Virtual model-based continuous bioprocessing
A beginner's guide to the library

Mirko Pasquini and Kevin Colin

February 14, 2025

1 Framework and introduction

Let us first define a framework within which the library operates. We will consider steady-state experiments in continuous perfusion, i.e. for which the following steady-state mass-balance equation holds

$$Xq - Fc + Fc_{in} = 0 \tag{1}$$

where

- $q \in \mathbb{R}^{n_q}$ [1/day] is a vector of extracellular metabolites uptake-secretion rates;
- $c \in \mathbb{R}_+^{n_c}$ [mmol/L] is a vector of extracellular metabolites steady-state concentrations;
- $c_{in} \in \mathbb{R}_+^{n_c}$ [mmol/L] is a vector of concentrations in the feed medium (continuously fed in the bioreactor);
- $F \in \mathbb{R}_+$ [1/day] is the perfusion rate;
- $X \in \mathbb{R}_+$ [mmol/L] is the viable cell density in the bioreactor;

While this equation holds at steady-state for any continuous perfusion system, we will consider the case in which q is modelled as a function of the steady-state concentrations c and a set of kinetic parameters Θ, i.e. $q = q(c; \Theta)$. Given this parametric model we are able to consider (1) as a digital twin of a physical bioreactor, and we are able to simulate experiments, optimize the bioprocess, and iteratively update the model with new experimental data. This library is an attempt to collect the Matlab code that allows to perform these tasks.

Remark 1. *Any vector, both in this guide and in the code, is considered to be a column unless differently stated. Please be sure that this convention is maintained in the code, to avoid unexpected errors.*

2 Models

When we refer to a "model" we refer to a parametric model of the kinetic of the uptake-secretion rates. We assume the model to follow this structure

$$q(c; \Theta) = A_{mac} w(c; \Theta) \tag{2}$$

where $q \in \mathbb{R}^{n_q}$ is the vector of uptake-secretion rates of the considered metabolites, $w \in \mathbb{R}^{n_w}$ is the vector of macroreaction rates (see e.g. [1]), $A_{mac} \in \mathbb{R}^{n_w \times n_q}$ is called the macroreaction stoichiometric matrix, and describes how each macroreaction participates in each uptake-secretion rate. For the macroreaction rate w_i we will consider the following Monod-type kinetic

$$w_i(c; \Theta_i) = w_i^{max} \prod_{j=1}^{n_c} \underbrace{\frac{c_j}{c_j + k_{i,j}^{act}} \cdot \frac{1}{1 + k_{i,j}^{inh} c_j}}_{\text{Modulation function}} \tag{3}$$

where w_i^{max} is the maximal rate of the macroreaction w_i, $k_{i,j}^{act}$ is the activation coefficient describing the activation effect that metabolite j (with its steady-state concentration c_j) has on the rate w_i, while

$k_{i,j}^{inh}$ is the inhibition coefficient describing the inhibition effect that metabolite j (with its steady-state concentration c_j) has on the rate w_i. From this general double-component Monod structure we can also describe only-activation, only-inhibition or neutral effect, of metabolite j on the rate w_i, by respectively setting $k_{i,j}^{inh} = 0$, $k_{i,j}^{act} = 0$, or $k_{i,j}^{inh} = k_{i,j}^{act} = 0$. With this model structure, it follows that the rate q_k is then modelled as

$$q_k(c;\Theta) = \sum_{i=1}^{n_w} [A_{mac}]_{k,i} \cdot w_i^{max} \prod_{j=1}^{n_c} \frac{c_j}{c_j + k_{i,j}^{act}} \cdot \frac{1}{1 + k_{i,j}^{inh} c_j} \tag{4}$$

where $[A_{mac}]_{k,i}$ is the element (k,i) of the matrix A_{mac}. We can the see that, also given the steady-state mass-balance equation (1), a model is completely characterized by the stoichiometric matrix A_{mac}, and the set of kinetic parameters Θ (i.e. the maximal rates for all the macrorates, as well as all the activation and inhibition parameters).
In the next section we discuss how the kinetic parameters can be represented in a compact and useful way.

2.1 Parameters matrix and vector - a representation

For notational purposes, in the following we will define

$$\theta_{i,1} := w_i^{max} \tag{5}$$
$$\theta_{i,2j} := k_{i,j}^{act} \tag{6}$$
$$\theta_{i,2j+1} := k_{i,j}^{inh} \tag{7}$$

So that we can conveniently collect all the kinetic parameters in the matrix

$$\Theta = \begin{bmatrix} \theta_{1,1} & \dots & \theta_{1,2n_c+1} \\ \vdots & \ddots & \vdots \\ \theta_{n_w,1} & \dots & \theta_{n_w,2n_c+1} \end{bmatrix} = \begin{bmatrix} w_1^{max} & k_{1,1}^{act} & k_{1,1}^{inh} & \dots & k_{1,n_c}^{act} & k_{1,n_c}^{inh} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{n_w}^{max} & k_{n_w,1}^{act} & k_{n_w,1}^{inh} & \dots & k_{n_w,n_c}^{act} & k_{n_w,n_c}^{inh} \end{bmatrix} \tag{8}$$

