKKa}|\text{I}\kappa\text{B}\alpha|\text{NF}\kappa\text{B}) - i_1\text{NF}\kappa\text{B} \\
\text{NF}\kappa\text{B}_n &= i_1k_v\text{NF}\kappa\text{B} - a_1\text{I}\kappa\text{B}\alpha_n \text{NF}\kappa\text{B}_n \\
\text{A}\dot{20} &= c_4\text{A}20_t - c_5\text{A}20 \\
\text{A}\dot{20}_t &= c_2 + c_1\text{NF}\kappa\text{B}_n - c_3\text{A}20_t \\
\text{I}\kappa\dot{\text{B}}\alpha &= -a_2\text{IKKa} \text{I}\kappa\text{B}\alpha - a_1\text{I}\kappa\text{B}\alpha \text{NF}\kappa\text{B} + c_{4a}\text{I}\kappa\text{B}\alpha_t - c_{5a}\text{I}\kappa\text{B}\alpha - i_{1a}\text{I}\kappa\text{B}\alpha + e_{1a}\text{I}\kappa\text{B}\alpha_n \\
\text{I}\kappa\dot{\text{B}}\alpha_n &= -a_1\text{I}\kappa\text{B}\alpha_n \text{NF}\kappa\text{B}_n + i_{1a}k_v\text{I}\kappa\text{B}\alpha - e_{1a}k_v\text{I}\kappa\text{B}\alpha_n \\
\text{I}\kappa\dot{\text{B}}\alpha_t &= c_{2a} + c_{1a}\text{NF}\kappa\text{B}_n - c_{3a}\text{I}\kappa\text{B}\alpha_t \\
(\text{I}\kappa\text{B}\alpha|\dot{\text{NF}}\kappa\text{B}) &= a_1\text{I}\kappa\text{B}\alpha \text{NF}\kappa\text{B} - c_{6a}(\text{I}\kappa\text{B}\alpha|\text{NF}\kappa\text{B}) - a_3\text{IKKa} (\text{I}\kappa\text{B}\alpha|\text{NF}\kappa\text{B}) + e_{2a}(\text{I}\kappa\text{B}\alpha_n|\text{NF}\kappa\text{B}_n) \\
(\text{I}\kappa\text{B}\alpha_n|\dot{\text{NF}}\kappa\text{B}_n) &= a_i\text{I}\kappa\text{B}\alpha_n \text{NF}\kappa\text{B}_n - e_{2a}k_v(\text{I}\kappa\text{B}\alpha_n|\text{NF}\kappa\text{B}_n) \\
c\dot{g}en_t &= c_{2c} + c_{1c}\text{NF}\kappa\text{B}_n - c_{3c}cgen_t
\end{aligned}$$

where IKKn represents the cytoplasmic concentration of neutral form of IKK; IKKa, the cytoplasmic concentration of active form of IKK; IKKi, the cytoplasmic concentration of inactive IKK; IκBα, the cytoplasmic concentration of IκBα; IκBα_n, the nuclear concentration of IκBα; IκBα_t, the concentration of IκBα mRNA transcripts calculated per cytoplasmic volume V; (IKKa|IκBα), the cytoplasmic concentration of complexes IKKa and IκBα, equivalent notation is used for all the complexes; T_R is a logical variable representing the presence or absence of signal; k_v is the ratio of cytoplasmic to nuclear volumes.

In their paper, Lipniacki et al. (2004) fixed some of the model parameters by using values from the literature. To fit the unknown parameters, they used experimental data from previous works by Lee et al. [25] and Hoffmann et al. [21]:

$$\boldsymbol{\theta} = [t_1, t_2, c_{3a}, c_{4a}, c_5, k_1, k_2, k_3, k_{prod}, k_{deg}, i_1, e_{2a}, i_{1a}]^T \quad (\text{A.3})$$

Lipniacki et al. concluded that several different sets of parameters are capable of reproducing the data. This lack of identifiability may originate either in the structure of the model and observables selected (lack of structural identifiability) or in the type of experiments performed and the experimental noise (lack of practical identifiability). Our aim was to determine the origin of the problem and to use the model identification loop presented here to improve the quality of the parameter estimates.

The structural identifiability analysis performed under the following conditions [2]:

- Only the concentrations measured by Lee et al.[25] and Hoffman et al. [21] are at our disposal.
- Initial conditions correspond to those for wild type cells after resting.
- The TNF stimulus is activated.
- Only the set $\boldsymbol{\theta}$ in Eqn. are considered all the other parameters are assumed to be fixed.

reveals that all parameters in the set $\boldsymbol{\theta}$ are structurally identifiable and may in principle be identified from experimental data.

A numerical example will be formulated here by generating pseudo-experimental data, subsequently solving the parameter estimation problem and performing the identifiability analysis. The experimental scheme available from Lee et al. [25] and Hoffmann et al. [21] will be considered.

Let's define the model and the experimental scheme as independent files that can be then called from different AMIGO input files:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% nfkb_model.m file
%
% The model considered in this work was proposed in:
% Lipniacki T, Paszek P, Brasier A, Luxon B, Kimmel M: Mathematical model of
% NFκB regulatory module. J Theor Biol 2004, 228:195-215.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% INPUT FILE TO GENERATE FOR ITS USE IN AMIGO
%
% > Paths related data
%
% > Model:          model_type; n_st; n_par; n_stimulus;
%                   st_names; par_names; stimulus_names;
%                   eqns; par
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%=====
% PATHS RELATED DATA
%=====

results.pathd.results_folder='NFκB'; % Folder to keep results (within Results)
results.pathd.short_name='NFκB';      % To identify figures and reports

%=====
% MODEL RELATED DATA
%=====

inputs.model.input_model_type='charmodelF';
inputs.model.n_st=15;                  % Number of states
inputs.model.n_par=29;                 % Number of model parameters
inputs.model.n_stimulus=1;            % Number of stimuli variables
inputs.model.st_names=char('IKKn','IKKa','IKKi','IKKaIkBa','IKKaIkBaNFkB','NFkB','NFkBn','A20',...
                           'A20t','IkBa','IkBan','IkBat','IkBaNFkB','IkBanNFkBn','cgent');
inputs.model.par_names=char('a1','a2','t1','a3','t2','c1a','c2a','c3a','c4a','c5a',...
                           'c6a','c1','c2','c3','c4','c5','k1','k2','k3','kprod','kdeg',...
                           'kv','i1','e2a','i1a','e1a','c1c','c2c','c3c');
inputs.model.stimulus_names=char('Tr'); % Names of the stimuli
inputs.model.eqns=char('dIKKn=kprod-kdeg*IKKn-Tr*k1*IKKn',...
                      'dIKKa=Tr*k1*IKKn-k3*IKKa-Tr*k2*IKKa*A20-kdeg*IKKa-a2*IKKa*IkBa+t1*IKKaIkBa-a3*IKKa*IkBaNFkB+
                      t2*IKKaIkBaNFkB',...
                      'dIKKi=k3*IKKa+Tr*k2*IKKa*A20-kdeg*IKKi',...
                      'dIKKaIkBa=a2*IKKa*IkBa-t1*IKKaIkBa',...
                      'dIKKaIkBaNFkB=a3*IKKa*IkBaNFkB-t2*IKKaIkBaNFkB',...
                      'dNFkB=c6a*IkBaNFkB-a1*NFkB*IkBa+t2*IKKaIkBaNFkB-i1*NFkB',...
                      'dNFkBn=i1*kv*NFkB-a1*IkBan*NFkBn',...
                      'dA20=c4*A20t-c5*A20',...
                      'dA20t=c2+c1*NFkBn-c3*A20t',...
                      'dIkBa=-a2*IKKa*IkBa-a1*IkBa*NFkB+c4a*IkBat-c5a*IkBa-i1a*IkBa+e1a*IkBan',...
                      'dIkBan=-a1*IkBan*NFkBn+i1a*kv*IkBa-e1a*kv*IkBan',...
                      'dIkBat=c2a+c1a*NFkBn-c3a*IkBat',...
                      'dIkBaNFkB=a1*IkBa*NFkB-c6a*IkBaNFkB-a3*IKKa*IkBaNFkB+e2a*IkBanNFkBn',...
                      'dIkBanNFkBn=a1*IkBan*NFkBn-e2a*kv*IkBanNFkBn',...
                      'dcgent=c2c+c1c*NFkBn-c3c*cgent');
inputs.model.par=[0.5 0.2 0.1 1 0.1 5e-7 0 4e-4 0.5 1e-4 2e-5 5e-7 0 4e-4 0.5 3e-4 2.5e-3 0.1 ...
                  1.5e-3 2.5e-5 1.25e-4 5 2.5e-3 0.01 0.001 5e-4 5e-7 0 4e-4]; % Nominal value for the parameters

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%nfkb_experimental_scheme.m
%
% The experimental scheme available from:
% > Lee E, Boone D, Chai S, Libby S, Chien M, Lodolce J, Ma A: Failure to
% regulate TNF-induced NF-kB and cell death responses in A20-deficient
% mice. Science 2000, 289:2350-2354.
% > Hoffmann A, Levchenko A, Scott M, Baltimore D: The I $\kappa$ B-NF-kB signaling
% module: temporal control and selective gene activation. Science 2002,
% 298:1241-1245.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% INPUT FILE FOR THE EXPERIMENTAL SCHEME
%
% > Experimental scheme: n_exp; exp_y0iexp; t_fiexp;
% u_interpiexp; t_coniexp; uiexp
% n_obsiexp; obs_namesiexp; obsiexp
% (AMIGO_SData)==> n_siexp; t_siexp;
% data_type; noise_type; std_deviexp
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%=====
% EXPERIMENTAL SCHEME RELATED DATA
%=====

inputs.exps.n_exp=2; % Number of experiments

% EXPERIMENT 1
inputs.exps.n_obs{1}=6; % Number of observed quantities
inputs.exps.obs_names{1}=char('NFkB_n','TIkB_a_c','A2OmRNA',...
'TIKK','IKK_a','IkBa_t'); % Name of the observed quantities
inputs.exps.obs{1}=char('NFkB_n=NFkBn','TIkB_a_c=IkBa+IkBaNFkB','A2OmRNA=A2Ot',...
'TIKK=IKKn+IKKa+IKKi','IKK_a=IKKa','IkBa_t=IkBat'); % Observation function
inputs.exps.exp_y0{1}=[0.2 0 0 0 0 3.155e-4 2.2958e-3 4.78285e-003 2.8697e-6 2.50663e-3...
3.43573e-3 2.86971e-6 0.06 7.888e-5 2.86971e-6]; % Initial conditions
inputs.exps.t_f{1}=3*3600; % Experiment duration
inputs.exps.u_interp{1}='sustained'; % Stimulus definition
inputs.exps.t_con{1}=[0 3*3600]; % Switching times: Initial and final time
inputs.exps.u{1}=[1]; % Value of the stimulus
inputs.exps.n_s{1}=12; % Number of sampling times
inputs.exps.t_s{1}=60.*[0 5 15 30 45 60 75 90 105 120 150 180]; % Sampling times

% EXPERIMENT 2
inputs.exps.n_obs{2}=6; % Number of observed quantities
inputs.exps.obs_names{2}=char('NFkB_n','TIkB_a_c','A2OmRNA',...
'TIKK','IKK_a','IkBa_t'); % Name of the observed quantities
inputs.exps.obs{2}=char('NFkB_n=NFkBn','TIkB_a_c=IkBa+IkBaNFkB','A2OmRNA=A2Ot',...
'TIKK=IKKn+IKKa+IKKi','IKK_a=IKKa','IkBa_t=IkBat'); % Observation function
inputs.exps.exp_y0{2}=[0.2 0 0 0 0 3.155e-4 2.2958e-3 4.78285e-003 2.8697e-6 2.50663e-3...
3.43573e-3 2.86971e-6 0.06 7.888e-5 2.86971e-6]; % Initial conditions
inputs.exps.t_f{2}=3*3600; % Experiment duration
inputs.exps.u_interp{2}='pulse-down'; % Stimulus definition
inputs.exps.n_pulses{2}=1; % Number of pulses |-|_
inputs.exps.t_con{2}=[0 180 3*3600]; % Times of switching
inputs.exps.u_min{2}=[0]; inputs.exps.u_max{2}=[1]; % Min/max value for the stimulus
inputs.exps.n_s{2}=12; % Number of sampling times
inputs.exps.t_s{2}=60.*[0 5 15 30 45 60 75 90 105 120 150 180]; % Sampling times

```

A.3.2 Generating pseudo-experimental data: AMIGO_SData('nfkb_psdata')

Here the input file to generate pseudo-experimental data is depicted:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TITLE: The NFkB module
%
% The model considered in this work was proposed in:
% Lipniacki T, Paszek P, Brasier A, Luxon B, Kimmel M: Mathematical model of
% NFkB regulatory module. J Theor Biol 2004, 228:195-215.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% INPUT FILE TO GENERATE PSEUDO-EXPERIMENTAL DATA
% This is the minimum input file to generate pseudo-experimental data
% Default values are assigned to non defined inputs.
% Minimum required inputs:
%   > Paths related data
%   > Model
%   > Experimental scheme: n_exp; exp_y0iexp; t_fiexp;
%                           u_interpiexp; t_coniexp; uiexp
%                           n_obsiexp; obs_namesiexp; obsiexp
%   (AMIGO_SData)==> n_siexp; t_siexp;
%                           data_type; noise_type; std_deviexp
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nfkb_model          % LOADS MODEL
nfkb_experimental_scheme % LOADS EXPERIMENTAL SCHEME

%=====
% EXPERIMENTAL DATA RELATED INFO
%=====
inputs.exps.data_type='pseudo_pos'; % Type of experimental data: 'real' | 'pseudo' | 'pseudo_pos'(>=0)
inputs.exps.noise_type='homo_var';  % Type of experimental noise: Gaussian with zero mean and
                                     % Homoscedastic with constant variance: 'homo'
                                     % Homoscedastic with varying variance: 'homo_var'
                                     % Heteroscedastic: 'hetero'

% EXPERIMENT 1
inputs.exps.std_dev{1}=0.10;% Standard deviation of the noise for each experiment: Ex: 0.10 <=> 10%
% EXPERIMENT 2
inputs.exps.std_dev{2}=0.10;% Standard deviation of the noise for each experiment: Ex: 0.10 <=> 10%

