Experimental design for predictive models in microbiology depending on environmental variables

Polina Gaindrik[1,2], Jonas Pleyer[1,2], Daniel Heger[3], Christian Fleck[1,2]

1 University of Freiburg
2 Freiburg Center for Data Analysis and Modeling
3 tsenso GmbH

Abstract

The aim of predictive microbiology is to provide tools and methods for predicting the growth, survival, and death of microorganisms in different food matrices, under a range of environmental conditions. The parametrised mathematical models need to be calibrated using dedicated experimental data. To efficiently plan experiments model-based experimental design is used. In this chapter, we explain model-based experimental and provide step-by-step instructions for finding the optimal design using the well-known Baranyi&Roberts growth model as an example. To make this chapter self-consistence we provide a Python software *eDPM* for Ordinary Differential Equation-based models.
Keywords Optimal experimental design, Parameter estimation, Fisher Information matrix, Identifiability, Uncertainty

Introduction

Mathematical modelling is a widely used tool to describe, understand and predict further behaviour of living systems. In particular, in the field of Predictive Biology, one can find a large variety of works that dwell on building models of different levels of complexity controlled by model parameters, e.g., to describe bacteria growth [6]. Predictive microbiology is a subfield of food microbiology that uses mathematical models to predict the behaviour of microorganisms in food. It is based on the premise that the behaviour of microorganisms can be mathematically described by the use of mathematical models, which can take into account the effects of various factors such as temperature, pH, water activity, and the presence of preservatives, on the growth and survival of microorganisms. These models can then be used to predict the behaviour of microorganisms under different conditions, and to evaluate the effectiveness of various control measures, such as refrigeration, heat treatment, or the addition of preservatives. Predictive microbiology has many applications in food safety, as it can be used to assess the risk of foodborne illness, to design safe food

processing and storage practices, and to develop new food products with extended shelf-life. For successful predictions it is essential that the model parameters can be estimated from experimental data. Due to measurement noise, the parameters can only be estimated with some uncertainties. That gives rise to a set of important questions: given the model structure can all parameters be estimated? What should be measured and when? What are the confidence intervals for the parameters? How can the experimental effort be minimised? Answering these questions leads to finding the optimal experimental design (OED) where optimised experimental conditions and/or measurement times allow for a reduction of the experimental load without loss of information [2, 7]. In general, the Experimental Design procedure can be used not only for the parameter estimation but also for model discrimination [12, 15]. However, in this chapter, we focus on Experimental Design for parameter estimation. Comprehensive reviews can be found here: [1,8,18,23]. Depending on the goal, a researcher faces an important choice, which of the several sometimes contradicting objectives should be chosen for a particular case. For example, is it more desirable to have precise knowledge in a chosen set of parameters and less information about the remaining ones? What is the best balance between experimental effort and precision? Using multi-objective approaches, some attempts were made to answer these questions and to improve the experimental design by combining several objectives [13, 20]. For models which depend on external cues like temperature, it is *a priori* not clear whether constant are variable conditions are more efficient for the parameter estimation [11, 22]. The whole parameter estimation process starts with choosing a model structure. Once this is chosen a first parameter set is selected based on the literature review or prior data from previous experiments. Using this parameter set a first optimal experimental design is determined taking into account specific constraints, such as maximum number of measurements, measurement times, etc. Next, the designed experiments need to be performed and this data result into a new estimation of the parameters. If the confidence intervals of the parameters are sufficiently small, the scheme ends, otherwise one uses the new estimates as the starting point for the next experimental design (see Fig. 1). The process can be repeated several times to increase the precision of the parameter estimates until the desired accuracy is achieved. To test the scheme one can also perform numerical experiments and obtain *in-silico* data.

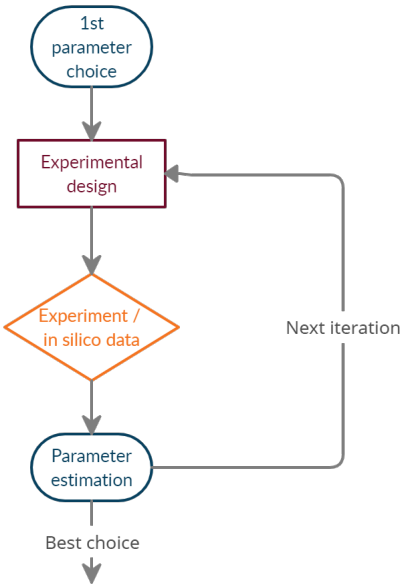


Figure 1. The workflow of the iterative process for model-based experimental design for parameter estimation.

Materials and Methods

There are several software packages available for model-based experimental design []. These packages comprise a large number of tools and due to this require some time to understand how to use them. to make this tutorial self-contained, we developed `eDPM` (Experimental Design for Predictive Microbiology), which is a set of tools to calculate the Sensitivity and Fischer Information Matrix and do parameter space exploration in order to find optimal results. We expect require a working installation of the popular scripting language Python $\geq$ 3.7 [21]. For installation instructions, please refer to the website python.org. We also In addition ... expect users, to be able to write, execute and display output of scripts. This tutorial can also be followed using jupyter notebooks [19]. Users can obtain it by installing `eDPM` from pypi.org. The python website has guides for installing packages packaging.python.org/. Most Unix-Based systems (eg. GNU/Linux) and Mac-OS can use `pip`

```
pip install eDPM
```

or `conda` to install the desired package.

```
conda install eDPM
```

The Documentation of the package is continuously updated. After the installation of the package is complete, we open a file with a text editor of choice and simply

start writing code. We begin by writing a so-called she-bang which is responsible to signalize that this file is to be executed with python. Afterwards, we import the needed packages. This will serve as the starting point for our script. In the following, we will append more code to it and utilise the methods developed in eDPM []. If line numbers are missing, these should be filled by blank lines for better readability.

```
#!/usr/bin/env python3

import numpy as np
from eDPM import *
```

Code Sample 1. Import statements to use eDPM

Model Formulation

Theory

As a starting point it is necessary to define the mathematical model. We restrict our discussion to biological system which can be described by a system of Ordinary Differential Equations:

$$\begin{cases} \dot{\mathbf{x}}(t) = f(t, \mathbf{x}, \mathbf{u}, \mathbf{p}) \\ \mathbf{x}(t_0) = \mathbf{x}_0 \end{cases} . \tag{1}$$

Here $\mathbf{x} = (x_1, x_2, ..., x_n)$ is a vector of state variables of the system with initial condition $\mathbf{x}_0$, t is the time, $\mathbf{u}$ is a vector of a externally controlled inputs into the system (e.g., supply of glucose or the external temperature) and $\mathbf{p}$ are the parameters of the system. We assume that a subset of the parameters $\mathbf{p}$ is unknown and should be estimated from data. Predominantly, the state variables cannot be directly observed, but rather the observables are functions of the state variables:

$$\mathbf{y}(t) = g(t, \mathbf{x}(t), \mathbf{u}, \mathbf{p}) + \epsilon(t), \tag{2}$$

where the function g is the observation or output function, and ϵ is the measurement noise. The observational noise is often assumed to be an independently distributed Gaussian noise with zero mean and variance σ^2: $\epsilon(t) \sim N(0, \sigma^2)$, $\forall t$.