It is useful in the implementation to consider also the parameters vector

$$\theta = \begin{bmatrix} \theta_{1,1} \\ \vdots \\ \theta_{1,2n_c+1} \\ \vdots \\ \theta_{n_w,1} \\ \vdots \\ \theta_{n_w,2n_c+1} \end{bmatrix} \tag{9}$$

and the two operators **vector** : $\mathbb{R}^{n_w \times (2n_c+1)} \to \mathbb{R}^{n_w \cdot (2n_c+1)}$ and **matrix** : $\mathbb{R}^{n_w \cdot (2n_c+1)} \to \mathbb{R}^{n_w \times (2n_c+1)}$ to convert one format to the other, i.e.

$$\theta = \mathbf{vector}(\Theta) \tag{10}$$
$$\Theta = \mathbf{matrix}(\theta) \tag{11}$$

In the library these two operators are implemented in the functions
`construct_vector_from_parameters_matrix` and
`construct_matrix_from_parameters_vector` (which can be found in the folder `utils`).

2.2 Model structure

A model structure is a Matlab structure that completely describes a model through the macroreaction stoichiometric matrix A_{mac}, the matrix of kinetic parameters Θ, the flow rate F and the viable cell density X. The model structure is used by many functions in the libraries, and in particular has the following fields:

- `model.F` : flow rate of perfusion [1/day] (we usually consider `model.F = 1`)

- `model.Xv` : viable cell density in the bioreactor at steady-state [mmol/L] (we usually consider `model.Xv = 10.81*60`)
- `model.Amac` : macroreaction stoichiometric matrix, obtained as $A_{ext}E$, where A_{ext} is the extracellular stoichiometric matrix and E is the matrix of EFMs (see e.g. [1]);
- `model.theta_matrix` : matrix of parameters. See Section 2.1.

2.3 Loading a model

To load a model the function `load_model` is used. The function will take as input Excel files containing all the information regarding a particular model. Check a model example in the folder `models` for the files structure, and follow that exactly, if you want to use your own models, to guarantee compatibility and avoid any issue. Each model is contained in a different folder (e.g. `T2024-1-Net2-VN4`) which contains a parameters Excel file and a supplementary file containing the macroreaction stoichiometric matrix.

Remark 2. *In the folder you will notice that there is a* `parameters` *file and a* `reordered_parameters` *one. The reordered one is the one that will be loaded in the model, for which the metabolites are in the same order as the stoichiometric matrix. See* ***Q.1.*** *in Section 6.*

Remark 3. *The supplementary file contains different sheets other than Sheet 1 containing the macroreaction stoichiometric matrix. These sheets are not used by the library, and will probably be removed soon.*

3 Identification toolbox

In this context with identification we refer to the steps that allow to find the subset of EFMs $\mathbf{E}$ and the parameters matrix Θ, completely characterizing the metabolic model, starting from raw measurements data of steady-state concentrations and uptake-secretion rates (possibly together with transcriptomics information). The data will be contained in the library folder `data`. The code-base relative to identification will be in the library folder `macroscopic-modeling`.

Important remark: additionally to some files required by the toolbox (see Section 3.2), all the variables required to configure the modeling options are gathered in the Matlab script `MAIN_MODELING.m`. The user should only interact with this script. Only the values of the variables of this file should be modified and the variable names must stay the same! No variables should be deleted.

3.1 Brief description of the methodology

From the measurements of the cell specific consumption/production rates $\hat{q}(k)$ and the concentrations of several extracellular metabolites $\hat{c}(k)$, we wish to determine the most relevant macroscopic stoichiometric matrix A_{mac} and the kinetic parameters Θ of (4) which best fit the data.

The macroscopic stoichiometric matrix is expressed as $A_{mac} = A_{ext}E$ with A_{ext} the extracellular stoichiometric matrix and E the matrix whose columns correspond to elementary flux modes. While A_{ext} is known, the EFM matrix E is the one that is estimated from the data.

In a first step, the idea is to estimate the most relevant EFMs which fit the data q_{meas} by removing the Monod model structure for the macroscopic rates in w. We simply consider w as a free non-negative variable for each data of $q(k)$. The following optimization problem is solved

$$\min_{\substack{E \\ w(1),\cdots,w(N)}} \sum_{k=1}^{N} ||q(k) - A_{ext}Ew(k)||_2^2 \tag{12}$$

$$\text{Subject to} \quad E \text{ subset of EFMs} \tag{13}$$

$$w(k) \geq 0 \text{ for each } k = 1, \cdots, N \tag{14}$$

This problem can be solved efficiently with the column generation algorithm [1]. At the end, we have an optimal subset E of n_w EFMs and the corresponding macroscopic rate vector data $w^*(1), \cdots, w^*(N)$.