```

To generate pseudo-experimental data for the given experimental scheme type:

```

>> AMIGO_Prep('nfkb_psdata')
>> AMIGO_SData('nfkb_psdata')

```

Together with the plots of evolution of observables plus experimental data vs time a typical display will be as follows:

```

.....

-----> Calculating simulated experimental data for synthetic problems.
      Experimental noise being used:
      Homocedastic noise with varying variance.
      Maximum standard deviation:

*Experiment 1: 10.000000 (percent)
*Experiment 2: 10.000000 (percent)
-----

>>>   Generated experimental data for each experiment:

Experimental data 1:
inputs.exps.exp_data{1}=[
1.2669e-002  6.1880e-002  5.4264e-006  1.9070e-001  1.4943e-002  4.5282e-006
1.0421e-001  1.2293e-002  4.1026e-005  2.0745e-001  8.0195e-002  2.8359e-005
2.5579e-001  2.8736e-003  8.4920e-005  2.1439e-001  6.5629e-002  5.5768e-005
2.9200e-001  8.5916e-003  1.2035e-004  2.4198e-001  3.0735e-003  1.5975e-004
1.5901e-001  5.8807e-002  1.9135e-004  1.7245e-001  6.7410e-003  1.8668e-004
..... ];

Error data 1:
Standard deviation: 10.000000%
inputs.exps.error_data{1}=[
1.0373e-002  6.2696e-004  2.5567e-006  9.3011e-003  1.4943e-002  1.6585e-006
2.3188e-002  1.2133e-002  3.0723e-005  7.4205e-003  3.8519e-003  1.8056e-005
6.7458e-003  9.5396e-004  1.9642e-005  1.4568e-002  1.4234e-002  9.5102e-006
1.8157e-002  1.0648e-002  3.0481e-005  4.2250e-002  2.5942e-003  8.9116e-006
2.3392e-002  3.7096e-003  1.5172e-006  2.7151e-002  4.7805e-003  6.1912e-006
.... ];

Experimental data 2:
inputs.exps.exp_data{2}=[
2.7147e-002  7.2361e-002  1.1897e-005  2.0524e-001  5.6295e-005  1.6117e-005
7.1570e-002  4.0223e-003  1.0209e-005  2.2432e-001  4.9878e-002  2.8342e-006
2.4388e-001  1.1398e-002  5.9992e-005  1.9437e-001  1.7457e-002  6.8357e-005
2.4466e-001  3.4153e-002  1.7205e-004  1.9706e-001  6.2349e-003  1.4616e-004
1.3671e-001  6.1833e-002  2.2058e-004  1.7437e-001  2.9857e-004  2.0572e-004
.... ];

Error data 2:
Standard deviation: 10.000000%
inputs.exps.error_data{2}=[
2.4851e-002  9.8541e-003  9.0269e-006  5.2361e-003  5.6295e-005  1.3247e-005
5.5809e-002  3.8270e-003  9.3060e-008  2.4282e-002  4.0755e-006  7.4683e-006
1.6968e-002  8.0610e-003  5.1358e-006  5.4963e-003  1.2443e-003  3.2290e-006
1.8537e-002  1.1218e-002  2.3733e-005  2.6641e-003  1.9782e-003  2.1590e-006
2.6797e-002  1.3149e-003  3.4647e-005  2.5424e-002  7.1982e-004  1.9785e-005
.... ];

----->Plotting results....

----->Results (report and struct_results.mat) and plots were kept in the directory:
Results\ NFkB\ SData_NFkB_run1

```

Results will be kept in the folder `Results\NFkB\SData_NFkB_run1` as indicated in the last line of the output and will be organised as follows:

AMIGO Path\Results

NFkB

SData_NFkB_run1

data_plot_exp1.fig
 data_plot_exp2.fig
 NFkB_sdata_input_run1.m
 report_NFkB_run1.m
 strreport_run1.mat

The folder SData_NFkB_run1 keeps:

- > A copy of the input file
- > A .m report with inputs and results
- > Two .fig files with the plots of the evolution of observables together with the pseudo-experimental data vs time for experiment 1 and 2 respectively.
- > A .mat file which keeps the inputs. and results. structures.

Figure A.14: Contents of folder Results\NFkB\SData_NFkB_run1

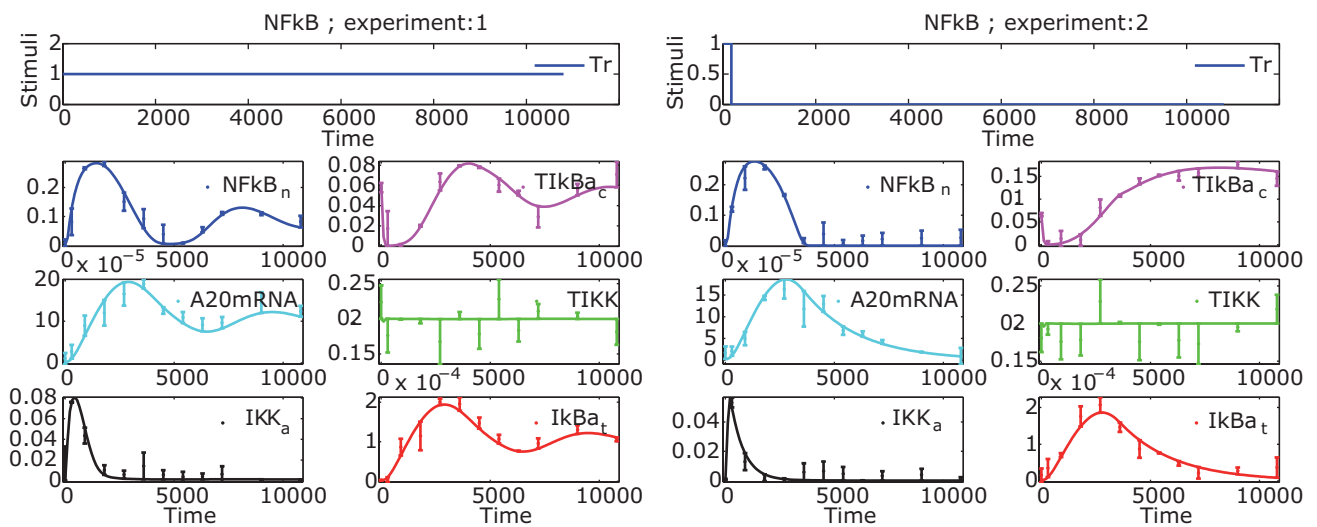


Figure A.15: The NF κ B module: Observables evolution and pseudo-experimental data vs time. Note: To generate pseudo-experimental data nominal value of parameters defined in inputs.model.par is being used.

The user may load inputs. and results. structures any time by typing:

```
>> load strreport_nfkb_run1.mat
```

The information is organised as follows:

[illegible]

A.3.3 Solving the parameter estimation problem: AMIGO_PE('nfkb_pe')

Now the problem of estimating the parameters of the model from the experimental data above is considered. With that aim the following `nfkb_pe` input file is generated:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TITLE: The NFkB module.          INPUT FILE TO ESTIMATE MODEL UNKNOWNNS
%      (AMIGO_PE)==>>  data_type; noise_type; exp_data{iexp}; [error_data{iexp}]
%                        id_global_theta; [id_global_theta_y0]
%                        [id_local_theta{iexp}];[id_local_theta_y0{iexp}]
%                        global_theta_max; global_theta_min; [global_theta_guess];
%                        [global_theta_y0_max];[global_theta_y0_min]; [global_theta_y0_guess];
%                        [local_theta_max{iexp}];[local_theta_min{iexp}]; [local_theta_guess{iexp}]
%                        [PEcost_type];[lsq_type];[llk_type]
%                        []:optional inputs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nfkb_model                % LOADS MODEL
nfkb_experimental_scheme   % LOADS EXPERIMENTAL SCHEME
%=====
% EXPERIMENTAL DATA RELATED INFO
%=====
inputs.exps.data_type='real';      % Type of experimental data: 'real'
inputs.exps.noise_type='homo_var'; % Gaussian Homoscedastic with varying variance: 'homo_var'

% EXPERIMENT 1 : COPY MATRICES FROM RESULTS OF AMIGO_SData('nfkb_psdata')
inputs.exps.exp_data{1}=...
[ 1.2669e-002  6.1880e-002  5.4264e-006  1.9070e-001  1.4943e-002  4.5282e-006
  1.0421e-001  1.2293e-002  4.1026e-005  2.0745e-001  8.0195e-002  2.8359e-005
  2.5579e-001  2.8736e-003  8.4920e-005  2.1439e-001  6.5629e-002  5.5768e-005
  2.9200e-001  8.5916e-003  1.2035e-004  2.4198e-001  3.0735e-003  1.5975e-004
  1.5901e-001  5.8807e-002  1.9135e-004  1.7245e-001  6.7410e-003  1.8668e-004
  2.7352e-002  7.1891e-002  1.6524e-004  1.7904e-001  2.4773e-004  2.0223e-004
  3.4334e-003  7.9096e-002  1.1346e-004  2.2031e-001  1.0285e-002  1.2105e-004
  7.8331e-003  6.2006e-002  7.1724e-005  1.9185e-001  7.3342e-003  5.0684e-005
  5.8919e-002  5.4425e-002  8.1396e-005  1.7215e-001  6.1064e-003  5.2071e-005
  8.9897e-002  4.3884e-002  7.4471e-005  2.0622e-001  3.2625e-003  1.0311e-004
  1.1045e-001  4.8631e-002  1.2085e-004  2.3096e-001  5.5002e-003  1.1759e-004
  5.0787e-002  4.2017e-002  1.0379e-004  2.1401e-001  7.1844e-003  1.1820e-004];
inputs.exps.error_data{1}=...
[ 1.0373e-002  6.2696e-004  2.5567e-006  9.3011e-003  1.4943e-002  1.6585e-006
  2.3188e-002  1.2133e-002  3.0723e-005  7.4205e-003  3.8519e-003  1.8056e-005
  6.7458e-003  9.5396e-004  1.9642e-005  1.4568e-002  1.4234e-002  9.5102e-006
  1.8157e-002  1.0648e-002  3.0481e-005  4.2250e-002  2.5942e-003  8.9116e-006
  2.3392e-002  3.7096e-003  1.5172e-006  2.7151e-002  4.7805e-003  6.1912e-006
  3.2897e-002  7.1411e-003  1.3147e-005  2.0456e-002  1.1163e-003  2.3850e-005
  3.2826e-003  2.8356e-004  1.9761e-005  2.0760e-002  9.0867e-003  1.2176e-005
  1.7880e-003  4.9595e-003  2.3807e-005  7.7973e-003  6.1469e-003  4.4847e-005
  1.3290e-002  4.2314e-003  5.5702e-006  2.7630e-002  4.8499e-003  2.3755e-005
  1.6307e-002  4.3695e-003  8.2800e-006  6.3119e-003  1.9336e-003  2.0360e-005
  4.0986e-003  1.6098e-003  1.0762e-006  3.1070e-002  4.2079e-003  2.1838e-006
  1.1905e-002  1.6236e-002  7.0952e-006  1.4160e-002  5.9722e-003  7.3141e-006];

% EXPERIMENT 2 : LOADS DATA FROM RESULTS OF AMIGO_SData('nfkb_psdata')
temp=load(strcat(pwd,'\ Results\ NFkB\ SData_NFkB_run1\ strreport_NFkB_run1.mat'),'results');
inputs.exps.exp_data2=temp.results.sim.exp_data2;
inputs.exps.error_data2=temp.results.sim.error_data2;
clear temp;
```

```

%=====
% UNKNOWNNS RELATED DATA
%=====
inputs.PEsol.id_global_theta=char('t1','t2','c3a','c4a','c5','k1','k2','k3',...
                                   'kprod','kdeg','i1','e2a','i1a');
inputs.PEsol.global_theta_max=1.*ones(1,13);      % Maximum allowed values for the parameters
inputs.PEsol.global_theta_min=1e-10.*zeros(1,13); % Minimum allowed values for the parameters

%=====
% COST FUNCTION RELATED DATA
%=====
inputs.PEsol.PEcost_type='llk';%> 'lsq' (weighted least squares default)
                                %> 'llk' (log likelihood)

inputs.PEsol.llk_type='homo_var';% [] To be defined for llk function:
                                %> 'homo': all data weighted in iexp weighted by the given
                                    constant variance
                                %> 'homo_var': data weighted taking into account error_data{iexp}
                                %> 'hetero': standard deviation assumed to be linearly dependent
                                    on the observable

```

In a first approximation to the problem the range for the parameters is selected to be $[1e - 10, 1]$, and the initial guess the default mean value. The problem is solved by means of local methods as follows:

```

>> AMIGO_PE('nfkb','rb1','local_dn2fb')    Tip: The run identifier gives a clue
                                             of the maximum bound used for the unknowns
>> AMIGO_PE('nfkb','rb1','local_fmincon')

```

The fits obtained in both cases are poor, with mean relative residuals up to 300%. To asses whether the problem is multimodal, a multistart of local solvers, with 200 starts from different initial guesses within the bounds, is used:

```

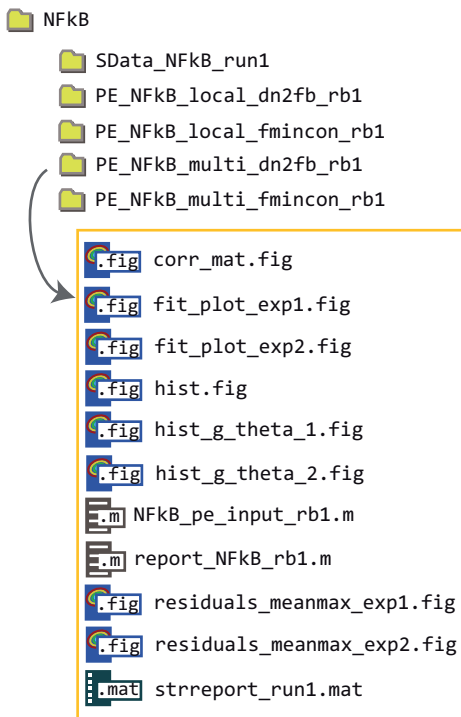
>> AMIGO_PE('nfkb','rb1','multi_dn2fb')
>> AMIGO_PE('nfkb','rb1','multi_fmincon')

```

The best solution obtained for the multistart of dn2fb corresponds to $llk = 66.88$ obtained in 1312s and the best found by the multistart of fmincon was $llk = 424.76$ in 898s. This already reveals certain multimodality.

Let's consider the results obtained by 'multi_dn2fb' in more detail. Results will be kept in folder `Results\NFkB\PE_NFkB_multi_dn2fb_rb5` which is organised as follows:

AMIGO Path\Results



The folder `PE_NFkB_multi_dn2fb_rb5` keeps:

- > A copy of the input file
- > A .m report with inputs and results
- > Two .fig files with the best fits for experiments 1 and 2
- > Several .fig files to keep histograms of values obtained for the cost function and the unknowns for the different starts
- > Two .fig files with mean and maximum residuals per observable for experiments 1 and 2
- > A .mat file which keeps the inputs. and results. structures.

Figure A.16: Contents of folder Results\NFkB\PE_NFkB_multi_dn2fb_rb5

The user may load inputs. and results. structures any time by typing:

```
>> load strreport_nfkb_rb1.mat
```

The information is organised as follows:

```
inputs.
    model.: [1x1 struct], structure that keeps all model related inputs
    exps.: [1x1 struct], structure that keeps experimental scheme and data
    ivpsol.: [1x1 struct], structure that keeps information related to IVP and sens solvers
    PEsol.: [1x1 struct], structure that keeps information related to the
              parameter estimation problem
    nlpsol.: [1x1 struct], structure that keeps information related to the NLP solver
    input_file.: 'nfkb_pe'
    pathd.: [1x1 struct], structure that keeps AMIGO path

results.
    pathd.: [1x1 struct], structure that keeps all paths and files names
    plotd.: [1x1 struct], structure that keeps information related to figures
    sim.: [1x1 struct], structure that keeps results of simulation
    nlpsol.: [1x1 struct], structure that keeps results and statistics of the NLP problem
    fit.: [1x1 struct], structure that keeps all results related to the
            best fit
```

```

results.sim.
    tsim: {[1x100 double] [1x100 double]}, cell array of simulation times
                                                for experiments 1 and 2
    states: {[100x15 double] [100x15 double]}, cell array of states values vs
                                                time for experiments 1 and 2 at optimum
    obs: {[100x6 double] [100x6 double]}, cell array of observables vs time
                                                for experiments 1 and 2 at optimum

results.nlpsol.
    fbest: 66.88, cost function at the optimum
    vbest: [1x13 double], vector of optimum unknown values
func_vector_multistart: [200x1 double], vector of best cost functions achieved for every start
    v_vector_multistart: [200x13 double], matrix of unknowns best values achieved at each start
    cpu_time: 1311.7, Computational cost

results.fit.
    residuals: {[12x6 double] [12x6 double]}, cell array of residuals at the optimum
                                                per sampling time per observable
                                                for experiments 1 and 2
    rel_residuals: {[12x6 double] [12x6 double]}, cell array of relative residuals at
                                                the optimum per sampling time per
                                                observable for experiments 1 and 2
    ms: {[12x6 double] [12x6 double]}, cell array of observables at the optimum
                                                per sampling time per observable for
                                                experiments 1 and 2
    g_FIM: [13x13 double], Fisher Information Matrix at optimum
    g_corr_mat: [14x14 double], Correlation Matrix at optimum
    g_var_cov_mat: [13x13 double], Variance-Covariance Matrix at optimum
    thetabest: [1x13 double], Vector of best parameter values
    fbest: 66.88, Cost function at the optimum
    cpu_time: 1311.7, Computational cost

```

Having a look at the histograms corresponding to the cost function and the parameter values achieved by the multistart it can be easily observed that: a) there is a clear distribution on the solutions, b) there is a clear tendency to converge to the bounds and c) there is more distribution on the parameter values than on the cost function. These lead us to conclude that the problem is multimodal but also poorly identifiable.

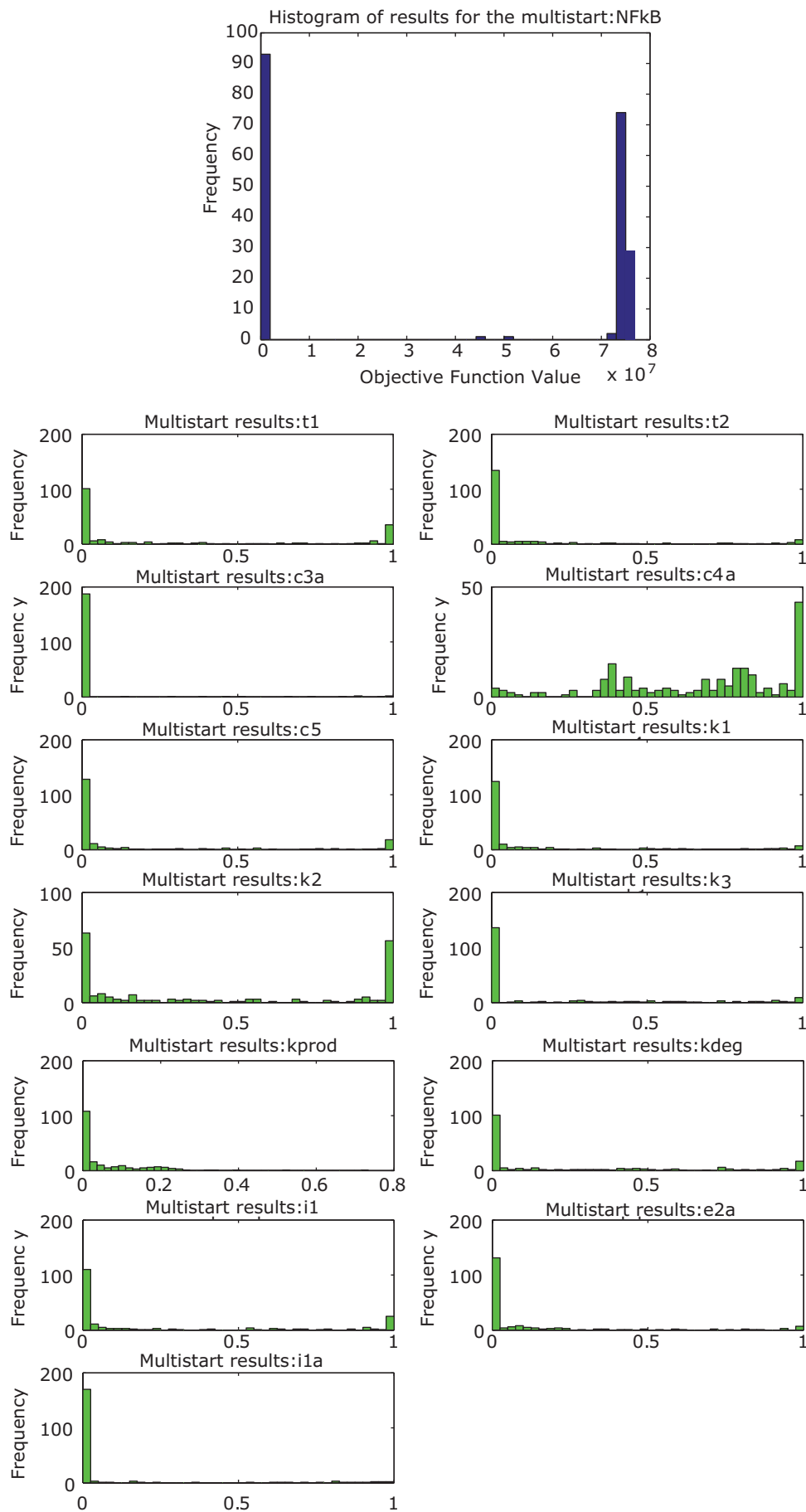


Figure A.17: The NF κ B module: Solutions of the parameter estimation problem with the multistart of dn2f. Histograms of solutions achieved.

Solving the problem with global optimization methods:

```
>> AMIGO_PE('nfkb_pe','rb1','de')           %Solve with de
>> AMIGO_PE('nfkb_pe','rb1','sres')          %Solve with sres
```

and sequential hybrid and metaheuristics:

```
>> AMIGO_PE('nfkb_pe','rb1')                 %Solve with ssm (default)
>> AMIGO_PE('nfkb_pe','rb1','fssm')           %Solve with fssm
>> AMIGO_PE('nfkb_pe','rb1','hyb_de_dn2fb')    %Solve with a sequential hybrid: de-dn2fb
>> AMIGO_PE('nfkb_pe','rb1','hyb_sres_fmincon') %Solve with a sequential hybrid: sres-fmincon
>> AMIGO_PE('nfkb_pe','rb1','globalm')         %Solve with globalm
```

Note that it is necessary to modify the defaults of the different optimizers to solve the problem. This may be done by editing `ssm_options`, `de_options`, `sres_options` and `globalm_options`.

The sequential hybrid and the metaheuristics were able to solve the problem in very reasonable computational costs. For example the hybrid of DE and dn2fb reported the global optimum in 196 s and fssm in 233 s.

Following figures show illustrative examples of the best fit and the mean and maximum /residuals per observable and experiment as depicted by the toolbox:

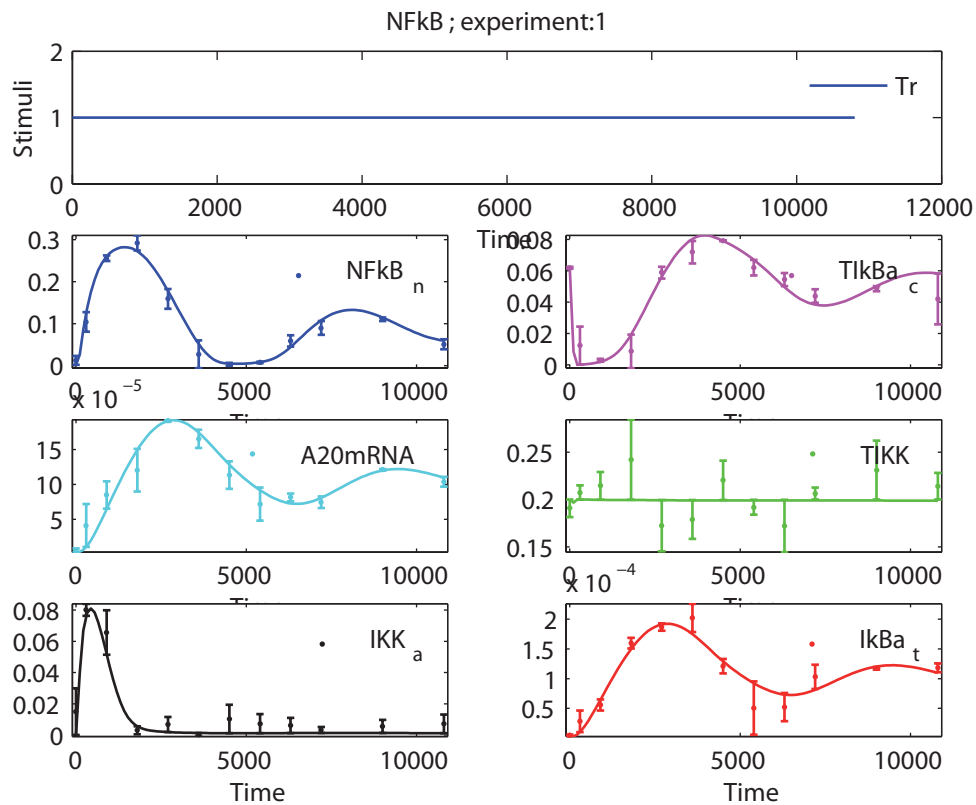


Figure A.18: NF κ B module: Best fit for the experiment 1.

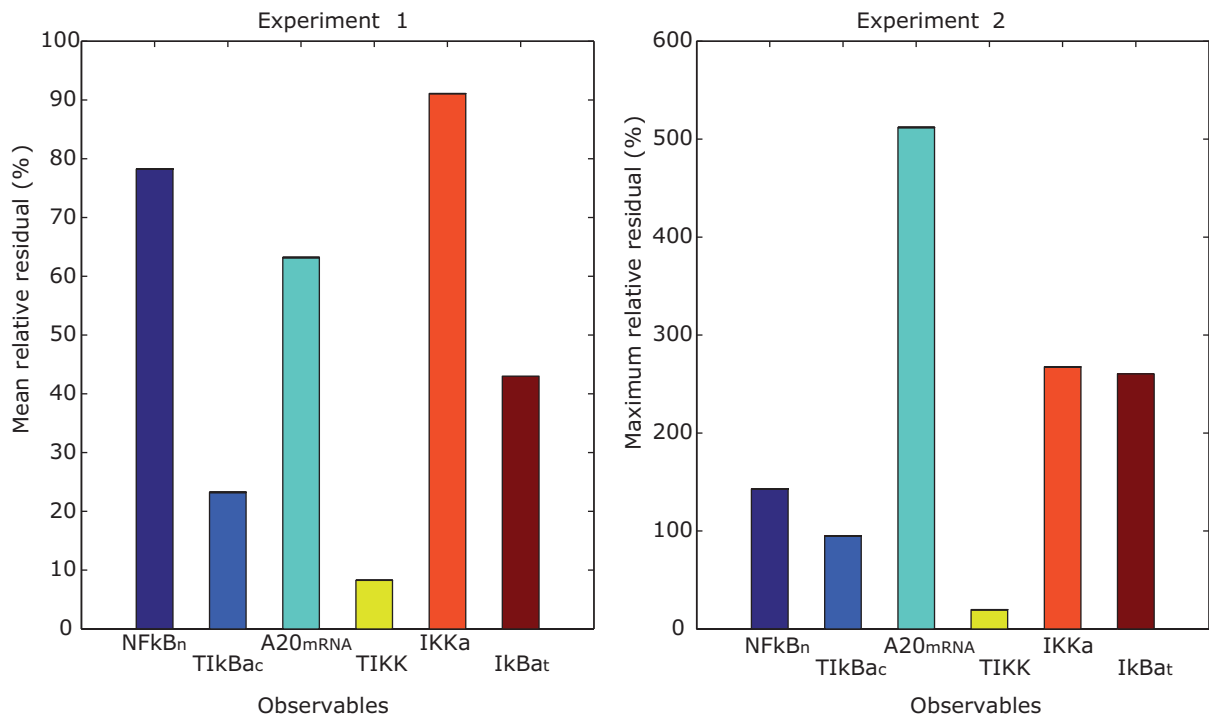


Figure A.19: $\text{NF}\kappa\text{B}$ module: Mean and maximum relative residuals corresponding to the best fit for experiment 2.

The global solution $\text{llk}=66.84$ (never found by the multistart) corresponded to the following values for the parameters, which are very closed to the ‘real’ ones:

t1	:	1.0157e-001	+-	4.4262e-001;
t2	:	1.1696e-001	+-	8.5194e-002;
c3a	:	3.9791e-004	+-	6.8682e-006;
c4a	:	5.0996e-001	+-	3.0064e-002;
c5	:	3.2561e-004	+-	4.9463e-005;
k1	:	2.4896e-003	+-	7.8444e-005;
k2	:	1.0197e-001	+-	3.5143e-002;
k3	:	1.4730e-003	+-	7.2185e-005;
kprod	:	2.3831e-005	+-	7.5581e-006;
kdeg	:	1.2004e-004	+-	3.8315e-005;
i1	:	2.4147e-003	+-	2.1806e-004;
e2a	:	8.9010e-003	+-	6.8265e-002;
i1a	:	1.0283e-003	+-	1.6208e-004;

However the confidence regions for some of them are significant, this is particularly true for $t1$, $t2$ or $e2a$, reflecting some practical identifiability problems. The correlation matrix confirms that some pairs of parameters are highly correlated:

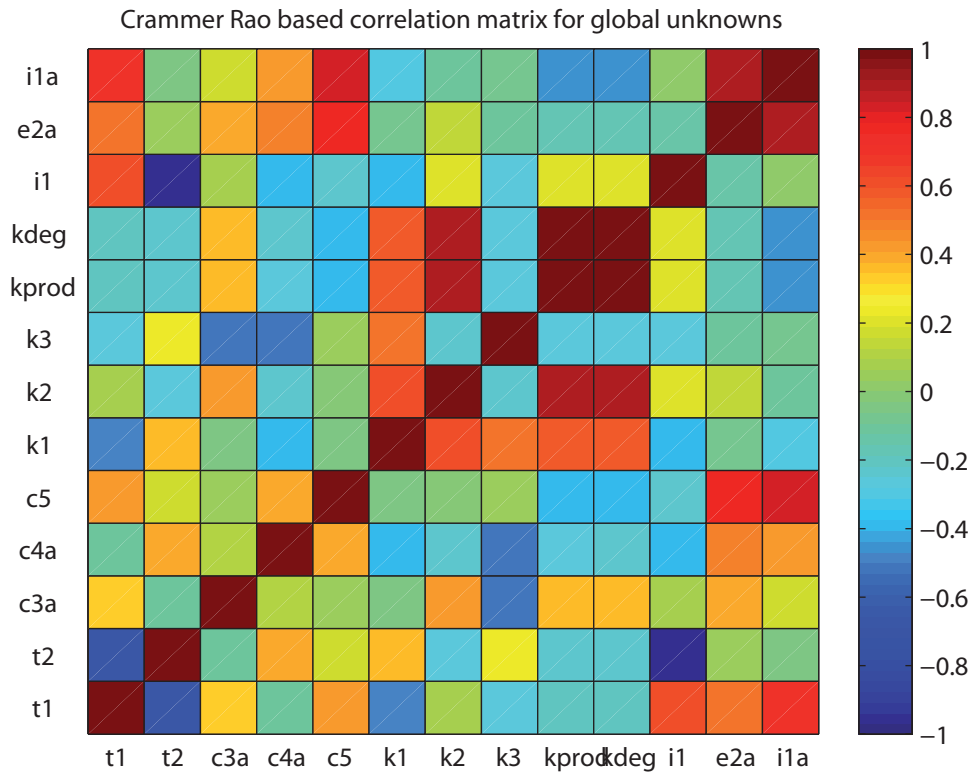


Figure A.20: NF κ B module: Cramer-Rao based correlation matrix corresponding to the best fit.

For example the pairs: k_{deg} and k_{prod} or i_{1a} and e_{2a} report a correlation value larger than 0.95; t_1 and t_2 are also significantly correlated and they are also correlated with i_{1a} and e_{2a} .

A.3.4 Performing the identifiability analysis: AMIGO_ContourP('nfkb_pe')

To analyse the practical identifiability in the vicinity of the global solution the contour plots of the log-likelihood function may be plotted by pairs of parameters. To do so take the best solution for the parameters and take it as initial guess in the `nfkb_pe` file and use it as a reference to generate the bounds (a maximum of 100% error in the estimation of the parameters can be assumed). This has been implemented in the file `nfkb_ident` as follows:

```
%=====
% UNKNOWNNS RELATED DATA
%=====
inputs.PEsol.global_theta_max=[1.0157e-001 1.1696e-001 3.9791e-004 5.0996e-001 3.2561e-004 ...
2.4896e-003 1.0197e-001 1.473e-003 2.3831e-005 1.2004e-004 2.4147e-003 8.901e-003 1.0283e-003 ];
inputs.PEsol.global_theta_max=2.*inputs.PEsol.global_theta_guess; % Maximum allowed values for the
                                                                    parameters
inputs.PEsol.global_theta_min=0.5.*inputs.PEsol.global_theta_guess; % Minimum allowed values for the
                                                                    parameters
```

Type:

```
>> AMIGO_ContourP('nfkb_ident','opt') %To draw contour plots in the vicinity of the optimum
```

For this task plots (.fig files) will be kept in the folder: `Results\NFkB \Contours_NFkB_opt`. Here some illustrative examples are shown:

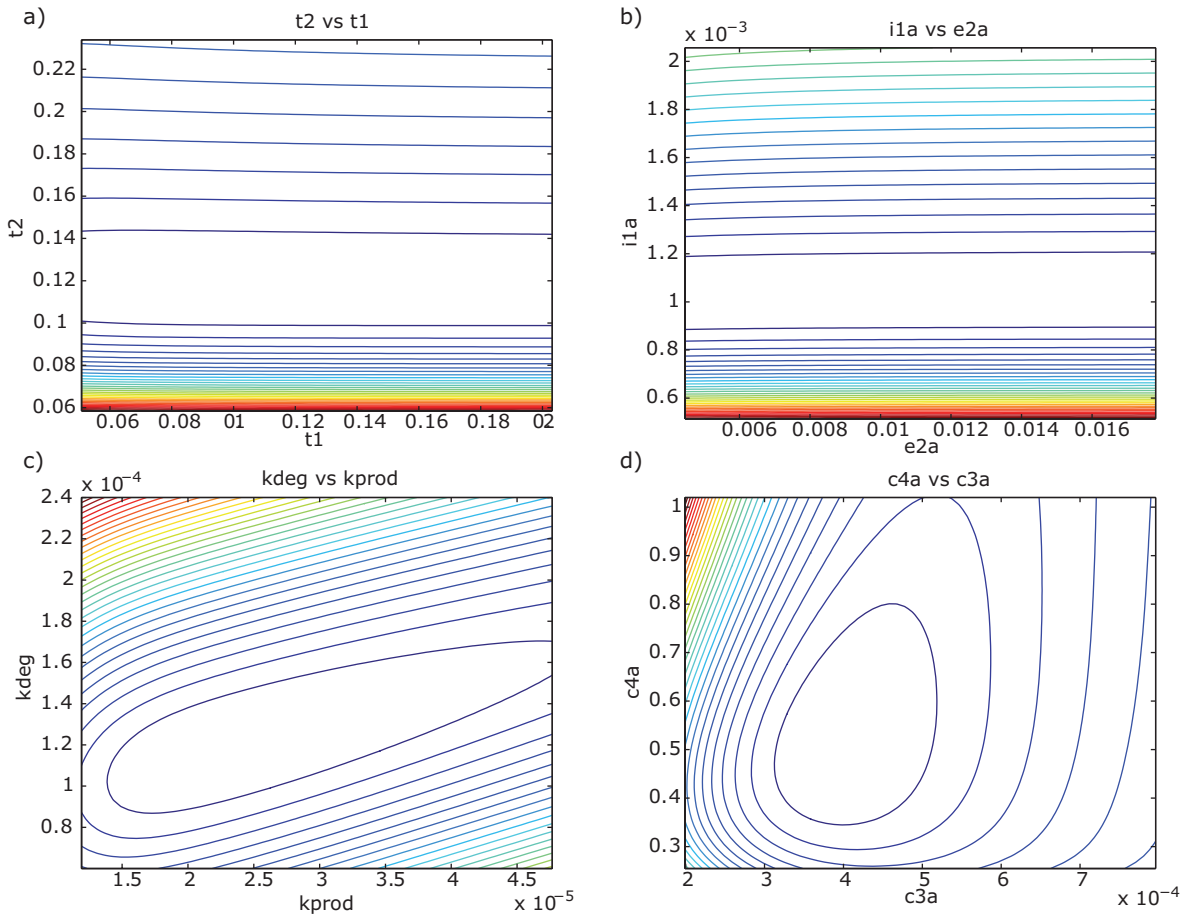


Figure A.21: NF κ B module: a) Two negatively highly correlated parameters, with one poorly-identifiable parameter (t_1); b) Two positively highly correlated parameters, with one poorly-identifiable parameter (e_{2a}); c) Two highly correlated, identifiable parameters; d) Two poorly correlated parameters.

Note that the user may access all numerical results by loading within the file `strreport_NFkB_opt.mat` the structure `results.contour`.

A.3.5 Robust identifiability analysis: `AMIGO_RIdent('nfkb_pe')`

In addition one may perform the robust identifiability analysis to assess quality of parameter estimates taking into account the experimental error and without the assumptions behind the Crammer-Rao inequality.

Type, for example:

```
>> AMIGO_RIdent('nfkb_ident','opt')      %To perform the robust identifiability analysis with ssm
>> AMIGO_RIdent('nfkb_ident','opt','local_dn2fb') %Robust identifiability analysis with dn2fb
>> AMIGO_RIdent('nfkb_ident','opt','fssm') %Robust identifiability analysis with fssm
```

Note that, as for parameter estimation, solver options should be changed for the robust identifiability analysis. This may be done by editing `ssm_options`, `de_options`, `sres_options` and `globalm_options`. In this case, one should take into consideration that the initial guess will be close the global optimum therefore maximum cpu time or number of iterations should be considerably reduced

as compare to the case of parameter estimation. This will prevent for excessively large computational costs.

Results reveal, as expected, serious problems to identify t_1 and e_{2a} , in fact for most of the runs the optimization converged to the bounds allowed as it can be seen in the figures:

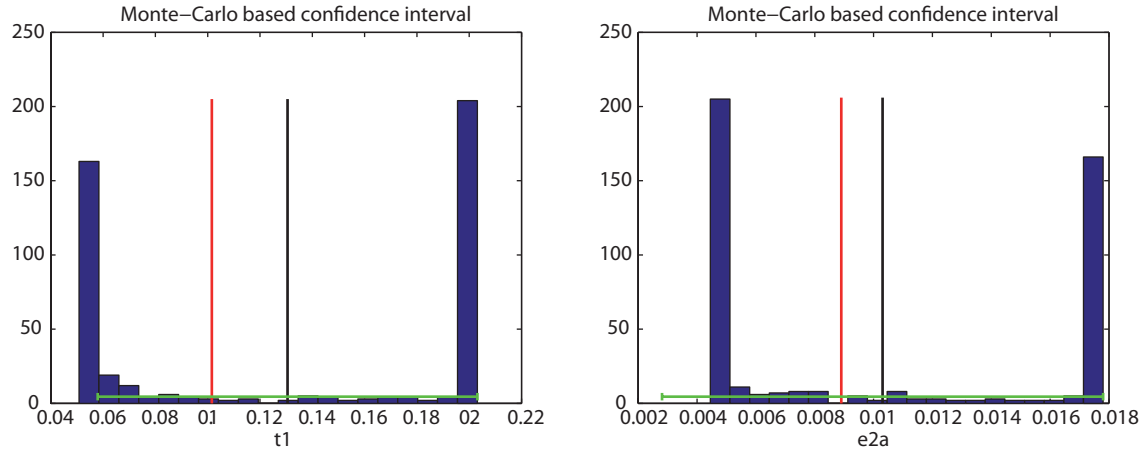


Figure A.22: NF κ B module: Robust confidence intervals for t_1 and e_{2a} . These exemplify the case of lack of identifiability. The confidence interval corresponds to the allowed range for the parameters. Note that the red line indicates the optimum value used as reference and the black line relates to the mean value obtained by the robust analysis.

Next figure shows examples of well identifiable parameters, with a confidence regions below the 10%:

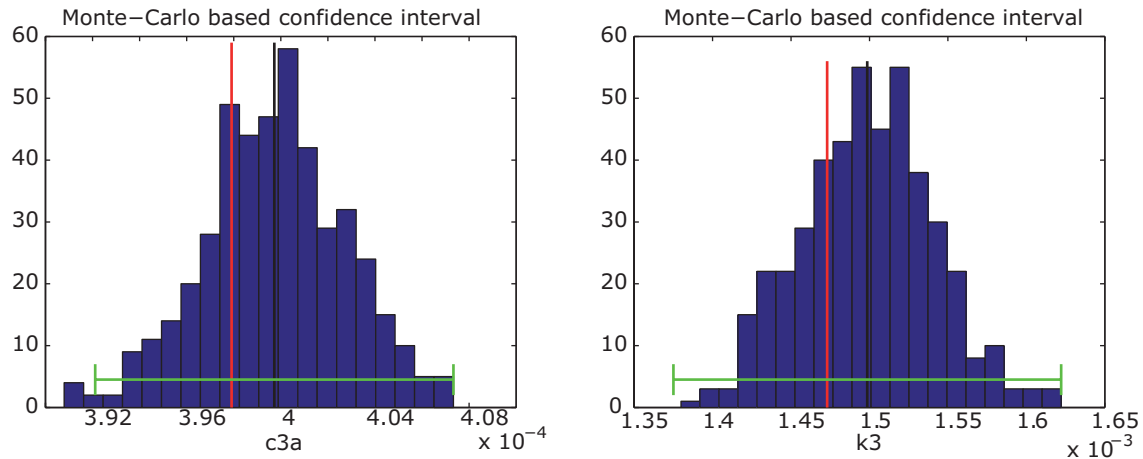


Figure A.23: NF κ B module: Robust confidence intervals for k_3 and c_{3a} . These exemplify the case of identifiable parameters. Note that the red line indicates the optimum value used as reference and the black line relates to the mean value obtained by the robust analysis. For the examples, both lines are close together indicating good identifiability properties and that the initial guess is in general successful to reproduce the experimental data within the given experimental error.

In addition to the confidence intervals, AMIGO allows to visualise the confidence hyper-ellipsoid by pairs of parameters. The following examples illustrate different possibilities:

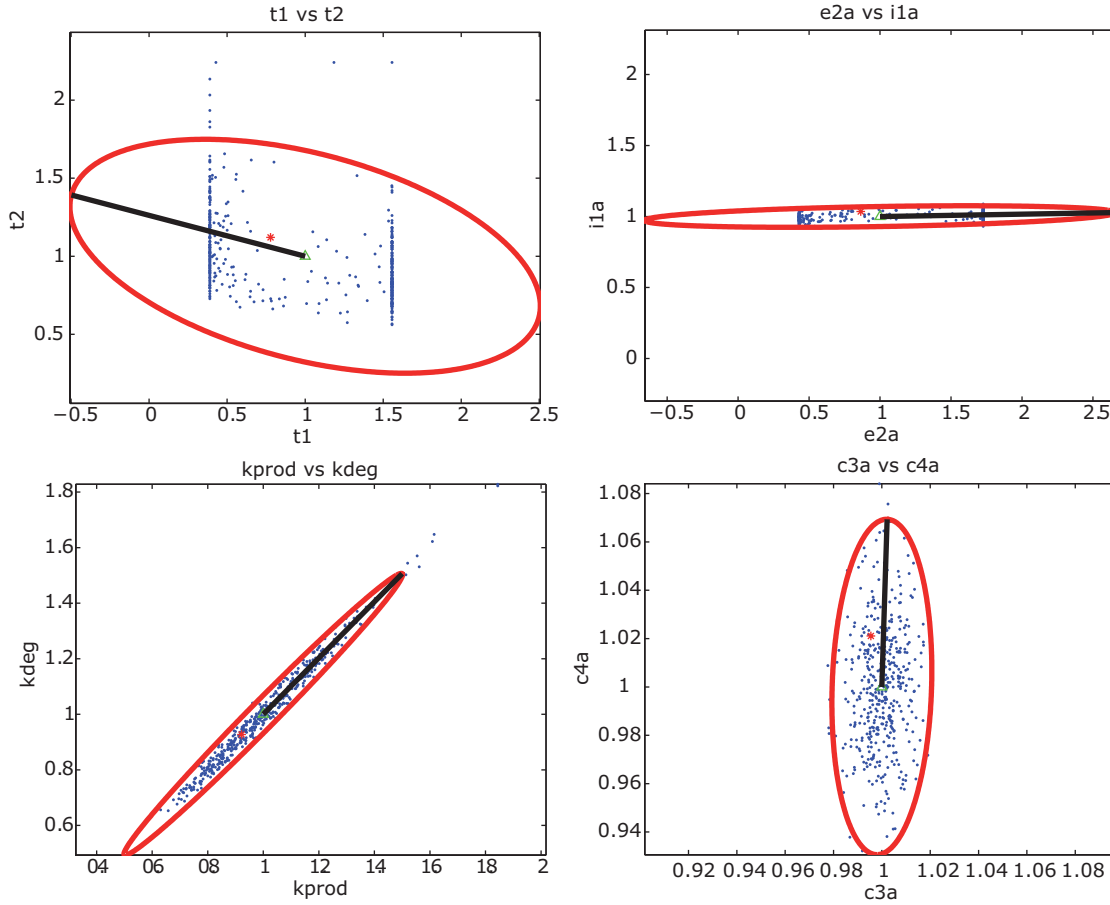


Figure A.24: NF κ B module: a) Poorly-identifiable parameter (t_1), clear tendency to converge to the bounds ; b) Two positively highly correlated parameters, with one poorly-identifiable parameter (e_{2a}); c) Two highly correlated, identifiable parameters; d) Two poorly correlated highly identifiable parameters. NOTE: parameter values are given as fraction of unity using the mean value (green triangle) as the reference, the red star indicates the initial guess.

The eccentricity plot allows to rapidly assess the correlation by pairs of parameters, note that the eccentricity is defined in the range $[0, 1]$, 0 corresponding to a circle, thus to complete uncorrelated parameters.

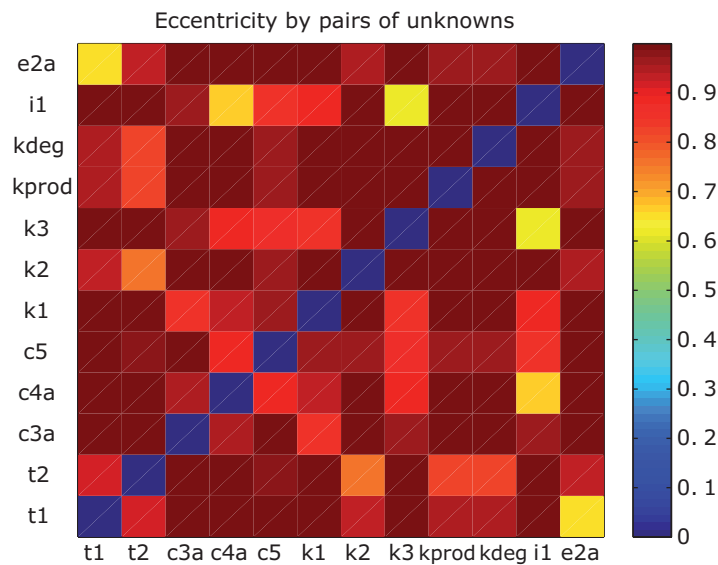


Figure A.25: NF κ B module: Eccentricity by pairs of parameters. Eccentricity of an ellipse is a measure of how nearly circular the ellipse is, the closer to the circle the closer the eccentricity to 0. This helps to assess the correlation between parameters.

The user may load inputs. and results. structures any time by typing:

```
>> load strreport_nfkb_opt.mat
```

The information is organised as follows:

```
inputs.
    model.: [1x1 struct], structure that keeps all model related inputs
    exps.: [1x1 struct], structure that keeps experimental scheme and data
    ivpsol.: [1x1 struct], structure that keeps information related to IVP and sens solvers
    PEsol.: [1x1 struct], structure that keeps information related to the
            parameter estimation problem
    nlpsol.: [1x1 struct], structure that keeps information related to the NLP solver
    rid.: [1x1 struct], structure that keeps information related to RIdent
    input_file.: 'nfkb_pe'
    pathd.: [1x1 struct], structure that keeps AMIGO path

results.
    pathd.: [1x1 struct], structure that keeps all paths and files names
    plotd.: [1x1 struct], structure that keeps information related to figures
    rid.: [1x1 struct], structure that keeps results of RIdent
```

```

results.rid.
  vtheta_guess: [1x13 double], values of unknowns used as initial guess for the analysis
  sorted_dist: [1x500 double], vector of ordered euclidean distances of the different
                                solutions to the initial guess
sorted_dist_max: [1x500 double], vector of ordered max distances of the different solutions
                                to the initial guess
sorted_dist_max95: [451x1 double], sorted max distances for the 0.05-0.95 interquantile range
sorted_dist_95: [451x1 double], sorted euclidean distances for the 0.05-0.95 interquantile
                                range
  sort_index_95: [1x450 double], sorted index of the solutions for the 0.05-0.95 interquantile
                                range
      best95: [450x13 double], matrix of selected values for the unknowns
  best95_norm: [450x13 double], matrix of selected values for the unknowns normalised by
                                the mean value
      mu: [1x13 double], mean of unknowns over the cloud
      lambda: [1x13 double], distance from the mean to the initial guess by components
  lambda_total: 0.0333, euclidean distance from the mean to the initial guess
confidence_interval: [1x13 double], robust confidence intervals
confidence_norm: [1x13 double], robust confidence intervals given as a fraction of one
  semi_major: [12x13 double], semi-major axes of the ellipses by pairs of parameters
  semi_minor: [12x13 double], semi-minor axes of the ellipses by pairs of parameters
  ecc: [13x13 double], matrix with the eccentricities of the ellipses by pairs of
                                parameters
      ecc_max: 0.9999, maximum eccentricity (the closer to 0 the most uncorrelation)
      ecc_min: 0.6223, minimum eccentricity
      ecc_mean: 0.9364, mean eccentricity
      alfa: [13x13 double], matrix with angles of the ellipses by pairs of parameters with
                                respect to X+ axis
      alfa_max: 80.2128, maximum angle
      alfa_min: 2.0441, minimum angle
      alfa_mean: 11.9047, mean angle
ellipse_pseudo_vol: 2.3305e-004, pseudo-volume of the confidence hyper-ellipsoid
  mc_corrmat: [14x14 double], Monte-Carlo based correlation matrix

```

A.4 The model of a three step pathway by Mendes

A.4.1 Introduction

The model of a pathway consisting of three enzymatic steps including the enzymes and mRNAs explicitly is considered. The scheme of the pathway is as follows:

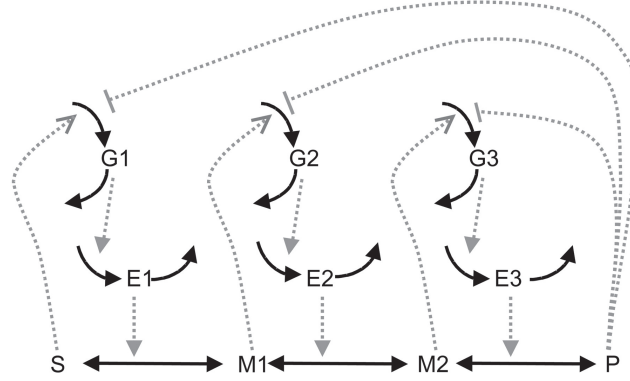


Figure A.26: Three step pathway. S and P are the pathway substrate and product; M₁ and M₂ are intermediate metabolites of the pathway; E₁, E₂, and E₃ are the enzymes; G₁, G₂, and G₃ are the mRNA species for the enzymes [31].

And the mathematical representation:

$$\begin{aligned}
 \dot{G}_1 &= \frac{V_1}{1 + (\frac{P}{K_{i1}})^{ni_1} + (\frac{Ka_1}{S})^{na_1}} - k_1 G_1 \\
 \dot{G}_2 &= \frac{V_2}{1 + (\frac{P}{K_{i2}})^{ni_2} + (\frac{Ka_2}{M_1})^{na_2}} - k_2 G_2 \\
 \dot{G}_3 &= \frac{V_3}{1 + (\frac{P}{K_{i3}})^{ni_3} + (\frac{Ka_3}{M_2})^{na_3}} - k_3 G_3 \\
 \dot{E}_1 &= \frac{V_4 G_1}{K_4 + G_1} - k_4 E_1 \\
 \dot{E}_2 &= \frac{V_5 G_2}{K_5 + G_2} - k_5 E_2 \\
 \dot{E}_3 &= \frac{V_6 G_3}{K_6 + G_3} - k_6 E_3 \\
 \dot{M}_1 &= \frac{kcat_1 E_1 (\frac{1}{Km_1})(S - M_1)}{1 + \frac{S}{Km_1} + \frac{M_1}{Km_2}} - \frac{kcat_2 E_2 \frac{1}{Km_3}(M_1 - M_2)}{1 + \frac{M_1}{Km_3} + \frac{M_2}{Km_4}} \\
 \dot{M}_2 &= \frac{kcat_2 E_2 \frac{1}{Km_3}(M_1 - M_2)}{1 + \frac{M_1}{Km_3} + \frac{M_2}{Km_4}} - \frac{kcat_3 E_3 \frac{1}{Km_5}(M_2 - P)}{1 + \frac{M_2}{Km_5} + \frac{P}{Km_6}}
 \end{aligned} \tag{A.4}$$

The parameter estimation problem associated to this model is considered a benchmark for new optimization methods and has been object of intensive research. (see for example, [31, 33, 41, 40], among others). A factorial plan consisting of 16 experiments under sustained substrate and product stimulation have been traditionally considered to estimate ell model parameters. Note however that the model is structurally non identifiable. Only a subset of parameters is locally identifiable. Thus we will consider here the case of estimating: $na_2, na_3, k_1, k_2, k_3, k_4, k_6, V_1, V_2, V_3, V_5, K_5$.

Let's define the model and the experimental scheme as independent files that can be then called from different AMIGO input files:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mendes_model.m file
%
% TITLE: The three step pathway by Mendes (2001)
%       Mendes P, 2001. Modeling large biological systems from functional
%       genomic data: Parameter estimation. In Foundations of systems
%       biology (ed. H. Kitano), pp. 163-186. MIT Press, Cambridge, MA.
%
% MODEL:
%       dG1dt = V1/(1+(P/Ki1)^ni1+(Ka1/S)^na1)- k_1*G1;
%       dG2dt = V2/(1+(P/Ki2)^ni2+(Ka2/M1)^na2) - k_2*G2;
%       dG3dt = V3/(1+(P/Ki3)^ni3+(Ka3/M2)^na3) - k_3*G3;
%       dE1dt = V4*G1/(K4+G1) - k_4*E1;
%       dE2dt = V5*G2/(K5+G2) - k_5*E2;
%       dE3dt = V6*G3/(K6+G3) - k_6*E3;
%       dM1dt = kcat1*E1*(1/Km1)*(S-M1)/(1+S/Km1+M1/Km2)-kcat2*E2*(1/Km3)*(M1-M2)/(1+M1/Km3+M2/Km4);
%       dM2dt = kcat2*E2*(1/Km3)*(M1-M2)/(1+M1/Km3+M2/Km4)-kcat3*E3*(1/Km5)*(M2-P)/(1+M2/Km5+P/Km6);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       INPUT FILE TO GENERATE FOR ITS USE IN AMIGO
%       > Paths related data
%       > Model:           model_type; n_st; n_par; n_stimulus;
%                           st_names; par_names; stimulus_names;
%                           eqns; par
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%=====
% PATHS RELATED DATA
%=====
results.pathd.results_folder='Mendes'; % Folder to keep results (within Results)
results.pathd.short_name='Mendes';      % To identify figures and reports
%=====
% MODEL RELATED DATA
%=====

inputs.model.input_model_type='charmodelF';
inputs.model.n_st=8;                    % Number of states
inputs.model.n_par=36;                  % Number of model parameters
inputs.model.n_stimulus=2;              % Number of stimuli variables
inputs.model.st_names=char('G1','G2','G3','E1','E2','E3','M1','M2');
inputs.model.par_names=char('V1','Ki1','ni1','Ka1','na1','k_1','V2','Ki2',...
    'ni2','Ka2','na2','k_2','V3','Ki3','ni3','Ka3','na3','k_3',...
    'V4','K4','k_4','V5','K5','k_5','V6','K6','k_6',...
    'kcat1','Km1','Km2','kcat2','Km3','Km4','kcat3','Km5','Km6');
inputs.model.stimulus_names=char('S','P'); % Names of the stimuli
inputs.model.eqns=char('dIKKn=kprod-kdeg*IKKn-Tr*k1*IKKn',...
    'dIKKa=Tr*k1*IKKn-k3*IKKa-Tr*k2*IKKa*A20-kdeg*IKKa-a2*IKKa*IcBa+t1*IKKa*IcBa-a3*IKKa*IcBaNFkB+
    t2*IKKa*IcBaNFkB',...
    'dG1=V1/(1+(P/Ki1)^ni1+(Ka1/S)^na1)- k_1*G1',...
    'dG2= V2/(1+(P/Ki2)^ni2+(Ka2/M1)^na2) - k_2*G2',...
    'dG3= V3/(1+(P/Ki3)^ni3+(Ka3/M2)^na3) - k_3*G3',...
    'dE1= V4*G1/(K4+G1) - k_4*E1',...
    'dE2= V5*G2/(K5+G2) - k_5*E2',...
    'dE3= V6*G3/(K6+G3) - k_6*E3',...
    'dM1=kcat1*E1*(1/Km1)*(S-M1)/(1+S/Km1+M1/Km2)-kcat2*E2*(1/Km3)*(M1-M2)/(1+M1/Km3+M2/Km4)',...
    'dM2=kcat2*E2*(1/Km3)*(M1-M2)/(1+M1/Km3+M2/Km4)-kcat3*E3*(1/Km5)*(M2-P)/(1+M2/Km5+P/Km6)');
inputs.model.par=[1 1 2 1 2 1 1 1 2 1 2 1 1 1 2 1 2 ...
    1 0.1 1 0.1 0.1 1 0.1 0.1 1 0.1 1 1 1 1 1 1 1 1 1]; % Nominal value for the parameters

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%menes_experimental_scheme.m
%
% EXPERIMENTAL SCHEME: 16 experiments performed under different S and P
%                      (inputs) conditions and where all states are measured
%                      at 21 equidistant sampling times
%
% REFERENCES:
%   >Moles, C. G., Pedro Mendes and Julio R. Banga (2003) Parameter
%   estimation in biochemical pathways: a comparison of global
%   optimization methods. Genome Research, 13(11):2467-2474
%   >Rodriguez-Fernandez, M., J. A. Egea and J. R. Banga (2006) Novel
%   Metaheuristic for Parameter Estimation in Nonlinear Dynamic Biological
%   Systems. BMC Bioinformatics 7:483.
%
% NOTE!!!: [] indicates that the corresponding input may be omitted,
%           default value will be assigned
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%=====
% EXPERIMENTAL SCHEME RELATED DATA
%=====
inputs.exps.n_exp=16; % Number of experiments

% Most inputs are common to all experiments therefore a loop over
% experiments is defined
for iexp=1:inputs.exps.n_exp
    inputs.exps.obs{iexp}='states'; % All states in model are measured
    inputs.exps.exp_y0{iexp}=[6.6667e-1 5.7254e-1 4.1758e-1...
    4.0e-1 3.6409e-1 2.9457e-1 1.419 9.3464e-1]; % Initial conditions for each experiment
    inputs.exps.t_f{iexp}=120; % Experiments duration
    inputs.exps.n_s{iexp}=21; % Number of sampling times
    inputs.exps.u_interp{iexp}='sustained'; % [] Stimuli definition|
    inputs.exps.t_con{iexp}= [0 120]; % Control switching times: Initial and final
end
% VALUES OF INPUTS FOR THE DIFFERENT EXPERIMENTS
% FACTORIAL PLAN covering combinations of 4 levels for each input: [S;P]
inputs.exps.u{1}=[0.1; 0.05];
inputs.exps.u{2}=[0.1; 0.13572];
inputs.exps.u{3}=[0.1; 0.3684];
inputs.exps.u{4}=[0.1; 1];
inputs.exps.u{5}=[0.46416; 0.05];
inputs.exps.u{6}=[0.46416; 0.13572];
inputs.exps.u{7}=[0.46416; 0.3684];
inputs.exps.u{8}=[0.46416; 1];
inputs.exps.u{9}=[2.1544; 0.05];
inputs.exps.u{10}=[2.1544; 0.13572];
inputs.exps.u{11}=[2.1544; 0.3684];
inputs.exps.u{12}=[2.1544; 1];
inputs.exps.u{13}=[10; 0.05];
inputs.exps.u{14}=[10; 0.13572];
inputs.exps.u{15}=[10; 0.3684];
inputs.exps.u{16}=[10; 1];