To demonstrate how Experimental Design works, we use in this tutorial the widely employed mathematical model by Baranyi and Roberts model [5], which was devised to describe bacteria growth. The model introduces two state variables $\mathbf{x} = (x_1, x_2)$, where $x_1(t)$ denotes the cell concentration of a bacterial population at the time t and $x_2(t)$ defines a physiological state of the cells, the process of adjustment (lag-phase):

$$\begin{cases} \dot{x}_1(t) = \frac{x_2(t)}{x_2(t)+1}\mu^{\max}\left(1 - \frac{x_1(t)}{x_1^{\max}}\right)x(t) = f_1 \\ \dot{x}_2(t) = \mu^{\max}x_2(t) = f_2 \end{cases} . \tag{3}$$

Here $\mu^{\max}$ determines the maximum growth rate, and $x_1^{\max}$ is the maximal bacteria concentration due to environmental constraints. To account for the influence of the temperature on the activity of the model, we will use the 'square root' or Ratkowsky-type model for the maximum growth rate []

$$\sqrt{\mu^{\max}} = b(T - T_{\min}), \tag{4}$$

where b is the regression coefficient, and $T_{\min}$ is the minimum temperature at which the growth can occur. As the observable we choose the bacteria count, i.e:

$$y(t) = x_1(t) + \epsilon(t), \tag{5}$$

a common choice in predictive microbiology. By this and the equations (3), (4) the system is fully defined. Here $x_1^{\max}, b, T_{\min}$ are the parameters that we estimate using observational data y at measurement times t_i, and temperature T is an input of the system. Based on this model, we would like to optimize the choice of measurement times as well as temperatures (inputs) of the system to find the optimal experimental design.

Code

In order to be able to solve the equations numerically, we first need first to define the Ordinary Differential Equation (ODE) system in python. As was explained in the previous sections, this system consists of the equation (1) with initial values t_0, x_0. Next, we need to define all numerical values present in the system. We distinguish between time points t_i at which the result of the ODE is evaluated, inputs u, which alter the behaviour of our system (for example temperature, humidity, etc.), parameters p, which are the quantities that we want to estimate and other arguments which might be needed to solve the ODE. Table 1 provides an overview of these quantities and gives corresponding names in the code. In the following, we will step-by-step explain how to specify all variables needed by using the example of the Baranyi-Roberts Model (Eq. 3).

Description	Formula	Code
ODE	f	`def ode_fun(t, x, u, p, ode_args):`
	$\partial f/\partial x$	`def ode_dfdx(t, x, u, p, ode_args):`
	$\partial f/\partial p$	`def ode_dfdp(t, x, u, p, ode_args):`
Observable	g	`def obs_fun(t, x, u, p, ode_args)`
(optional)	$\partial g/\partial x$	`def obs_dgdx(t, x, u, p, ode_args)`
	$\partial g/\partial p$	`def obs_dgdp(t, x, u, p, ode_args)`
Initial value	x_0	`x0`
Initial time	t_0	`t0`
Time points	t_i	`t`
Inputs	u	`u`
Parameters	p	`p`
Other arguments		`ode_args`

Table 1. Summary of user-defined functions and variables.

Defining the ODE We must define a few functions in python. The first function will be the right-hand side of equation (1), i.e., we need to implement the Baranyi-Roberts model (3) into a function. The implementation can be seen in Figure 2. We explain the individual steps to create this function:

- `def ode_fun(t, x, u, P, ode_args)`
  The function name can be chosen freely, but the order of required function arguments is fixed by the definition.

- `(x1, x2) = x`
  `(Temp,...) = u`
  `(x_max, b, Temp_min) = p`
  Unpack the input vectors `x,u,p` for easy access of their components

- `mu_max = b**2 * (Temp - Temp_min)**2`
  Calculate the maximum growth rate.

- `return [`
  `mu_max * (x2/(x2 + 1)) * (1 - x1/x_max) * x1`
  `mu_max * x2`
  `]`
  Calculate the right hand side of the ODE, store it in a list (`[ ... ]`) and return it.

```python
def baranyi_roberts_ode(t, x, u, p, ode_args):
    (x1, x2) = x
    (Temp, ) = u
    (x_max, b, Temp_min) = p
    # Define the maximum growth rate
    mu_max = b**2 * (Temp - Temp_min)**2
    return [
        mu_max * (x2/(x2 + 1)) * (1 - x1/x_max) * x1,
        mu_max * x2
    ]
```

Code Sample 2. Definition of the Baranyi-Roberts ODE model.

Parameter Estimation

After defining the model, the user need to provide an initial parameter set. This could be chosen from the literature or estimated from previously gathered experimental data. It is common to assume that the observational or measurement noise is Gaussian white noise (e.g., no temporal correlations) with zero mean and variance $\sigma(t)^2$: $\epsilon(t) \sim N(0, \sigma(t)^2)$, $\forall t$. In this case the logarithm of the likelihood function (log-likelihood) is given by:

$$\ln L(\mathbf{p}) \propto -\sum_i \frac{\left(\mathbf{g}^i(\mathbf{p}) - \mathbf{d}^i\right)^2}{2\sigma_i^2}. \tag{6}$$

We introduced the following short hand notations:

- $\mathbf{d}^i$: data measured at time $t = t_i$
- $\mathbf{g}^i(\mathbf{p})$: the observation function depending on the parameter vector $\mathbf{p}$ evaluated at time $t = t_i$
- σ_i^2: variance of the observational noise at time $t = t_i$.

The method of maximum likelihood consist in searching for the parameter vector $\mathbf{p}$ which maximizes the likelihood or equivalently minimizes the log-likelihood [10].