In a second step, we estimate a Monod model for each macroscopic rate with the data $w^*(1), \cdots, w^*(N)$. For each $i = 1, \cdots, n_w$. This means that, for each $i = 1, \cdots, n_w$, we identify the kinetic parameters w_i^{max}, $k_{i,1}^{act}$, $k_{i,1}^{inh}$, $k_{i,2}^{act}$, $k_{i,2}^{inh}$, $\cdots$ k_{i,n_c}^{act} and k_{i,n_c}^{inh} from the data $\{w_i^*(k)\}_{k=1}^N$ where $w_i^*(k)$ is the i-the element of the vector $w^*(k)$.

The originality of the approach coded in the identification toolbox is that all the measured extracellular concentrations (some metabolites can be excluded by the user) are inputs of the Monod models as double-components and we use the data to estimate all the activation and inhibition parameters.

However, this leads to an increase of the number of kinetic parameters to be estimated, which can lead to overfitting issues. In the code, l_1 norm regularization is used to fit the data $w^*(1), \cdots, w^*(N)$ while setting several parameters to 0 by solving

$$\min_{\Theta_i} \sum_{k=1}^{N} (w_i^*(k) - w_i(c(k), \Theta_i))^2 + \lambda \sum_{j=1}^{n_c} (k_{i,j}^{act} + k_{i,j}^{inh}) \tag{15}$$

where $\lambda \geq 0$ is the regularization parameter. The larger λ is, the more identified kinetic parameters equal to 0. For selecting the value of λ, leave-one-out cross-validation is implemented in the code. Starting with a grid of values specified by the user, the dataset of size N is partitioned such that $N-1$ data points are used for training, while the remaining data point serves as the validation set. The parameter vector Θ_i of each macroscopic rate w_i is determined using the $N-1$ training points by solving (15) and the squared error is computed between the model $q(c, \Theta)$ and the validation measurement. This is repeated N times, each time using a different data point as the validation set and the N validation squared errors are averaged. This procedure is performed for every value of λ in the grid. The algorithm selects the optimal λ as the one that minimizes the average validation error. For that value of λ, the identification problem (15) is solved one last time by using the N data as training data.

While regularization prevents overfitting, it also comes with a bias for the data fit. In order to reduce the bias effect, we add another identification step. The idea is to adjust the non-zero kinetic parameters by leveraging Bayesian estimation. The motivation of using Bayesian estimation is to add prior information on the kinetic parameters, in the form of a prior density function called prior, which can prevent overfitting. The available prior knowledge we have on the activation and inhibition parameters is that they are nonnegative and can have different orders of magnitude. Consequently, a log-normal prior distribution is considered for each non-zero activation and inhibition parameters. The mean and standard deviation (called hyperparameters of the prior distributions) must be tuned using the data. This is done in the code by using the Expectation Maximization (EM) algorithm This algorithm iteratively tunes the hyperparameter such that the log-marginal likelihood distribution is maximized. However, this distribution is often intractable to be computed and so sampling techniques are required in order to get some activation and inhibition parameter samples in order to properly tune the hyperparameter. The sampling technique used in the toolbox is called Metropolis-Hastings within Gibbs sampling.

Since each modulation function was chosen as a double component for model flexibility, we expect that many of them are activation, inhibition, or neutral effects. For this purpose, a last step is done which consists in checking if each identified modulation function can be reduced into an activation, an inhibition, or a neutral effect. Two tests are performed on each identified modulation function model evaluated with the training data

- Test 1 : we compute the maximum and minimum among the modulation function values computed the training data. If the relative variation in percentage between the minimum and the maximum is **below** an user-defined threshold (in percentage), then the modulation function can be considered a neutral effect (constant function equal to 1).
- Test 2 : if the previous test failed, another one is conducted in order to verify if the modulation function can be reduced into an activation or an inhibition effect. For this purpose, we compute the activation model and the inhibition model which optimally fit the modulation function model evaluated with the training data. If one of these models provides a reasonable fit, the modulation function is reduced to the corresponding effect.

3.2 File and software requirements

The identification toolbox requires several files :

- an excel file containing the cell specific consumption and secretion rates data $q(k)$. In the excel file, the first column corresponds to each experimental condition, the second column to the measurements instants and the remaining columns the values $q(k)$ for each the measured metabolites, one column per metabolite. See the fake data example of the toolbox in order to understand the required format of this excel file.
- an excel file containing the concentration data $c(k)$. In the excel file, the first column corresponds to each experimental condition, the second column to the measurements instants and the remaining columns the values $q(k)$ for each the measured metabolites, one column per metabolite. See the fake data example of the toolbox in order to understand the required format of this excel file.
- an excel of .mat file which contains the metabolic network model. See the fake data example of the toolbox in order to understand the required format of this excel file or the.mat file.

For large scale metabolic network, it is recommended to use MOSEK for the column generation. It is included as an option in the library. It is also recommended to use parallel computing, especially in the case where many EFMs are computed. This option has been coded in the library.

Important remark : only the versions after 10.0 should be installed for MOSEK. In addition, it is important to add the pathway to the folder `r2017aom` in `MAIN_MODELING.m`.