```

A.4.2 Parameter estimation under sustained stimulation: AMIGO_PE('mendes_pe')

The objective is to compute the 36 model parameters under the experimental scheme above and the following bounds for the parameters: Hill coefficients are allowed to vary within the range (0.1, 10) and all other parameters allowed to vary within the range ($1e-6$, $1e3$). This problem has been extensively considered in the literature ([33, 41, 40]). The problem is formulated as follows:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mendes_pe.m
%
% PARAMETER ESTIMATION: To find the 36 unknown parameters from a set of 16
%   experiments performed under different S and P (inputs) conditions
%   and where all states are measured at 21 equidistant sampling times
%   Parameters are classified in:
%       Hill coefficients: allowed to vary within the range (0.1, 10)
%       and all other parameters allowed to vary within the
%       range (1e-6, 1e3).
%
% REFERENCES:
%   >Moles, C. G., Pedro Mendes and Julio R. Banga (2003) Parameter
%   estimation in biochemical pathways: a comparison of global
%   optimization methods. Genome Research, 13(11):2467-2474
%   >Rodriguez-Fernandez, M., J. A. Egea and J. R. Banga (2006) Novel
%   Metaheuristic for Parameter Estimation in Nonlinear Dynamic Biological
%   Systems. BMC Bioinformatics 7:483.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mendes_model
mendes_experimental_scheme
%=====
% EXPERIMENTAL DATA RELATED INFO
%=====
inputs.exps.data_type='real';      % Type of data: 'pseudo' | 'pseudo_pos' | 'real'
inputs.exps.noise_type='homo';     % Experimental noise: Homoscedastic constant variance: 'homo'
load('mendes_exp_data.mat');      % Reads data from a .mat file
for iexp=1:inputs.exps.n_exp
inputs.exps.exp_data[iexp]=[G1(:,iexp) G2(:,iexp) G3(:,iexp) ...
    E1(:,iexp) E2(:,iexp) E3(:,iexp) M1(:,iexp) M2(:,iexp)];
end
%=====
% UNKNOWNNS RELATED DATA
%=====
% GLOBAL UNKNOWNNS, Maximum and minimum allowed values
inputs.PEsol.id_global_theta=char('k_3', 'na3', 'na2', 'k_6', 'k_2', 'k_4', 'k_1', 'V3', 'V2',...
    'V1', 'V5', 'K5');
inputs.PEsol.global_theta_max=[ 1e3 10 10  1e3 1e3 1e3 1e3 1e3 1e3 1e3 1e3 1e3];
inputs.PEsol.global_theta_min=[ 1e-6 0.1 0.1 1e-6 1e-6 1e-6 1e-6 1e-6 1e-6 1e-6 1e-6 1e-6];
%=====
% COST FUNCTION RELATED DATA
%=====
inputs.PEsol.PEcost_type='lsq';    % 'lsq' (weighted least squares default)
inputs.PEsol.lsqr_type='Q_expmax'; % 'Q_I' (weighting matrix the identity)
                                     % 'Q_expmax' (weighting matrix using max exp data)

```

Type: AMIGO_Prep('mendes_model') to preprocess the model and AMIGO_Pe('mendes_pe') to solve the parameter estimation with ssm.

In a few seconds the global objective corresponding to $J_{lsq} = 0$ is achieved corresponding to the following parameter values:

```

k_3   : 1.0000e+000 +- 8.7695e+000;
na3   : 2.0000e+000 +- 7.0158e-001;
na2   : 2.0000e+000 +- 7.2765e-001;
k_6   : 1.0000e-001 +- 1.7043e-002;
k_2   : 9.9991e-001 +- 6.8614e+000;
k_4   : 1.0000e-001 +- 1.2587e-002;
k_1   : 1.0001e+000 +- 4.3768e+000;
V3    : 1.0000e+000 +- 8.7711e+000;
V2    : 9.9991e-001 +- 6.8795e+000;
V1    : 1.0001e+000 +- 4.3692e+000;
V5    : 1.0000e-001 +- 7.2961e-002;
K5    : 1.0000e+000 +- 1.2206e+000;

```

Note that even though the parameter values correspond to the nominal ones that were used to generate the pseudo-experimental data, the confidence regions for some of them are significantly large, in many cases (k_3 , k_2 , k_1 , V_3 , V_2 , V_1 , K_5) over the 100%. In addition the correlation matrix reveal some highly correlated pairs of parameters.

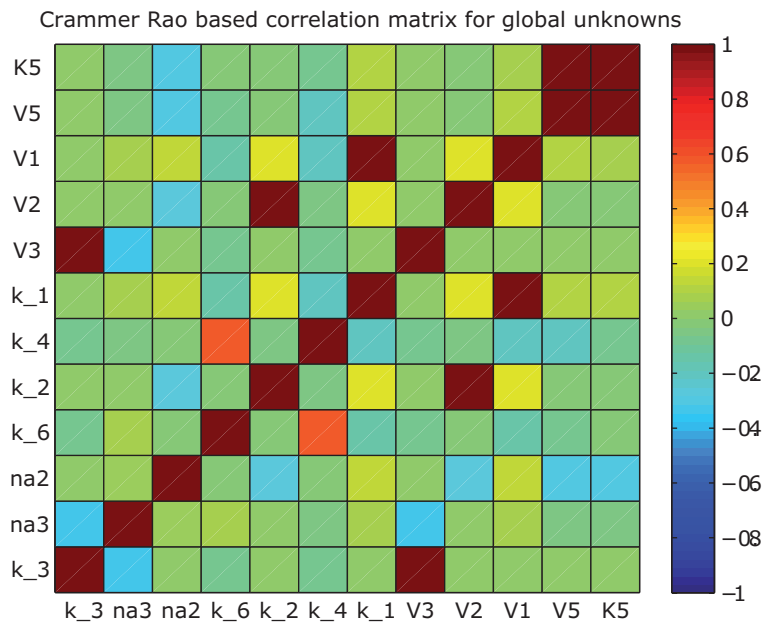


Figure A.27: Three step pathway: Correlation matrix at the optimum obtained by a sustained experimental scheme.

In order to improve the practical identifiability one may consider to design some new experiments. To identify which type of experiments would be more informative, it is possible to implement several experiments and to perform a sensitivity analysis to asses under which conditions the model becomes more sensitive to the model parameters.

A.4.3 Sensitivity analysis under dynamic stimulation: AMIGO_LRank('mendes_uvar')

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mendes_experimental_scheme_uvar.m
% EXPERIMENTAL SCHEME: 8 experiments performed under different S and P (inputs) conditions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
inputs.exps.n_exp=8; % Number of experiments
for iexp=1:inputs.exps.n_exp
    inputs.exps.obs{iexp}='states'; % All states in model are measured
    inputs.exps.exp_y0{iexp}=[6.6667e-1 5.7254e-1 4.1758e-1...
    4.0e-1 3.6409e-1 2.9457e-1 1.419 9.3464e-1]; % Initial conditions for each experiment
    inputs.exps.t_f{iexp}=120; % Experiments duration
    inputs.exps.n_s{iexp}=21; % Number of sampling times
end
inputs.exps.n_exp=8; % Number of experiments
% EXPERIMENT 1: SUSTAINED STIMULATION FOR BOTH INPUTS
inputs.exps.u_interp{1}='sustained';
inputs.exps.u{1}=[0.1; 0.05];
inputs.exps.t_con{1}= [0 120];
% EXPERIMENT 2: ONE PULSE-UP: PULSE FOR THE TWO INPUTS AT THE SAME TIME
inputs.exps.u_interp{2}='pulse-up';
inputs.exps.n_pulses{2}=1;
inputs.exps.u_min{2}=[0.1; 0.05 ]; inputs.exps.u_max{2}=[10; 1];
inputs.exps.t_con{2}= [0 40 80 120];
% EXPERIMENT 3: ONE PULSE-DOWN: PULSE FOR THE TWO INPUTS AT THE SAME TIME
inputs.exps.u_interp{3}='pulse-down';
inputs.exps.n_pulses{3}=1;
inputs.exps.u_min{3}=[0.1; 0.05 ]; inputs.exps.u_max{3}=[10; 1];
inputs.exps.t_con{3}= [0 60 120];
% EXPERIMENT 4: SUSTAINED FOR P AND PULSE-DOWN FOR S
inputs.exps.u_interp{4}='pulse-down';
inputs.exps.n_pulses{4}=1;
inputs.exps.u_min{4}=[0.1; 0.05 ]; inputs.exps.u_max{4}=[10; 0.05];
inputs.exps.t_con{4}= [0 60 120];
% EXPERIMENT 5: PULSE-DOWN FOR S AND P of DIFFERENT DURATIONS
inputs.exps.u_interp{5}='step';
inputs.exps.n_steps{5}=3;
inputs.exps.u{5}(1,:)= [10 0.1 0.1]; inputs.exps.u{5}(2,:)= [1 1 0.05];
inputs.exps.t_con{5}= [0 60 80 120]; % Switching times: t_con should be of size n_steps+1:
% Every t_con indicates when the step is started and the last t_con indicates the end of last step
% EXPERIMENT 6: PULSE-UP FOR S AND P at DIFFERENT LOCATIONS
inputs.exps.u_interp{6}='step';
inputs.exps.n_steps{6}=5;
inputs.exps.u{6}(1,:)= [0.1 0.1 10 10 0.1]; inputs.exps.u{6}(2,:)= [0.05 1 1 0.05 0.05];
inputs.exps.t_con{6}= [0 30 50 70 90 120];
% EXPERIMENT 7: STEP-WISE FOR S AND P ( S illustrates the implementation of a stair-wise profile)
inputs.exps.u_interp{7}='step';
inputs.exps.n_steps{7}=7;
inputs.exps.u{7}(1,:)= [10 10 7 7 3 3 1];
inputs.exps.u{7}(2,:)= [0.05 0.75 0.75 1 1 0.5 0.1];
inputs.exps.t_con{7}= [0 15 40 60 80 90 110 120];
% EXPERIMENT 8: LINEAR-INTERPOLATED PROFILE FOR S AND P
inputs.exps.u_interp{8}='linear';
inputs.exps.n_linear{8}=8;
inputs.exps.u{8}(1,:)= [0.1 10 0.2 0.35 0.5 0.35 0.2 0.2 ];
inputs.exps.u{8}(2,:)= [0.05 0.5 0.75 1 1 1 0.55 0.25];
inputs.exps.t_con{8}= [0 15 30 45 60 80 95 120];