Experimental Design

After defining the model and setting the initial parameter vector $\mathbf{p}_0$ we can proceed to the Experimental Design. In essence Experimental Design comprises maximizing an objective function depending on the model, the initial parameter vector $\mathbf{p}_0$, external input $\mathbf{u}$ into the system, and a set of discrete observation times t_i at which the potential measurements should be performed. The objective function needs to quantifies the information the observation $\mathbf{y}$ has about the parameter vector $\mathbf{p}$. A common choice here is constructing objective functions based on the Fisher information matrix (FIM) [14]. According to the so-called

'Cramer-Rao inequality', the Fisher information is inversely proportional to the minimal squared estimation error [9]. Due to this relationship maximization of the information leads to a minimization of the variance or uncertainty in the parameters. In the following we explain one of the ways to calculate the Fisher information matrix for an ODEs system (1) and the observable function (2) [14].

Sensitivity Calculation

An important ingredient into the FIM are local sensitivities. Assume that functions f and g are differentiable functions with respect to the state variables $\mathbf{x}$ and parameters $\mathbf{p}$. The local sensitivities are defined by $s^x_{ij} = (\mathrm{d}x_i/\mathrm{d}p_j)$. These can be calculated using an enhanced system of the ODEs:

$$\begin{cases} \dot{x}_i(t) = f_i(t, \mathbf{x}, \mathbf{u}, \mathbf{p}) \\ \dot{s}^x_{ij} = \sum_k \frac{\partial f_i}{\partial x_k} s^x_{kj} + \frac{\partial f_i}{\partial p_j} \end{cases} . \tag{7}$$

For the FIM we need the sensitivities of observables functions $(\mathrm{d}y_i/\mathrm{d}p_j) = s_{ij}$. We determine these at a certain times t_m and input u_n using the solutions of s^x_{ij}:

$$s_{ij}(t_m, u_n) = \sum_k \left.\frac{\partial g_i}{\partial x_k}\right|_{t_m,u_n} s^x_{kj}(t_m, u_n) + \left.\frac{\partial g_i}{\partial p_j}\right|_{t_m,u_n} . \tag{8}$$

In the case the parameters are of very different scale (e.g., on the second and on the day scale), it may be advisable to use the relative or normalised sensitivities to improve not the absolute but the relative accuracy of the parameter estimates.:

$$\tilde{s}_{ij}(t_m, u_n) = \frac{\mathrm{d}\ln(y_i)}{\mathrm{d}\ln(p_j)} = \left.\frac{\mathrm{d}y_i}{\mathrm{d}p_j}\frac{p_j}{y_i}\right|_{t_m,u_n} . \tag{9}$$

These sensitivities are the elements of the sensitivity matrix [16]. For example, in case of two observables $\mathbf{y} = (y_1, y_2)$, two different inputs $\mathbf{u} = (u_1, u_2)$, N different measurement times and N_p parameters, sensitivity matrix reads:

$$S = \begin{pmatrix} s_{11}(t_1,u_1) & \dots & s_{1N_p}(t_1,u_1) \\ \vdots & & \vdots \\ s_{11}(t_N,u_1) & \dots & s_{1N_p}(t_N,u_1) \\ s_{11}(t_1,u_2) & \dots & s_{1N_p}(t_1,u_2) \\ \vdots & & \vdots \\ s_{11}(t_N,u_2) & \dots & s_{1N_p}(t_N,u_2) \\ s_{21}(t_1,u_1) & \dots & s_{2N_p}(t_1,u_1) \\ \vdots & & \vdots \\ s_{21}(t_N,u_1) & \dots & s_{2N_p}(t_N,u_1) \\ s_{21}(t_1,u_2) & \dots & s_{2N_p}(t_1,u_2) \\ \vdots & & \vdots \\ s_{21}(t_N,u_2) & \dots & s_{2N_p}(t_N,u_2) \end{pmatrix} \tag{10}$$

This matrix we will use to directly calculate the FIM via equation:

$$F = S^T Q^{-1} S, \tag{11}$$

where Q is the covariance matrix of measurement error []. If the measurements are independent, then only the diagonal elements of the matrix are non-zero:

$$Q = \begin{pmatrix} \sigma_1^2(t_1, u_1) & 0 & 0 & \dots & 0 \\ 0 & \sigma_1^2(t_2, u_1) & 0 & \dots & 0 \\ 0 & 0 & \sigma_1^2(t_1, u_2) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_2^2(t_N, u_2) \end{pmatrix}. \tag{12}$$

Here $\sigma_i^2(t_m, u_n)$ is the error of the measurements of observable y_i at time point t_m with input u_n. The error can be either estimated from data or approximated by some error model. For example, one may consider that the error contribution consists of an absolute part that stays constant and a relative part that is proportional to observable value. Then the diagonal elements of the covariance matrix take form:

$$\sigma_i^2(t_m, u_n) = \gamma_{\text{abs}} + \gamma_{\text{rel}} \cdot y_i(t_m, u_n), \tag{13}$$

where γ_{abs} and γ_{rel} are the coefficients determining the absolute and the relative error contribution, respectively. In case one uses relative sensitivities $\tilde{s}_{ij}(t_m, u_n)$, one needs to use relative measurements errors for the matrix Q:

$$\tilde{\sigma}_i^2(t_m, u_n) = \frac{\sigma_i^2(t_m, u_n)}{y_i^2(t_m, u_n)}. \tag{14}$$

Note that this approach is heuristic and the matrix F is not the Fisher information matrix [].

Numerical Sensitivity Calculation

For the calculation of the sensitivities (Eq. 7) we need to supply the derivatives of f with respect to the parameters and state variables. We define $\mathbf{x} = (x_1, x_2, x_3)$ and $\mathbf{p} = (p_1, p_2, p_3)$ with $p_1 = x_{max}$, $p_2 = b$ and $p_3 = T_{\min}$. Firstly, we calculate the mathematical derivatives $\partial f_i / \partial p_j$ and $\partial f_i / \partial x_j$ and then implement the corresponding functions. The first component of the Baranyi-Roberts model reads:

$$\dot{x}_1(t) = f_1(t, \mathbf{x}, \mathbf{u}, \mathbf{p}) = \mu^{\max} \frac{x_2(t)}{x_2(t) + 1} \left(1 - \frac{x_1(t)}{x_1^{\max}}\right) x_1(t), \tag{15}$$

where $\sqrt{\mu^{\max}} = b(T - T_{\min})$. It follows:

$$\frac{\partial \mu^{\max}}{\partial p_1} = 0 \tag{16}$$

$$\frac{\partial \mu^{\max}}{\partial p_2} = 2b(T - T_{\min})^2 \qquad = \frac{2\mu^{\max}}{b} \tag{17}$$