3.3 Outputs of the identification toolbox

At the end of the call of the identification toolbox, several files are saved in a subfolder specified by the user with the variable `name_save_file` which is placed in the folder `models`. A mat file with the same name as the file name in `name_save_file` which contains all the variables chosen by the user in `MAIN_MODELING.m`, the identified kinetic parameters, the computed EFMs, etc. Furthermore, four excel files are created :

- an excel file with the same name as `name_save_file` whose different sheets contain various model properties:
  - Sheet 1 : it contains the macroscopic stoichiometric matrix (equal to the extracellular stoichiometric matrix multiplied by the EFM matrix E).
  - Sheet 2 : it gathers all the identified parameters where each row corresponds to a given macroscopic rate. The first column gathers all the maximal rate constants w_i^{max}. The other columns alternate between the activation and inhibition parameters $k_{i,j}^{act}$ and $k_{i,j}^{inh}$ for each metabolite concentration.
  - Sheet 3 : in the first column, it contains all the identified macroscopic reactions where all the extracellular metabolites which are involved. For each macroscopic reaction, the second column gives the different reactions which are participating.
  - Sheet 4 : in the first column, it contains all the identified macroscopic reactions where only the MEASURED extracellular metabolites which are involved. For each macroscopic reaction, the second column gives the different reactions which are participating.
  - Sheet 5 : the relative errors computed during Step 4 are written in Sheet 5.
  - Sheet 6 : the absolute errors computed during Step 4 are written in Sheet 6.
  - Sheet 7 : the cell specific rates computed by the kinetic models and evaluated with the training and prediction data are store in this sheet.
- an excel file `supplementary.xls` which contains the macroscopic stoichiometric matrix. **This file is required by the model-based optimization toolbox**.
- an excel file `supplementary_ext.xls` which contains the macroscopic stoichiometric matrix, including the extracellular metabolites which are note measured. **This file is required by the model-based optimization toolbox**.
- an excel file `parameters.xls` which contains the identified Monod parameters. **This file is required by the model-based optimization toolbox**.

3.4 Partition of the identification toolbox

The identification toolbox follows four main steps/functions which should be done in the following order :

- Step 1: reading the input files and data treatment such as smoothing, normalization, averaging.
- Step 2: Elementary Flux Modes (EFMs) identification.
- Step 3: Monod identification of the macroscopic rates.
- Step 4: treatment of the results (plots, saving files for optimization, etc.)

3.5 Declaration of the necessary variables in MAIN_MODELING.m

The fours steps require several variables and options set by the user. For this purpose, a .file called `MAIN_MODELING.m` is included in the library where all the required variables and their effect in the modeling process are detailed. Among many things, the user specifies the directory where the required files are located, it chooses the EFM computation method, it can select if he wants to average the data per experimental condition (useful for reducing the modeling error with steady-state data), normalize or smooth the data, etc.

The code `MAIN_MODELING.m` runs at the end the function `run_all_the_steps.m` This is the mastercode which will call successively the functions dedicated to each step in order to perform the modeling as described in the methodology in Section 3.1.

3.6 Step 1: data treatment

The code related to the first step is called `Step_1_file_treatment`. This file loads all the excel files required for the identification. The user must indicate in `MAIN_MODELING.m` the directory where the file is located in the variables

- `directory_file_cell_specific_rate_data_qext` for the excel file containing the cell specific rate data $q(k)$. See the file `fake_data_example_qext.xls` in the folder /data/K-net data in order to analyze the required format for this file.
- `directory_file_concentration_data_cext` for the excel file containing the concentration data $c(k)$. See the file `fake_data_example_cext.xls` in the folder /data/K-net data in order to analyze the required format for this file.
- `directory_file_metabolic_network` for the excel or mat file containing the metabolic network model. See the file `fake_data_example_metabolic_network.xls` in the folder `/data/K-net data` in order to analyze the required format for this file.

For the files containing the data $q(k)$ and $c(k)$, it identifies every different experimental condition and the corresponding data. Regarding the metabolic network, it splits the stoichiometric matrix into its extracellular and intracellular counterparts $\mathbf{A}_{ext}$ and $\mathbf{A}_{int}$..

Finally, it also smooths, averages and normalizes the data according to the user choice indicated in `MAIN_MODELING.m`. It also splits the data between training and prediction if the user specifies a condition as a prediction data set in the variable `prediction_media` in `MAIN_MODELING.m`.

3.7 Step 2: Elementary Flux Modes (EFMs) identification

The data-driven EFM computation is performed with the function `Step_2_EFM_computation.m`.

Before that, a first initial test of the metabolic network is performed to check if it is adequate for modeling purposes. Using the data $q(k)$, it first computes an optimal flux vector $v(k)$ such that the following optimization problem is minimized

$$\min_{v(1),\cdots,v(N)} \sum_{k=1}^{N} ||q(k) - A_{ext}v(k)||_2^2 \tag{16}$$

$$\text{Subject to} \quad A_{int}v(k) = 0 \text{ for each } k = 1, \cdots, N \tag{17}$$

$$v_{irrev}(k) \geq 0 \text{ for each } k = 1, \cdots, N \tag{18}$$

where A_{int} is the intracellular stoichiometric matrix and $v_{irrev}(k)$ the subvector of the flux vector containing only the irreversible fluxes. Then, for each metabolite whose cell specific rate is measured, we compute the fit. If at least one metabolite has a fit less than 50%, the code warns the user that either the data are too noisy or the metabolic network is too simple for modeling the main features of the data.