```

The local sensitivity analysis for the nominal value of the parameters results:

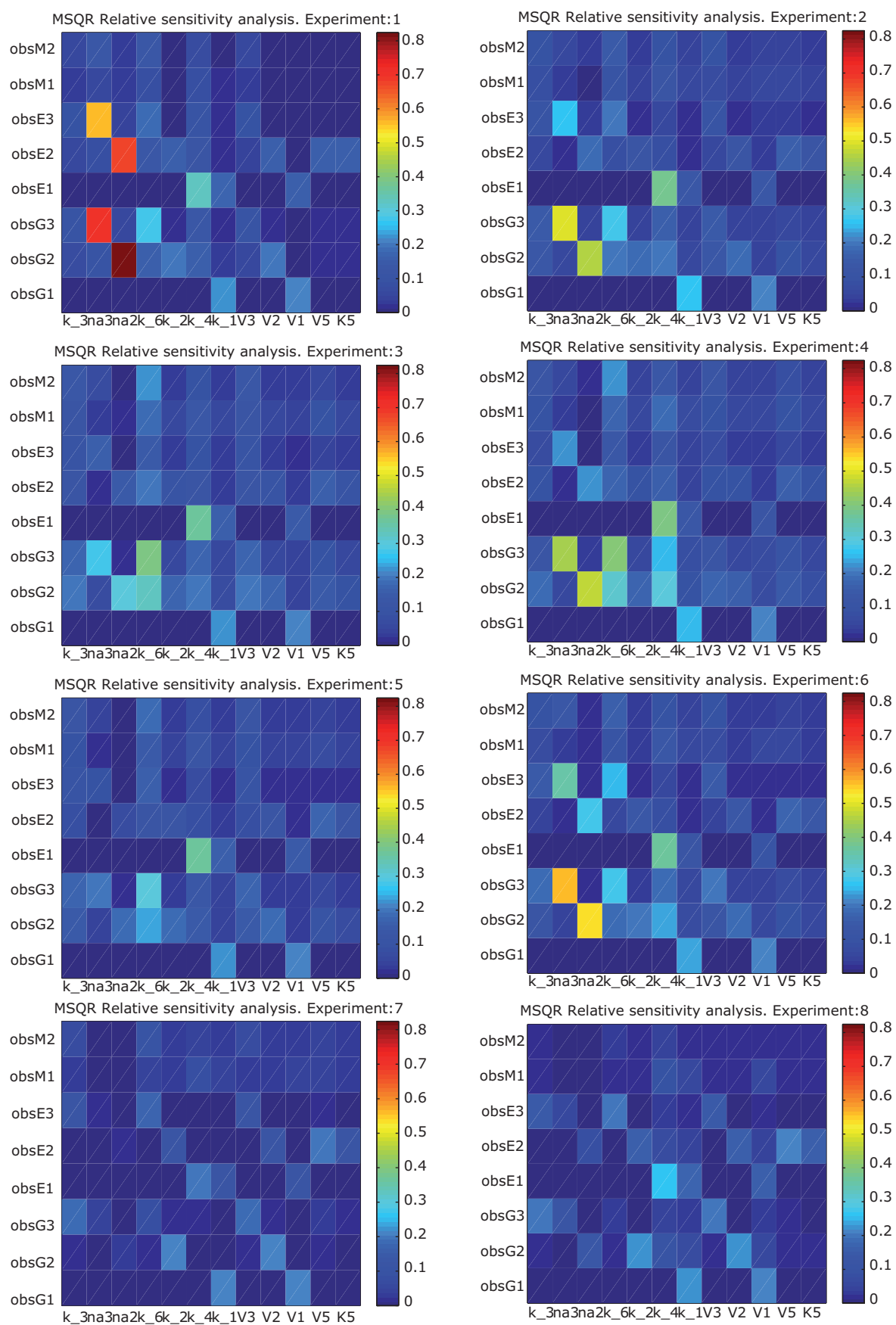


Figure A.28: Three step pathway: Local sensitivity analysis for the nominal value of the parameters under different dynamic stimulation profiles.

Focusing our attention on the subset of parameters with larger confidence regions, it seems that experiments 2 – 6 result, in general, in more sensitivity of the model to those parameters. Therefore pulse-wise or step-wise experiments seem to be more informative for the purpose of parameter estimation. The following step is then to optimally design new experiments.

A.4.4 Solving the optimal experimental design problem: `AMIGO_OED('mendes_oed')`

The following example is intended to show how to implement a parallel-sequential experimental scheme to improve identifiability, to take into account previous experiments and to design new experiments to obtain complementary information for the purpose of parameter estimation. Assuming that we can modify the substrate input profiles and taking into consideration previous results, it seems that pulse-wise or step-wise S profiles would be more informative for the purpose of parameter estimation.

In particular two experiments will be designed:

- Experiment 17: pulsed stimulation of S is assumed. The location and duration of the pulses will be optimized as well as the number and location of sampling times and experiment duration.
- Experiment 18: a step-wise stimulation for S will be allowed within the maximum and minimum values. Note that, with step-wise profiles we may end-up in pulse-wise profiles if the latter are optimal.

In both experiments the quantity of product will be assumed constant through out the experiment.

The input file would be as follows:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mendes_oed.m
%
%      (AMIGO_OED)==>>  exp_y0_type; tf_type; u_type; ts_type
%                        u_interp, [n_steps], [n_pulses]
%                        [exp_y0_min /max]; [tf_min/tf_max];
%                        [ts_min_dist]; [u_min/u_max]
%                        [exp_dataiexp]; [error_dataiexp]
%                        id_global_theta; [id_global_theta_y0]
%                        [global_theta_guess]; [global_theta_y0_guess];
%                        PEcost_type; [lsq_type]; [llk_type]; OEDcost_type
%                        []:optional inputs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mendes_model
%=====
% EXPERIMENTAL SCHEME RELATED DATA
%=====
    inputs.exps.n_exp=18;                                % Total number of experiments
%FIXED (PREVIOUS) EXPERIMENTS
for iexp=1:16
    inputs.exps.obs{iexp}='states';
    inputs.exps.exp_y0{iexp}=[6.6667e-1 5.7254e-1 4.1758e-1...
    4.0e-1 3.6409e-1 2.9457e-1 1.419 9.3464e-1];          % Initial conditions for each experiment
    inputs.exps.t_f{iexp}=120;                             % Experiments duration
    inputs.exps.n_s{iexp}=21;                              % Number of sampling times
    inputs.exps.u_interp{iexp}='sustained';                % [] Stimuli definition
    inputs.exps.t_con{iexp}= [0 120];                     % Control switching times: Initial and final
end
```

```

% FACTORIAL PLAN covering combinations of 4 levels for each input
inputs.exps.u{1}=[0.1; 0.05];      inputs.exps.u{2}=[0.1; 0.13572];
inputs.exps.u{3}=[0.1; 0.3684];    inputs.exps.u{4}=[0.1; 1];
inputs.exps.u{5}=[0.46416; 0.05];  inputs.exps.u{6}=[0.46416; 0.13572];
inputs.exps.u{7}=[0.46416; 0.3684]; inputs.exps.u{8}=[0.46416; 1];
inputs.exps.u{9}=[2.1544; 0.05];   inputs.exps.u{10}=[2.1544; 0.13572];
inputs.exps.u{11}=[2.1544; 0.3684]; inputs.exps.u{12}=[2.1544; 1];
inputs.exps.u{13}=[10; 0.05];      inputs.exps.u{14}=[10; 0.13572];
inputs.exps.u{15}=[10; 0.3684];    inputs.exps.u{16}=[10; 1];

%EXPERIMENTS TO BE OPTIMALLY DESIGNED

%EXPERIMENT 17: PULSE-UP experiment with 3 pulses and 21 equidistant sampling times
inputs.exps.obs{17}='states';      % All states in model are measured
inputs.exps.exp_y0_type{17}='fixed'; % Type of initial conditions: 'fixed' | 'od' (to
                                     be designed)
inputs.exps.exp_y0{17}=[6.6667e-1 5.7254e-1 4.1758e-1...
4.0e-1 3.6409e-1 2.9457e-1 1.419 9.3464e-1]; % Fixed Initial conditions
inputs.exps.u_type{17}='od';        % Type of stimulation: 'fixed'
inputs.exps.u_interp{17}='pulse-up'; % Stimuli definition for experiment 17
inputs.exps.n_pulses{17}=2;
inputs.exps.u_min{17}=[0.1; 0.05];
inputs.exps.u_max{17}=[10; 0.05];   % Minimum and maximum value for the inputs S and P
inputs.exps.tf_type{17}='od';        % [] Type of experiment duration: 'fixed'(default)
                                     | 'od' (to be designed)

inputs.exps.tf_max17= 120;           % Minimum and maximum experiment duration
inputs.exps.tf_min17=60;             % [] Type of sampling times: 'fixed'(default) | 'od'
inputs.exps.ts_type{17}='od';        % First allowed sampling time
inputs.exps.ts_0{17}=0.0;           % Minimum distance between sampling times
inputs.exps.ts_min_dist{17}=6.0;

%EXPERIMENT 18: PULSE-DOWN experiment with 2 pulses and optimally located sampling times
inputs.exps.obs{18}='states';        % All states in model are measured
inputs.exps.exp_y0_type{18}='fixed';  % Type of initial conditions: 'fixed' | 'od'
inputs.exps.exp_y0{18}=[6.6667e-1 5.7254e-1 4.1758e-1...
4.0e-1 3.6409e-1 2.9457e-1 1.419 9.3464e-1]; % Fixed Initial conditions
inputs.exps.u_type{18}='od';          % Type of stimulation: 'fixed' | 'od'
inputs.exps.u_interp{18}='step';      % Stimuli definition for experiment 18
inputs.exps.n_steps{18}=4;
inputs.exps.u_min18=[0.1 0.1 0.1 0.1; 0.05 0.05 0.05 0.05];
inputs.exps.u_max18=[10 10 10 10; 0.05 0.05 0.05 0.05]; % Minimum and maximum value for S and P
inputs.exps.tf_type{18}='fixed';       % [] Type of experiment duration: 'fixed' | 'od'
inputs.exps.t_f{18}=120;              % Experiment duration
inputs.exps.ts_type{18}='fixed';       % [] Type of sampling times: 'fixed' | 'od'
inputs.exps.n_s{18}=21;               % Number of sampling times
%=====
% PARAMETERS TO BE CONSIDERED FOR OED
%=====
inputs.PEsol.id_global_theta=char('V1','k_1','V2', 'k_2','V3','k_3', 'K5');
inputs.PEsol.global_theta_guess=[1.0001 1.0001... % Nominal value of the parameters to compute the FIM
9.9991e-001 9.9991e-001 1.0000 1.0000 1.0000];
%=====
% COST FUNCTION RELATED DATA
%=====
inputs.PEsol.PEcost_type='lsq';        % Details of the cost function used in PE
inputs.PEsol.lsqr_type='Q_expmx';     % This information is necessary to compute the FIM
inputs.OEDsol.OEDcost_type='Eopt';    % Alphabetical criteria: Dopt| Eopt| Aopt| Emod|
                                     % DoverE

```

Type:

```
AMIGO_OED('mendes_oed')
AMIGO_OED('mendes_oed','Eopt','sres')
AMIGO_OED('mendes_oed','Dopt','hyb_de_fmincon')
```

to solve the optimal experimental design problem with `ssm`, `sres` or `hyb_de_fmincon` respectively.

NOTES:

- `ssm` or `fssm` have been shown to be very successful in dealing with dynamic optimization problems [12], therefore being recommended for OED. Take into consideration that `dn2fb` and `n2fb` are designed to solve least squares problems, therefore modify `ssm_options` or `fssm_options` not to use those solvers as locals.
- One may use the run identifier to indicate the cost function used for OED.

Even allowing for limited flexibility in the design of the experiments, results reveal a substantial reduction in the confidence regions for the parameters.

k_3	:	1.0000e+000	+-	8.7695e+000;	---OED--->	4.1299e+000	(-53%)
na3	:	2.0000e+000	+-	7.0158e-001;			
na2	:	2.0000e+000	+-	7.2765e-001;			
k_6	:	1.0000e-001	+-	1.7043e-002;			
k_2	:	9.9991e-001	+-	6.8614e+000;	---OED--->	3.5207e+000	(-49%)
k_4	:	1.0000e-001	+-	1.2587e-002;			
k_1	:	1.0001e+000	+-	4.3768e+000;	---OED--->	1.9380e+000	(-56%)
V3	:	1.0000e+000	+-	8.7711e+000;	---OED--->	4.1269e+000	(-53%)
V2	:	9.9991e-001	+-	6.8795e+000;	---OED--->	3.5293e+000	(-49%)
V1	:	1.0001e+000	+-	4.3692e+000;	---OED--->	1.9350e+000	(-56%)
K5	:	1.0000e+000	+-	1.2206e+000;	---OED--->	1.9241e-001	(-84%)

Following figures show the two optimally designed experiments:

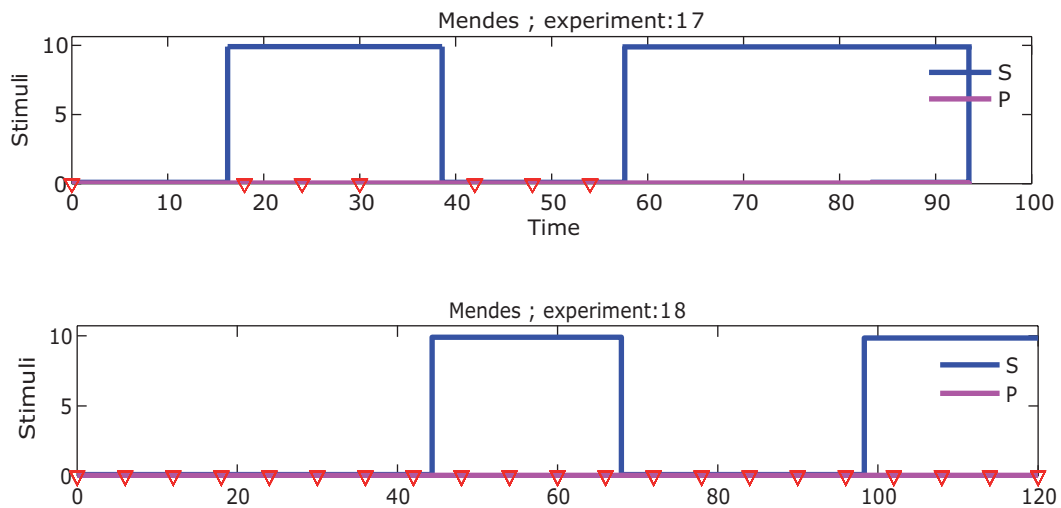


Figure A.29: Three step pathway: Optimally designed experiments. Experiment 17: pulse-up with two pulses for S and P kept constant; free final time duration between 60 and 120 and minimum distance between sampling times of 6. Experiment 18: step-wise profile with 4 steps for S and P kept constant.

It should be noted that for the experiment 17 the number and location of sampling times was optimally designed, resulting in a very reduced number of necessary sampling times (7). It is also remarkable that optimal step-wise experiment 18 results in a pulse-up stimulation. Further improvements would be possible by designing new experiments.

Results will be kept in the folder:

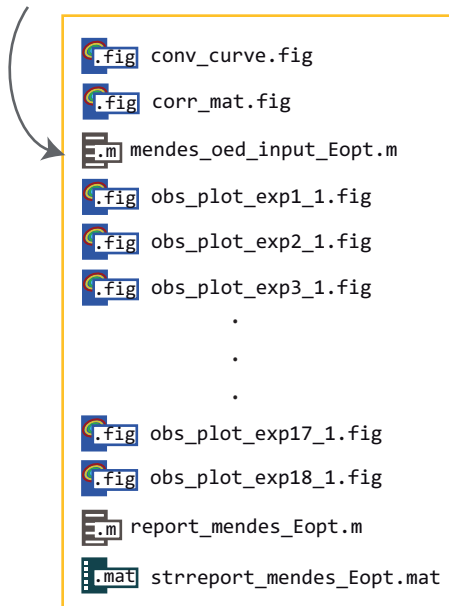
Results\Mendes_3steppath_model\OED_Mendes_hyb_sres_fmincon_Eopt

as indicated in the last line of the output and will be organised as follows:

AMIGO Path\Results

Mendes_3steppath_model

OED_Mendes_hyb_sres_fmincon_Eopt



The folder `Mendes_3steppath_model` keeps:

- > fcn.f and sens.f the FORTRAN code generated during preprocessing
- > A .m file to compute observation function

The folder `OED_Mendes_hyb_sres_fmincon_Eopt` keeps:

- > A copy of the input file
- > A .m report with inputs and results
- > Several .fig files with the plots of the evolution of states with time for the overall experimental scheme (18 experiments in this case).
- > corr_mat.fig plot of the correlation matrix for the OED
- > conv_curve.fig plot of the convergence curve for the NLP solver
- > A .mat file which keeps the inputs. and results. structures.

Figure A.30: Contents of folder Results\circadian-tutorial\SModel_circadian_run1

The user may load inputs. and results. structures at by typing:

```
>> load strreport_mendes_Eopt.mat
```

The information is organised as follows:

```
inputs.
    model.: [1x1 struct], structure that keeps all model related inputs
    exps.: [1x1 struct], structure that keeps experimental scheme and data
    ivpsol.: [1x1 struct], structure that keeps information related to IVP and sens solvers
    PEsol.: [1x1 struct], structure that keeps information related to the
        parameter estimation problem
    OEDsol.: [1x1 struct], structure that keeps information related to the
        optimal experimental design problem formulation
    nlpsol.: [1x1 struct], structure that keeps information related to the NLP solver
    rid.: [1x1 struct], structure that keeps information related to RIdent
    input_file.: 'nfkb_pe'
    pathd.: [1x1 struct], structure that keeps AMIGO path
```

```

results.
  pathd.: [1x1 struct], structure that keeps all paths and files names
  plotd.: [1x1 struct], structure that keeps information related to figures
  sim.: [1x1 struct], structure simulation results for the optimal experimental scheme
  nlpsol.: [1x1 struct], structure that keeps information about the NLP solution
           (best cost, best design, convergence curve, CPU time, ect.)
  oed.: [1x1 struct], structure that keeps results of OED

results.oed.
  n_exp: 18,          total number of experiments in the experimental scheme
  n_obs: {[8] [8] [8] [8] [8] [8] [8] [8] [8] [8] [8] [8] [8] [8] [8] [8]}, cell array with the
           number of observables for each experiment
  obs: {1x18 cell},  cell array with the observables per experiment
  n_s: {1x18 cell},  cell array with the number of sampling times per experiment
  t_s: {1x18 cell},  cell array with the sampling times per experiment
  t_f: {1x18 cell},  cell array with the final time per experiment
  u: {1x18 cell},    cell array with the stimuli values per experiment
  t_con: {1x18 cell}, cell array with the switching times for the stimuli
  exp_y0: {1x18 cell}, cell array with the initial conditions per experiment
  w_sampling: {1x18 cell}, cell array with the weights for the sampling times(>0.5 when
           used)
  sens_t: {1x18 cell}, cell array with the sensitivities of observables with respect
           to the parameters at each sampling time for the OED
  r_sens_t: {1x18 cell}, cell array with the relative sensitivities of observables with
           respect to the parameters at each sampling time for the OED
  ms: {1x18 cell},    cell array with the observables values at each sampling time
           for the OED
  g_FIM: [7 x 7 double], global Fisher information matrix for the OED
  g_corr_mat: [7 x 7 double], global correlation matrix for the OED
  conf_intervals: [1.9350 1.9380 3.5293 3.5207 4.1269 4.1299 0.1924], parameters confidence
           intervals for the OED

```

Bibliography

- [1] E. Balsa-Canto, A.A. Alonso, and J.R. Banga. Computational procedures for optimal experimental design in biological systems. *IET Systems Biology*, 2(4):163–172, 2008.
- [2] E. Balsa-Canto, A.A. Alonso, and J.R. Banga. An iterative identification procedure for dynamic modeling of biochemical networks. *BMC Systems Biology*, 4:11, 2010.
- [3] E. Balsa-Canto, J. R. Banga, and A. A. Alonso. An optimal identification procedure for model development ins systems biology: Applications in cell signalling. In F. Allgöwer and M. Reuss, editors, *Foundations of Systems Biology in Engineering*, pages 51–56, 2007.
- [4] E. Balsa-Canto and J.R. Banga. Advanced model identification using global optimization. *Tutorial at the 9th International Conference on Systems Biology. ICSB. Goteborg, Sweden.*, 2008.
- [5] E. Balsa-Canto and J.R. Banga. AMIGO: A model identification toolbox based on global optimization. In *Computer Applications in Biotechnology*, Leuven, 2010.
- [6] E. Balsa-Canto, M. Peifer, J.R. Banga, J. Timmer, and C. Fleck. Hybrid optimization method with general switching strategy for parameter estimation. *BMC Systems Biology*, 2:26, DOI:10.1186/1752-0509-2-26, 2008.
- [7] E. Balsa-Canto, M. Rodriguez-Fernandez, A. A. Alonso, and J. R. Banga. Computational design of optimal dynamic experiments in systems biology: a case study in cell signaling. In M. Cánovas, J.L. Iborra, and A. Manjón, editors, *Understanding and Exploiting Systems Biology in Bioprocesses and Biomedicine*, pages 103–117. Fundación CajaMurcia, 2006.
- [8] J. R. Banga and E. Balsa-Canto. Parameter estimation and optimal experimental design. *Essays in Biochemistry*, 45:195–210, 2008.
- [9] H.G. Bock. *Recent advances in parameter identification techniques for ordinary differential equations.*, pages 95–121. Numerical Treatment of Inverse Problems in Differential and Integral Equations. Deuffhard P. and Hairer E., Editors. Birkhäuser. 1983.
- [10] R. Brun and P. Reichert. Practical identifiability analysis of large environmental simulation models. *Water Resources Res.*, 37:1015–1030, 2001.
- [11] J Dréo, A Petrowski, E Taillard, and P Siarry. *Metaheuristics for hard optimization. Methods and case studies.* Springer, 2006.
- [12] J. A. Egea, E. Balsa-Canto, M.G. Garcia, and J. R. Banga. Dynamic optimization of nonlinear processes with an enhanced scatter search method. *Ind. & Eng. Chem. Res.*, 48(9):4388–4401, 2009.

- [13] W. R. Esposito and C. A. Floudas. Global optimization of nonconvex problems with differential-algebraic constraints. In *“European Symposium on Computer Aided Process Engineering-10”, S. Pierucci (Ed.), Elsevier, Amsterdam, The Netherlands*, pages 73–78, 2000.
- [14] X. J. Feng and H. Rabitz. Optimal identification of biochemical reaction networks. *Biophys. J.*, 86(3):1270–1281, 2004.
- [15] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, Inc., New York, 2nd edition, 1987.
- [16] C.A. Floudas. *Deterministic Global Optimization: Theory, Methods and Applications*. Kluwer Academics, The Netherlands, 2000.
- [17] K.G. Gadkar, R. Gunawan, and F.J. Doyle III. Iterative approach to model identification of biological networks. *BMC Bioinformatics*, 6:155, 2005.
- [18] M.R. Garcia. *Identification and real time optimisation in the food processing and biotechnology industries*. PhD thesis, University of Vigo, Spain, 2008.
- [19] C. Y. Gau and M. A. Stadtherr. Reliable nonlinear parameter estimation using interval analysis: Error in variable approach. *Comp. & Chem. Eng.*, 24:631–637, 2000.
- [20] A. Hodgkin and A. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.*, 117:500–544, 1952.
- [21] A. Hoffmann, A. Levchenko, M.L. Scott, and D. Baltimore. The I κ B-NF- κ B signaling module: temporal control and selective gene activation. *Science*, 298:1241–1245, 2002.
- [22] M. Joshi, A. Seidel-Morgenstern, and A. Kremling. Exploiting the bootstrap method for quantifying parameter confidence intervals in dynamical systems. *Metabolic Engineering*, 8:447–455, 2006.
- [23] A. Kremling and J. Saez-Rodriguez. Systems biology - an engineering perspective. *J. Biotechnol.*, 129:329–351, 2007.
- [24] C. Kreutz and J. Timmer. Systems biology: experimental design. *FEBS J.*, 276:923–942, 2009.
- [25] E.G. Lee, D.L. Boone, S. Chai, S.L. Libby, M. Chien, J.P. Lodolce, and A. Ma. Failure to regulate TNF-induced NF- κ B and cell death responses in A20-deficient mice. *Science*, 289:2350–2354, 2000.
- [26] Y. Lin and M. A. Stadtherr. Deterministic global optimization for parameter estimation of dynamic systems. *Ind. & Eng. Chem. Res.*, 45:8438–8448, 2006.
- [27] T. Lipniacki, P. Paszek, A.R. Brasier, B. Luxon, and M. Kimmel. Mathematical model of NF κ B regulatory module. *J. Theor. Biol.*, 228:195–215, 2004.
- [28] L. Ljung. *System identification: Theory for the user*. Prentice Hall, New Jersey, 1999.
- [29] J.C.W. Locke, A.J. Millar, and M.S. Turner. Modelling genetic networks with noisy and varied experimental data: the circadian clock in arabidopsis thaliana. *Journal of Theoretical Biology*, 234:383–393, 2005.
- [30] Sugimoto M, Kikuchi S, and Tomita M. Reverse engineering of biochemical equations from time-course data by means of genetic programming. *BioSystems*, 80:155–164, 2005.

- [31] P. Mendes. *Foundations of Systems Biology*, chapter Modelling large biological systems from functional genomic data: Parameter estimation. MIT Press, kitano, h. edition, 2001.
- [32] P. Mendes and D.B. Kell. Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. *Bioinformatics*, 14(10):869–883, 1998.
- [33] C.G. Moles, P. Mendes, and J.R. Banga. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome Research*, 13:2467–2474, 2003.
- [34] S. G. Nash and A. Sofer. *Linear and Nonlinear Programming*. McGraw- hill, 1996.
- [35] P.M. Pardalos, H.E. Romeijna, and H. Tuyb. Recent developments and trends in global optimization. *J Comp and App Math*, 124:209–228, 2000.
- [36] M. Peifer and J. Timmer. Parameter estimation in ordinary differential equations for biochemical processes using the method of multiple shooting. *IET Systems Biology*, 1:78–88, 2007.
- [37] J. Pinter. *Global Optimization in Action. Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications*. Kluwer Academics, Netherlands, 1996.
- [38] P.K. Polisetty, E.O. Voit, and E.P. Gatzke. Identification of metabolic system parameters using global optimization methods. *Theor. Biol. & Med. Mod.*, 3:4, 2006.
- [39] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics*. Springer-Verlag, New York, U.S.A., 2000.
- [40] M. Rodriguez-Fernandez, J. A. Egea, and J.R. Banga. Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems. *BMC Bioinformatics*, 7:483, 2006.
- [41] M. Rodriguez-Fernandez, P. Mendes, and J.R. Banga. A hybrid approach for efficient and robust parameter estimation in biochemical pathways. *Biosystems*, 83(2-3):24, 2006.
- [42] K. Schittkowski. *Numerical Data Fitting in Dynamical Systems - A Practical Introduction with Applications and Software*. Kluwer Academic, 2002.
- [43] GAF Seber and CJ Wild. *Nonlinear regression*. Wiley series in Probability and Mathematical Statistics. John Wiley & Sons, USA., 1989.
- [44] N.A.W. van Riel. Dynamic modelling and analysis of biochemical networks: Mechanism-based models and model-based experiments. *Brief. Bioinform.*, 7(4):364–374, 2006.
- [45] V. S. Vassiliadis. *Computational Solution of Dynamic Optimization Problems with General Differential-Algebraic Constraints*. PhD thesis, Imperial College, University of London, London, U.K., July 1993.
- [46] E. Walter and L. Pronzato. *Identification of Parametric Models from Experimental Data*. Springer, Masson, 1997.