$$\frac{\partial \mu^{\max}}{\partial p_3} = -2b^2(T - T_{\min}) = \frac{2\mu^{\max}}{T - T_{\min}} \tag{18}$$

and consequently:

$$\frac{\partial f_1}{\partial p_1}(t) = -\mu^{\max} \quad \frac{x_2(t)}{x_2(t)+1} \quad \frac{x_1(t)}{\left(x_1^{\max}\right)^2}\, x_1(t) \tag{19}$$

$$\frac{\partial f_1}{\partial p_2}(t) = \frac{2\mu^{\max}}{b} \quad \frac{x_2(t)}{x_2(t)+1}\left(1 - \frac{x_1(t)}{x_1^{\max}}\right) x_1(t) \tag{20}$$

$$\frac{\partial f_1}{\partial p_3}(t) = \frac{2\mu^{\max}}{T - T_{\min}} \frac{x_2(t)}{x_2(t)+1}\left(1 - \frac{x_1(t)}{x_1^{\max}}\right) x_1(t). \tag{21}$$

Similarly, the derivatives $\partial f / \partial x_i$ are given by:

$$\frac{\partial f_1}{\partial x_1}(t) = \mu^{\max} \quad \frac{x_2(t)}{x_2(t)+1} \quad \left(-\frac{1}{x_1^{\max}}\right) x_1(t) \tag{22}$$

$$\frac{\partial f_1}{\partial x_2}(t) = \mu^{\max}\left(\frac{1}{x_2(t)+1} - \frac{x_2(t)}{(x_2(t)+1)^2}\right)\left(1 - \frac{x_1(t)}{x_1^{\max}}\right) x_1(t). \tag{23}$$

The readers are encouraged to calculate the components $\partial f_2 / \partial p_i$ and $\partial f_2 / \partial x_i$ as an exercise. The resulting implemented functions in python can be seen in Figure 3.

```
20 def ode_dfdp(t, x, u, p, ode_args):
21     (x1, x2) = x
22     (Temp, ) = u
23     (x_max, b, Temp_min) = p
24     mu_max = b**2 * (Temp - Temp_min)**2
25     return [
26         [
27             mu_max * (x2/(x2 + 1)) * (x1/x_max)**2,
28             2 * b * (Temp - Temp_min)**2 * (x2/(x2 + 1))
29                 * (1 - x1/x_max)*x1,
30             -2 * b**2 * (Temp - Temp_min) * (x2/(x2 + 1))
31                 * (1 - x1/x_max)*x1
32         ],
33         [
34             0,
35             2 * b * (Temp - Temp_min)**2 * x2,
36             -2 * b**2 * (Temp - Temp_min) * x2
37         ]
38     ]
39
40 def ode_dfdx(t, x, u, p, ode_args):
41     (x1, x2) = x
42     (Temp, ) = u
43     (x_max, b, Temp_min) = p
44     mu_max = b**2 * (Temp - Temp_min)**2
45     return [
46         [
47             mu_max * (x2/(x2 + 1)) * (1 - 2*x1/x_max),
48             mu_max * 1/(x2 + 1)**2 * (1 - x1/x_max)*x1
49         ],
50         [
51             0,
52             mu_max
53         ]
54     ]
```

Code Sample 3. Derivatives of the function f of the Baranyi-Roberts model ODE.

Define numerical values

After the definition of the structure of the ODE, we need to specify numerical values. It is good practice, to gather such definitions in the `__main__` method of the python program as it was done in Figure 4. We start by defining the parameters of the ODE (line 59) and afterwards the initial values (line 62). Next, we constrain the optimisation routine to find the best possible time points in

the interval $t_i \in [t_{\text{low}}, t_{\text{high}}]$. To do this, we write the times as a dictionary with entries `{"lb":t_low, "ub":t_high, "n":n_times}` where `n_times` is the number of discrete time points at which we want to sample the system (line 65). In contrast, if we wanted to specify explicit values for the sampling points, we would have needed to supply a list of time values or a `np.ndarray` directly. One can see this approach in line 70 of the code example. Here, we supply a `np.ndarray` with explicit values, thus fixing the sampling points them for the optimisation routine.

```
57 if __name__ == "__main__":
58     # Define parameters
59     p = (np.exp(21.1), 0.038, 2)
60 
61     # Define initial conditions
62     x0 = np.array([np.exp(2.36), 1 / (np.exp(2.66)-1)])
63 
64     # Define interval and number of sampling points for times
65     times = {"lb":0.0, "ub":1500.0, "n":4}
66 
67     # Define explicit temperature points
68     Temp_low = 4.0
69     Temp_high = 8.0
70     n_Temp = 3
71 
72     # Summarize all input definitions in list (only temperatures)
73     inputs = [
74         np.linspace(Temp_low, Temp_high, n_Temp)
75     ]
```

Code Sample 4. The main function contains the actual values for our model definition.

Defining Explicit Values and Sampling

The difference between choosing explicit values and specifying a sampling range may be subtle at first glance, but in turn allows to very easily switch between fixed values and optimised sampling points. For example, suppose we want to optimise the temperature at which the experiments are performed, which means we let the optimisation algorithm pick the optimal temperature points such that the information gathered from the system is maximised. Then the difference between defining explicit values for the temperatures and defining an interval for optimisation can be understood in code sample 5.
The different variables can be treated individually as needed. Suppose, we have a system with temperatures and humidity as input variables. We could decide to have the temperature optimised, but fix the humidity explicitly, because

```python
# This will choose values explicitly
inputs = [
    np.linspace(Temp_low, Temp_high, n_Temp)
]
# >>> inputs
# [array([4., 6., 8.])]


# This will sample in the interval
inputs = [
    {"lb":Temp_low, "ub":Temp_high, "n":n_Temp}
]
# >>> inputs
# [{'lb': 4.0, 'ub': 8.0, 'n': 3}]
```

Code Sample 5. Difference between choosing explicit values and sampling over a given interval.

experimentally one can change the temperature continuously (or in discrete steps) but is restricted regarding the humidity. In our code, we simply would mix explicit and sampled definitions for individual inputs (see code sample 6). [MIR IST DIES NICHT GANZ KLAR, WIR MÜSSEN DARÜBER REDEN].

```python
inputs = [
    # These values will be sampled
    {"lb":Temp_low, "ub":Temp_high, "n":n_Temp},
    # These are fixed
    np.array([0.4, 0.45, 0.5, 0.55])
]
```

Code Sample 6. Mixing of explicit and sampling for inputs.