If the network is suited or the user still pursues the modeling despite the warning, then the EFMs are computed using column generation. Three different methods have been implemented and the choice is specified by the user with the char variable `computation_method_EFM` in `MAIN_MODELING.m`

- Global (`computation_method_EFM = 'global'`) : the EFMs are computed such that they fit all the data simultaneously.
- Sequential (`computation_method_EFM = 'sequential'`) : this method computes the EFM by fitting the data sequentially. We first compute the EFMs and macro-rates which fit the first measurement. We then try to fit the second measurement with this subset of EFMs. If the data are not optimally fitted, we compute new EFMS that we add to the first subset of EFMs. We repeat this procedure until the last measurement.
- Union (`computation_method_EFM = 'union'`) : this method computes the EFM by fitting each measurement separately. Therefore, N subsets of EFMs are computed. At the end, we compute the union of the N subsets of EFMs.

3.8 Step 3: Monod identification of the macroscopic rates

The Monod parameter matrix Θ is identified from the macroscopic rate data $w^*(k)$ obtained during Step 2 and the concentration $c(k)$ data. The Matlab function handling the Monod identification of the rates is called `Step_3_Monod_kinetic_identification_of_macroscopic_rates.m`. During this step, the following tasks are performed for each macroscopic rate w_i $(i = 1, \cdots, n_w)$

- Step 3.a: model selection with l_1 regularization if chosen. In `MAIN_MODELING.m`, the user specifies if he wants to use regularization or not by setting the boolean variable `regularization` to 1 for regularization, 0 otherwise. Regularization can be helpful in reducing the complexity of the model

  If regularization is chosen, then a set of non-negative regularization parameter λ must be indicated in the vector `lambda_grid`. In both cases, the optimization problem to be solved is non-convex which can lead to local optimum computation. In order to increase the chance of global optimum computation, the code proposes a multi-start strategy where the optimization problem is solved several times with a different initial point. The number of of multi-start initial points can be specified by the user with the variable `number_multistart_points`.
- Step 3.b: Bayesian estimation in order to compensate for the bias introduced by the regularization or improve the initial estimate in the case when no regularization is selected. The Bayesian estimation algorithm (isolated in the Matlab function `bayesian_estimation_Monod_kinetics.m`) requires several parameters such as the number of iterations `number_of_max_iterations_EM`, the burn-in period `burn_in_period_EM` and number of samples `number_samples_per_iteration_EM` used for the hyperparameter tuning, the number of sampling trials for the Metropolis-Hastings algorithm `number_trials_sampling_EM` and the additive perturbation parameter which is used in order to explore more the parameter space, increasing the chance to compute the global least-squares minimum.
- Step 3.c: modulation function reduction. As aforementioned, each modulation function is considered as a double-component at the beginning of the modeling. Some of them might be reduced into a neutral effect, activation or inhibition effect after Bayesian estimation without degrading the data fit too much. A first test is done in order to check if the modulation function can be reduced into a neutral effect (both the activation and inhibition parameters are set to 0). For this purpose, the code computes the relative variation in percentage between the minimum and maximum if the identified modulation function evaluated with the training data. If this relative error is <u>less than</u> a threshold in percentage (always non-negative) defined by the user (in the variable `threshold_fit_neutral_effect`), then the modulation function is reduced to a neutral effect. If not, a second test is performed in order to verify if it can be reduced into an activation or an inhibition effect. For this purpose, two least-squares optimizations are run in order to compute the optimal activation and inhibition functions which fit the identified modulation function

evaluated at the training concentration data. Then, the effect with the highest fit (in percentage) is selected. If that fit is above a threshold in percentage defined by the user in the variable `threshold_fit_activation_or_inhibition`, then the modulation function is reduced into the corresponding (activation or inhibition) effect.

3.9 Step 4: treatment of the results

Some final treatments of the modeling results (computation of modeling errors, plotting results, computing the macroscopic reaction, saving the results) are done in Step 4. The Matlab function taking care of this step is called `Step_4_Monod_kinetic_identification_of_macroscopic_rates.m`. For the variables required by this function, the user must indicate in `MAIN_MODELING.m`

- the number of plots to put per row (`number_of_plots_per_rows`) and column (`number_of_plots_per_columns`) in the Matlab figure which compares the output of the kinetic model and the data. Each plot is dedicated to one measured metabolite.
- the name (variable `name_save_file`) of the excel file where the information about the model (macroscopic reactions, kinetic parameters, errors, etc) is stored.

Several plots are displayed which show the training and prediction data for the different measured cell specific rates. They are compared to the output of the model without kinetics (with only the EFM computation and no parametric expression for the rates) and with the model with kinetics.
This code displays two tables in the command window

- the first table gives the relative error of each measured cell specific rate for the model without kinetics (with only the EFM computation and no parametric expression for the rates) and for the model with identified kinetics.
- the second table gives the absolute error of each measured cell specific rate for the model without kinetics (with only the EFM computation and no parametric expression for the rates) and for the model with identified kinetics.

4 Virtual-experiments

A virtual experiment is a simulation of the bioreactor behavior given a metabolic model (see previous Section), when a certain feed medium is chosen. In particular if we look back at the steady-state mass-balance equation

$$Xq(c;\Theta) - Fc + Fc_{in} = 0 \tag{19}$$

when referring to "simulating an experiment" or "running a virtual experiment" we refer to finding the value of c satisfying the above system of equations, for a given choice of the feed medium composition c_{in}. Since the system of equation (19) is non-linear (and non-convex) in c, and we also have a (physical) non-negativity constraint on the concentrations, to find the vector of steady-state concentrations we solve the following

$$\min_{c\geq 0} \|Xq(c;\Theta) - Fc + Fc_{in}\|_2^2 \tag{20}$$

This approach is implemented in the function `run_virtual_experiment`, which requires, other than the medium composition c_{in} and the kinetic model that the user choose to use, the initial concentration c_0 from which the solver is initialized, and the indexes of the coupled metabolites.

Remark 4. *Given the non-convexity of problem* (20)*, local minima are possible and, depending on the starting condition c_0, the simulation might converge to a different steady-state concentration vector. This is something I am still trying to properly understand and fix.*

4.1 Coupled and uncoupled metabolites

If we look at Eq. (19), we notice how this is a system of equations of the variables c (as c_{in} is decided a priori). The system of equations is non-linear, given the function $q(c;\theta)$ whose description is given in (4). However it is possible that some metabolites do not influence any uptake-secretion rates, i.e. their concentration does not appear in any of the expressions $q_k(c;\theta)$. These metabolites are called uncoupled, and they do not need to be considered in the solution of Eq. (19), as their concentration can be evaluated a posteriori once the other metabolites concentrations are found. We denote (intuitively) the metabolites that are not uncoupled as coupled metabolites.

This definition is important as it makes the solution of the mass-balance equation more efficient, and the user should provide (in many library functions) the indexes of the coupled metabolites in the vector of extracellular metabolites. The user can execute the function `extract_coupled_met_indices` in the `utils` folder to extract the vector of indexes corresponding to coupled metabolites. This function compares the list of metabolites labels in the kinetic parameters matrix and the list of metabolites labels in the stoichiometric matrix and it returns which ones of the latter are present in the former, and in which order.

5 Model-based optimization

One application of interest of this framework is that of model-based medium optimization. With medium optimization we refer to the optimization of the concentrations of some medium components (i.e. c_{in}) so that the harvest rate of some product of interest (e.g. mAb) is maximized at steady-state, while certain toxic byproducts concentrations are limited. A model-based approach differs from a classical design of experiments (DOE) from the fact that the model is used to predict the optimal medium composition, with the aim of reducing the number of experiments needed for process development.

All the code for model-based optimization is contained in the library folder `model-based-optimization`. Below an introduction of the different approaches considered (for now only nominal optimization).

5.1 Nominal optimization

With the goal set above, it seems natural to formulate the problem of model-based medium optimization as a constrained optimization problem

$$
\begin{aligned}
\text{maximize}_{c_{in},c} \quad & \tilde{\Phi}(c_{in}, c, q(c,\theta)) && && (21)\\
\text{subject to} \quad & \underline{c_{in,i}} \le c_{in,i} \le \overline{c_{in,i}}, && \forall i \in \{1,\dots,n_c\} && (22)\\
& \underline{c_i} \le c_i \le \overline{c_i}, && \forall i \in \{1,\dots,n_c\} && (23)\\
& \underline{q_i} \le q_i(c,\theta) \le \overline{q_i}, && \forall i \in \{1,\dots,n_q\} && (24)\\
& a^\top c_{in} \le b && && (25)\\
& X_v q_i(c,\theta) - F c_i + F c_{in,i} = 0 && \forall i \in \{1,\dots,n_c\} && (26)
\end{aligned}
$$

where (22) – (24) are bounds on the metabolites concentrations in the medium and the bioreactor at steady-state and on the uptake/secretion rates (with $\underline{c_{in}}$, $\underline{c}$, $\underline{q}$, $\overline{c_{in}}$, $\overline{c}$ and $\overline{q}$ being respectively their user-defined lower and upper bounds), while (25) is a constraint reflecting medium preparation limitations due to solubility. Both c_{in} and c are considered as decision variables of the problem above, however it is only the medium composition c_{in} that is chosen in any given experiment, while c will be a consequence of this choice. This is considered by the mass-balance equation constraint (26). The objective function (21) should reflect the amount of harvested product of interest. If we consider the metabolite indexed by k to be the one whose harvested amount we want to maximize, we can consider

$$
\tilde{\Phi}(c_{in}, c, q(c,\theta)) = (F - \mu(c,\theta)) \cdot q_k(c,\theta) \tag{27}
$$

where μ is the growth-rate, the model of which is available as one of the uptake-secretion rates described by the vector q.

In the library this problem is solved by calling the function
`nominal_optimization_given_metabolic_model`. Please refer to the documentation of that function for the needed input arguments and the structure of the output.