Define Model

After we have decided on the numerical values for our model, we need to put everything together. The `FisherModel` class serves as the entry point. Here, we simply supply every previously made definition of variables and methods. When using the syntax as shown in code sample 7, the order of arguments does not matter. However, when only using `FisherModel(x0, 0.0, ...)`, please pay attention to the order of arguments.

```
    # Create the FisherModel which serves as the entry point
    #  for the solving and optimization algorithms
    fsm = FisherModel(
        ode_x0=x0,
        ode_t0=0.0,
        ode_fun=baranyi_roberts_ode,
        ode_dfdx=ode_dfdx,
        ode_dfdp=ode_dfdp,
        ode_initial=x0,
        times=times,
        inputs=inputs,
        parameters=p
    )
```

Code Sample 7. Define the full model.

There are some optional arguments which can be set if desired ... [IST MIR NICHT GANZ KLAR. MÜSSEN DARÜBER REDEN]. For a full list of optional arguments, we refer to the documentation of the package.

Identifiability

Before proceeding with the optimisation, the reader need to check if the parameters of the system are identifiable, i.e, to examine if it is possible to obtain a unique solution for the parameters from the optimisation. It can happen that a subset of the parameters are non-identifiable. The non-identifiability can be due to the model structure or observables (structural non-identifiability) or insufficient data (practical non-identifiability) []. Structural non-identifiability should be avoided as it results in at least one parameter which could be freely chosen meaning there is no optimal value for this parameter. Fortunately, there is a quick and easy way to check whether the system is structural non-identifiable by calculating the rank of the sensitivity matrix []. For an identifiable system, the rank should coincide with the number of estimated parameters. Only if this condition is satisfied, we can continue with optimisation. In case the rank of the sensitivity matrix is less than the number of parameters it is necessary to change the structure of the model or change or increase the number of observables. Practical non-identifiability results in large confidence intervals []

[STELLEN WIR DENN KEINE ROUTINE DAZU BEREIT?]

Optimality Criteria

The Fisher Information matrix needs to be mapped on an objective function. There are several ways to do this resulting in different objectives also called criteria. Optimisation of the different objectives result in different experimental design, some of the most popular criteria are []:

- **D-optimality criterion** maximises the determinant $\det(F)$ of the FIM. Translated to the parameter space it means that the volume of the confidence region (see Fig. 2) (or the geometric mean of all errors) is minimised. The confidence region at a confidence level of α [WHAT IS α?] can be interpreted as there is a $\alpha \cdot 100\%$ chance that the best parameter fit will belong to this area in case of repeating the experiment and estimations multiple times. It is usually presented as a confidence ellipsoid. D-optimality is the most widely used criterion and is suitable even if the parameters have different dimensionalities [WHAT DOES THIS MEAN EXACTLY? THE OTHER CRITERIA ARE NOT SUITED IN THIS CASE? AND WHAT DOES DIMENSIONALITY MEAN?].

- **E-optimality criterion** maximises the smallest eigenvalue $\lambda_{\min}$ of the FIM, which is the same as reducing only the largest estimation error.

- **A-optimality criterion** maximises the sum of all eigenvalues $\sum_i \lambda_i$, which can be interpreted as minimising the algebraic mean of all errors.

- **Modified E-optimality criterion** maximises the ratio between the minimal and maximal eigenvalue $\lambda_{\min}/\lambda_{\max}$ [INTERPRETATION?].

Each of the criteria has its pros and cons so the reader should have a closer look at the various properties of these criteria, for instance, in Franceschini's and Macchietto's paper [8]. For the geometrical interpretation of the criteria using the confidence region shown in Fig. 2.

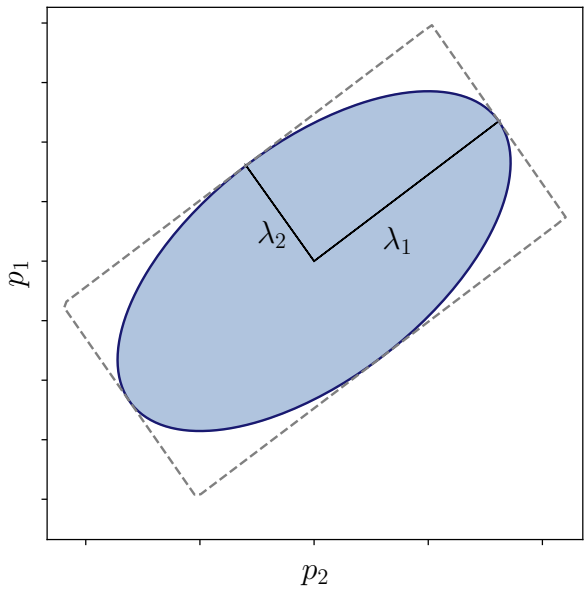


Figure 2. The confidence ellipsoid projection to the (p_1, p_2) parameter space. The ellipsoid shows the geometrical meaning of the optimality criteria. The center of the ellipsoid represents the estimated parameter values. The radii are the uncertainties of the estimates associated with different eigenvalues of the FIM λ_1, λ_2 ($\lambda_1 < \lambda_2$). The D-optimality aims to minimize the volume of the ellipsoid, the E-optimality minimizes the largest radius, A-optimality minimizes the perimeter of the rectangle that encloses the ellipse (dashed gray line), and, finally, Modified E-optimality tends to make the ellipse as spherical as possible.

Optimization

After defining the objective function based on the Fisher Information matrix the next step is to optimise the chosen optimality criteria. Finding the Experimental Design corresponds to finding the global maximum of the objective, which is a special case of optimal control problem []. This problem has been widely studied, and multiple numerical solution algorithms for both local and global optimisation were introduced []. In the supplied toolbox *eDPM* [In the suggested toolbox DO YOU MEAN THE SOFTWARE YOU SUPPLY?], three methods were implemented: differential evolution, basin-hopping, and brute force [WHAT DO YOU MEAN BY THIS?]. Quite good results [WHAT ARE QUIZE GOOD RESULTS? FAST CONVERGENCE?] can be achieved by the Differential evolution (DE) algorithm developed by Storn and Price (1996) [17]. It is one of the stochastic global optimisation methods appropriate for nonlinear dynamic problems. These algorithms have mild computational load but one cannot be sure that the absolute optimum is reached []. We give in the following a brief summary of the implemented methods. For more information the reader should turn to the available literature, e.g., [].