Remark 5. *It is usually the case that only a subset of the metabolites concentrations in the medium have to be designed. These metabolites are called decision metabolites, and their indexes (in the vector of all extracellular metabolites) need to be specified in the optimization problems. Despite an alternative to this might be to add equality constraints in the optimization (i.e. all the non-decision metabolites components in the medium are forced to be equal to their fixed value), doing so is highly inefficient and can bring computational issues in the solution of the optimization problem.*

Remark 6. *The user might be interested, instead of maximizing the harvest of a certain product of interest, to either maximize or minimize a given uptake-secretion rate of interest, i.e. to consider respectively the objective functions*

$$\tilde{\Phi}(c_{in}, c, q(c, \theta)) = q_k(c, \theta), \quad or \quad \tilde{\Phi}(c_{in}, c, q(c, \theta)) = -q_k(c, \theta)$$

This can be obtained by specifying the desired `objective-type` *in the library function* `nominal_optimization_given_metabolic_model` *(specifically these types are* `'harvest'`, `'rate-max'` *or* `'rate-min'`*).*

5.2 Distance of optimized media from the available data

When an optimized medium is designed, it is important to check its distance from the media already tested experimentally. An optimized medium which is very close to a previously tested medium will not provide an informative solution (e.g. it is possible that the model predicts a close value to the previous experiment, or a value that is not reliable).
This can be evaluated using the function `evaluate_distance_medium_from_dataset`, which will evaluate the distance of a given medium from each medium in a provided matrix of previously tested media (i.e. the available experimental data). The returned distance will be the smallest among all these distances. The user can decide the type of norm used to evaluate the distances (e.g. generally 1-norm or 2-norm) and has the option to normalize the media composition to reduce the effect of having different possible ranges for different metabolites in the media.

5.3 Sensitivity analysis with respect to medium composition perturbations

It might be of interest to check how the predictions of certain quantities, i.e. steady-state concentrations, rates and harvest rate (as defined in Eq. (27)), vary when an optimal medium composition is perturbed. This is implemented in the library in two ways:

1. **Numerical sensitivity**: A certain number of perturbed media realizations are generated within a provided range from the nominal medium. For all of these media the predictions of steady-state concentrations and uptake-secretion rates of the extracellular metabolites are evaluated for a provided model. For these data some descriptive statistics are evaluated, i.e. mean, standard deviation, minimum and maximum;

2. **Local sensitivity**: The local sensitivity correspond to the gradient of the quantities of interest listed above, with respect to the medium composition (relative to the components that are optimized). This type of sensitivity gives the information on how a small variation in each component of the medium would affect each steady-state concentration, rate and harvest rate locally.

The library functions performing these two sensitivity analysis are
`numerical_sensitivity_medium_variability` and `local_sensitivity_medium_variability` respectively.

5.4 Ranked optimization : associate a score to local solutions

The nominal optimization problem setup in Section 5.1 is a non-convex problem, and as such it is possible (almost certain in our context) that it will have several local solutions. To address this issue we usually start the optimization from many initial conditions, an approach which is referred to as multistart. Starting from different initial conditions we will converge to different local solutions of the optimization problem (some of this might, and generally will, coincide). However not all local solutions are equal, and before performing experiments we want to choose one (or more) of these solution which we deem to be the best. A common approach in global optimization is to simply consider the solution achieving the highest (or lowest in a minimization problem) objective function value. While this is certainly of interest for the design of experiment, we will also take into account the distance of any given solution from the dataset of media already tested experimentally and the sensitivity of each solution to perturbations of the nominal medium composition, for the reasons described in the above Sections 5.2 and 5.3.

In particular, for the k-th local solution of the optimization (i.e. the one obtained starting from the k-th initial condition in the set of all initial conditions) we can associate a score which is obtained as

$$z^k = \lambda_\Phi \Phi^k + \lambda_d d^k - \lambda_s s^k$$

where

- Φ^k is the predicted objective function value (which we are trying to maximize) for the k-th local solution of the optimization;
- d^k is the distance of the k-th optimal medium, obtained as the k-th local solution, from a dataset of media already tested experimentally;
- s^k is a score of sensitivity of the k-th local solution with respect to variability of its nominal composition. In this case we will consider the standard deviation of the considered objective function obtained through the numerical sensitivity analysis described in Section 5.3
- λ_Φ, λ_d and λ_s are non-negative weights denoting the importance of each of the above components to the overall score associate to each local solution.

The solutions are then ordered based on their score (in descending order), so that we can have a ranking of such solutions, informing us before the experimental phase. We can refer to this as **ranked optimization** and it is implemented in the library function `model_based_optimization_with_ranking`.