For differential evolution an initial population of candidate vectors for the Experimental Design (sampling times and inputs) is randomly chosen from the region of available values. Then each vector mutates by mixing with other candidate vector. To a chosen vector from the initial population D_0, we add a weighted difference between two other randomly chosen vectors from the same set ($D_{\text{rand1}} - D_{\text{rand2}}$). This process is called mutation and a new vector D_m is obtained. The next step is to construct a new trial solution. This is done by choosing the elements of the trial vector either from the initial D_0 or the mutated D_m vectors. For each new element of trial vector, a random number drawn uniformly from the interval [0, 1) and compared to the so-called recombination constant. If this random number is less than the recombination constant, then the trial vector element is chosen from the vector D_m, otherwise from D_0. The degree of mutation can be controlled by changing the recombination constant; the larger this constant is, the more often vector elements are chosen from D_m. Subsequently, the objective function is calculated using the trial vector and the result is compared to result obtained using the initial solution D_0; and the best of them is chosen for the next generation. This procedure is repeated for every solution candidate of the initial population, by means of which the new generation is build. The process of population mutation is repeated till the desired accuracy is achieved [UNKLAR; WAS IST DENN MIT "ACCURACY" GEMEINT?]. This method is rather simple and straightforward, and does not require any gradient calculation and is easy to parallelise [].

The second implemented method is basin-hopping developed by David Wales and Jonathan Doye [25]. It combines the Monte-Carlo and local optimisation and works as follows. [DIE FOLGENDEN SÄTZE SIND LEIDER VÖLLIG UNVERSTÄNDLICH; BITTE NEU FORMULIEREN] The classic Monte-Carlo algorithm implies that the values of the optimised vector are perturbed and are either accepted or rejected. However, in this modified strategy, after perturbation, the vector is additionally subjected to local minimisation. And only after this procedure the move is accepted according to the Metropolis criterion.

And lastly, a simple brute force method was implemented as well. It is a grid search algorithm calculating the objective function value at each point of a multidimensional grid in a chosen region. The advantage of this algorithm is that we can be sure that the global minimum is achieved [DAS STIMMT NICHT; DAS GITTER-OPTIMUM WIRD GEFUNDEN] because all possibilities are checked. However, the downside of this technique is that rather slow and inefficient. Even though discretisation and reduction of the whole number of possible values [GRID SEARCH ALREADY IMPLIES DISCRETISATION; WAS IHR MEINT IST WOHL EIN GRÖBERES GITTER?] can improve the performance, the computational time of this method allows optimisation only for a small number of optimisation times or inputs. Some other methods for the optimal control problem the reader can find, for example, in the papers of Banga *et al.* [3, 4].

Optimisation Code

The usage of three implemented optimisation methods is greatly simplified in our approach:

`fsr = find_optimal(fsm)`.

[OHNE ARGUMENTE, WIE WERDEN DENN DIE OPTIMIERUNGSROUTINEN AUFGERUFEN? ICH NEHME AN MIT DEFAULT EINSTELLUNGEN?] However, we retain their full functionality. Any optimisation argument of the aforementioned routines, can be specified:

```
fsr = find_optimal(
    # Required argument: The model to optimise
    fsm,
    # Our custom options
    criterion=fisher_determinant,
    relative_sensitivities=True,
    # Options from scipy.optimise.differential_evolution
    recombination=0.7,
    mutation=(0.1, 0.8),
    workers=-1,
    popsize=10,
    polish=False,
)
```

A full list of optional arguments can be seen in the scipy documentation [24]. In addition, there are some interesting optimisation options such as the optional arguments `relative_sensitivities,criterion` , which are responsible for using relative sensitivities $\frac{dy}{dp}\frac{p}{y}$ and specifying the optimality criterion as explained in the previous section [WAS SIND DIE DEFAULT PARAMETER?. Please view the full documentation for explanation. The resulting class `fsr` contains all definitions, current values and information on the optimisation process.

Plotting, Json, etc.

Our package also provides the option to save results into Json file and automatically plot results for the ode solutions, observables and sensitivities:

```
plot_all_observables(fsr)
plot_all_sensitivities(fsr)
json_dump(fsr, "baranyi.json").
```

Results

As a summary of this tutorial, we would like to present the resulting output of our package. To this end we study the growth of a bacterial colony consisting of

a single specie and describe it mathematically using the Baranyi and Roberts model given by Eqs. (3),(4). We would like to estimate the parameters of the model with as few experiments as possible to minimise the experimental effort. The observable is the bacterial count, i.e., the first component of the state variable vector $y = x_1$. As an additional constraint we need to consider that the available climate chambers can only operate the temperature in the range from 3 to 12 degrees. For the starting values of the parameters we take from literature $x_1^{\max} = 10^8$, $b = 0.2$, $T_{\min} = 1.0$, and the initial conditions of the ODEs are chosen to be $x_1(0) = 10^3$, $x_2(0) = 0.01$. Moreover, we choose for all the measurements points the error model given by Eq. (13) with $\gamma_{\text{abs}} = 0.3$ and $\gamma_{\text{rel}} = 0.1$. The task is to propose at which temperatures to perform the experiments and which sampling times to choose. The number of different temperatures of the experiments is not specified yet, but we aim to chose the lowest possible number. To solve this problem, we performed the Experimental Design using the D-optimality criterion, relative sensitivities, and the differential evolution optimisation method. To ensure the identifiability of the system, the rank of the sensitivity matrix should be equal to the number of parameters [WIE SOLL DER LESER DAS TESTEN?]. This condition is satisfied if at least two temperatures are measured and two time points for each temperature are measured, e.g., in total four experiments need to be performed [IST DAS WIRKLICH RICHTIG? DAS SCHEINT MIR EINE SEHR GERINGE ANZAHL AN EXPERIMENTEN ZU SEIN. BITTE MAL MIT DEN EXPERIMENTEN IN DER LITERATUR VERGLEICHEN, Z.B. HIER: 1. Gospavic, R., Kreyenschmidt, J., Bruckner, S., Popov, V. + Haque, N. Mathematical modelling for predicting the growth of Pseudomonas spp. in poultry under variable temperature conditions. International Journal of Food Microbiology 127, 290–297 (2008).] The resulting OED for such a system is presented in Fig. 3, 4.

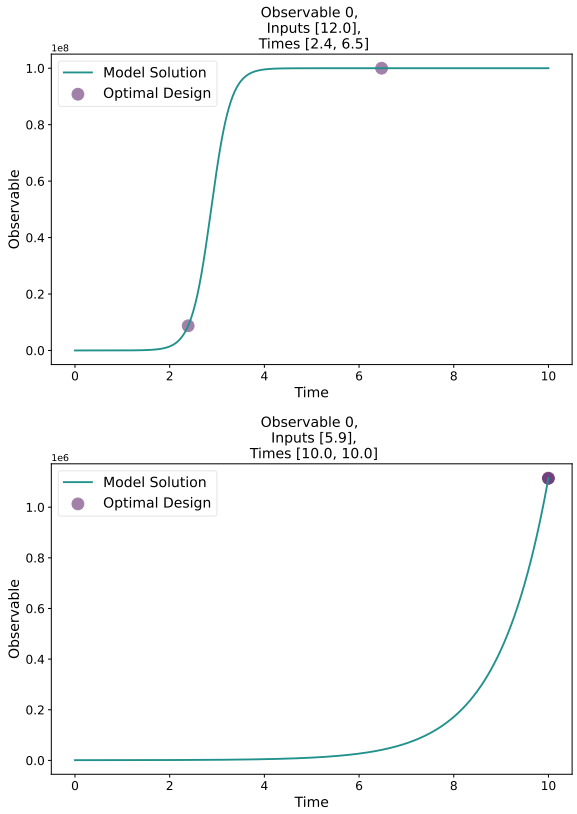
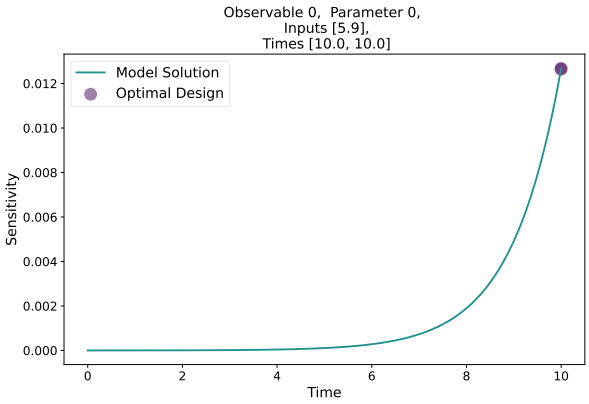
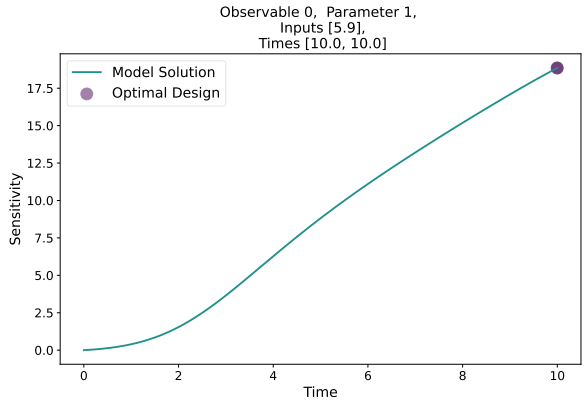
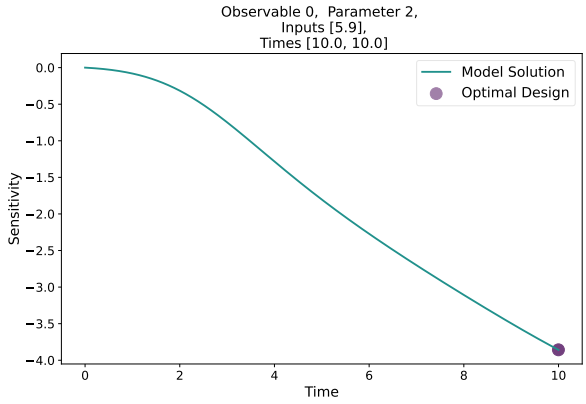


Figure 3. The example of the output of the Experimental Design optimisation procedure for the Baranyi and Roberts model. The line plot presents the model solution for the observable, and the scatter plot determines the time points chosen by Experimental Design. [DIE BILDER BITTE BESCHRIFTEN MIT A UND B, UND MEHR TEXT ALS ERKLÄRUNG. AUCH GIBT ES HIER KEINEN SCATTER PLOT, SONDERN FILLED CIRCLES ODER ÄHNLICH]

As can be noticed, the algorithm has chosen two different timepoints for temperature 12.0 and one time point for temperature value 5.9 Each of these time points corresponds to an extreme of at least one sensitivity (see Fig. 4). [WENN DAS STIMMT DANN SIND DOCH VIEL WENIGER EXPERIMENTE AUSREICHEND UM DIE PARAMETER ZU SCHÄTZEN ALS GEWÖHNLICH DURCHGEFÜHRT WERDEN. BITTE UNBEDINGT MIT DER LITERATUR VERGLEICHEN]


Observable 0, Parameter 0,
Inputs [5.9],
Times [10.0, 10.0]
Model Solution
Optimal Design
Sensitivity
Time


Observable 0, Parameter 1,
Inputs [5.9],
Times [10.0, 10.0]
Model Solution
Optimal Design
Sensitivity
Time


Observable 0, Parameter 2,
Inputs [5.9],
Times [10.0, 10.0]
Model Solution
Optimal Design
Sensitivity
Time