Remark 7. *The idea of the score associated to each local solution is to achieve a trade-off between its predicted performance, exploratory value and robustness to medium variability. The overall score value is dependent on the choice of the weights λs, which determine the importance we give to each aspect of a given solution. Notice that the user can also choose one (or more) of these weights to be* 0 *meaning that the particular aspect is not of interest in determining the best solution to test experimentally.*

Remark 8. *The user interested in seeing how the local solutions score would change by changing the non-negative weights λ_Φ, λ_d and λ_s can do so by simply calling the function `get_optimization_results_ranking` with the new weights as parameter.*

5.5 Optimization involving unmeasured (but modelled) metabolites

In some cases an extended macroreaction stoichiometric matrix could be available, with some metabolites that are unmeasured, but for which a kinetic model of their uptake-secretion rate is available. In this case the user can involve these metabolites in the optimization, by simply using the extended stoichiometric matrix and specifying the metabolites of interest in such matrix through the field `involved_met_indices` in the structure `index_data` which is passed to all optimization functions as an argument. If this field is not instantiated by the user, the optimization will automatically consider all the rows of the macroreaction stoichiometric matrix.

Remark 9. *Although some metabolites are unmeasured, if these are involved in the optimization (either in the objective function or in the constraints) the user should provide their bounds to the optimization procedure, as well as an initial value for initialization purposes.*

6 Troubleshooting

Q.1. My simulations/optimization returns very strange results (e.g. much different than what I expected). What is happening?
It is difficult to answer this because many things could be happening: the model is not perfect, the optimization or non-linear equation solvers fail, etc. However, in my experience, in many cases the answer is simpler and usually related to external files not following the required templates. For example the order of metabolites should be the same in all the model files (containing the stoichiometric matrix and the parameters matrix) and the data files. Most of the times this is not the case, and there are functions in the library that can pre-process these files so that compatibility is restored. These functions, to work properly, will read the metabolite labels in such files. *THESE LABELS SHOULD BE THE SAME IN EACH FILE.* "Glc_ext", "Glc_ex", "Glc" and "Glc ext" are all different labels and the above pre-processing functions will give you an error in the best-case, or produce unreliable results in the worst-case. Check

that your external files follow the correct template for the library (you can find examples of models and data files in the folders `models` and `data` respectively).

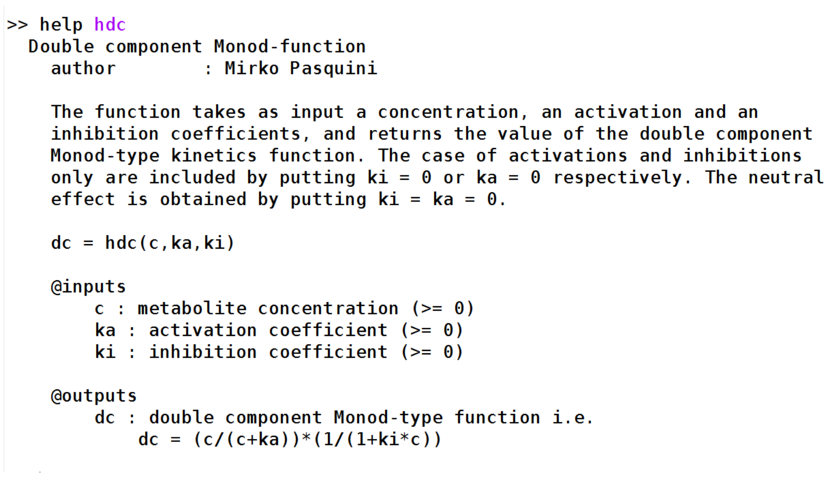
Q.2. I would like to use a function `xyz` but I do not know what are the input or the output parameters, and this document does not specify them. What can I do?
You can access the documentation of each function by using the `help` command in Matlab, followed by the name of the function you want to use. See Figure 1 as an example.

Q.3. I have no idea where to start. What is the best way to start using the library?
I would say that a good 80% of the library functionalities can be explored by simply taking a look at the usecase examples (in the `usecase-examples` library folder). These scripts will guide you through basic features of the library (loading a model, running a virtual experiment, optimizing using nominal optimization, etc.). Feel free to copy-paste these scripts and adjust them as you need.

Q.4. When I run my code I get an error saying "Unrecognized function or variable [...]". What should I do?
Be sure to run the code in the correct folder (e.g. most of the usecase examples should be executed while being in the folder `usecase-examples`) and be sure that all the library folders are visible to MATLAB (this can be done through Set Path in the Home tab of MATLAB).

References

[1] Erika Hagrot, Hildur Æsa Oddsdóttir, Meeri Mäkinen, Anders Forsgren, and Véronique Chotteau. Novel column generation-based optimization approach for poly-pathway kinetic model applied to cho cell culture. *Metabolic engineering communications*, 8:e00083, 2019.



```
>> help hdc
  Double component Monod-function
    author        : Mirko Pasquini

    The function takes as input a concentration, an activation and an
    inhibition coefficients, and returns the value of the double component
    Monod-type kinetics function. The case of activations and inhibitions
    only are included by putting ki = 0 or ka = 0 respectively. The neutral
    effect is obtained by putting ki = ka = 0.

    dc = hdc(c,ka,ki)

    @inputs
        c : metabolite concentration (>= 0)
        ka : activation coefficient (>= 0)
        ki : inhibition coefficient (>= 0)

    @outputs
        dc : double component Monod-type function i.e.
            dc = (c/(c+ka))*(1/(1+ki*c))
```

Figure 1: Documentation for the function `hdc`.