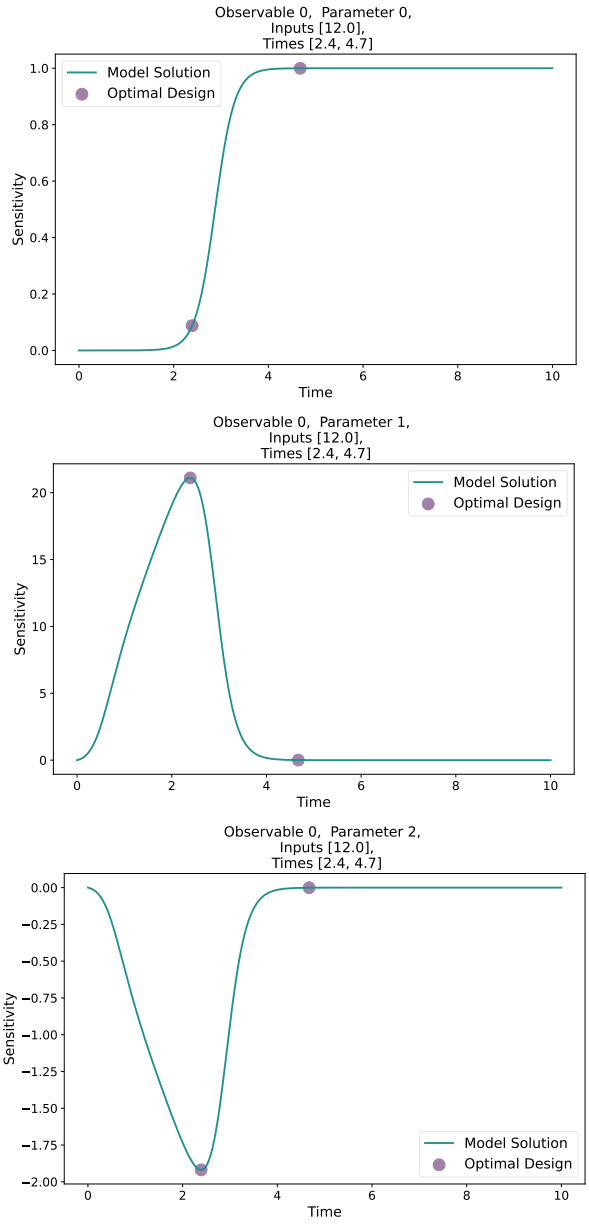


Figure 4. The example of the sensitivities calculated for the Experimental Design optimisation procedure for the Baranyi and Roberts model. The line plots present the model solution for the sensitivities, and the scatter plots determine the time points chosen by Experimental Design. [DIE BILDER IN PANELS ZUSAMMENFASSEN UND BESCHRIFTEN]

Based on the results of the Experimental Design, the optimal number of 419

measurement times is three, which may appear a very low number.fl Indeed, consider how the determinant increases with the number of measurement times in the logarithmic scale (see Fig. 5). It can be noticed that the slope of the curve is largest between point 1 and 2, in fact approximately fourteen orders of magnitude larger than the slope between the other points. For a higher number of times, informational profit stays at the same order of magnitude [WAS BEDEUTET DAS?]. As a result, the researcher can significantly reduce the experimental workload without a noticeable loss of information, which exemplifies the value of OED.

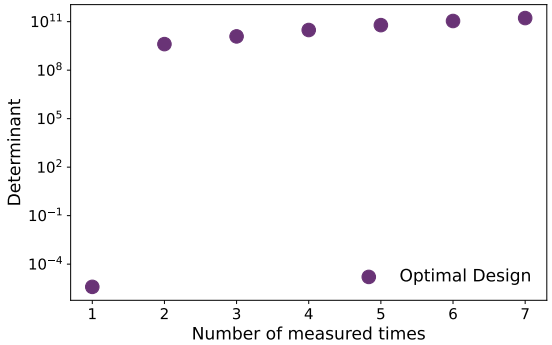


Figure 5. The determinant of the Fisher information matrix as a function of the number of measurement times for the Experimental Design consisted of two measured temperatures.

Two-species resource competition

As a second example we discuss a slightly more complicated system and extend the previous example of bacteria growth to a system where two species are present. The species interact by competitive inhibition because they depend on a common nutrient resource. Analogously to the previous example, we denote the concentration of the two different species as x_1 and y_1, resp. The system can be described by the following set of ODEs:

$$\begin{cases} \dot{x}_1 = \alpha_x R x_1 \\ \dot{y}_1 = \alpha_y R y_1 \\ \dot{R} = -\frac{R}{n_{\max}}(\alpha_x x_1 + \alpha_y y_1) \end{cases} \tag{24}$$

where α_x, α_y are the time and temperature dependent growth rates for population x_1, y_1, and R represents the nutrient pool concentration, respectively. Using the conservation quantity $x_1 + y_1 + n_{\max} R = n_{\max}$ this system can be reduced to:

$$\begin{cases} \dot{x}_1 = \alpha_x x_1 \left(1 - \frac{x_1+y_1}{n_{\max}}\right) \\ \dot{y}_1 = \alpha_y y_1 \left(1 - \frac{x_1+y_1}{n_{\max}}\right). \end{cases} \tag{25}$$

For the growth rates we use [**?**, 5]

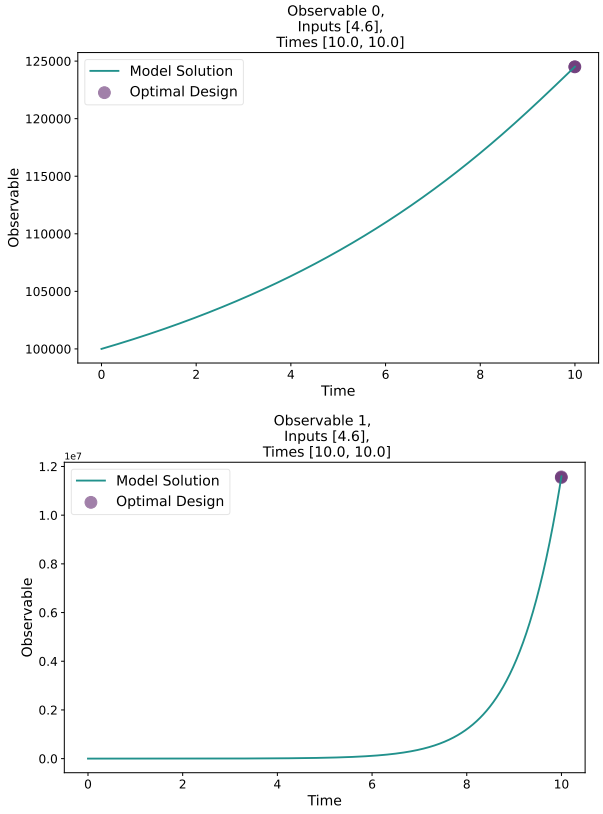
$$\alpha_x(t, T) = b_x^2 (T - T_{\min,x})^2 \frac{x_2(t)}{x_2(t) + 1} \tag{26}$$

Combining equations (25, 26), we get the system of four differential equations:

$$\begin{cases} \dot{x}_1 = b_x^2 (T - T_{\min,x})^2 \frac{x_2}{x_2+1} x_1 (1 - \frac{x_1+y_1}{n_{\max}}) \\ \dot{x}_2 = b_x^2 (T - T_{\min,x})^2 x_2 \\ \dot{y}_1 = b_y^2 (T - T_{\min,y})^2 \frac{y_2}{y_2+1} y_1 (1 - \frac{x_1+y_1}{n_{\max}}) \\ \dot{y}_2 = b_y^2 (T - T_{\min,y})^2 y_2 \end{cases} \tag{27}$$

where x_2, y_2 are the physiological states [WAS MEINT IHR DAMIT?] of the species x_1 and y_1, respectively. The parameter vector of this system reads: $\mathbf{p} = (n_{\max}, b_x, T_{\min,x}, b_y, T_{\min,y})$. We use as initial values for two species $(x_{10}, x_{20}, y_{10}, y_{20}) = (10^5, 0.1, 10^3, 0.1)$ and the starting vector $\mathbf{p}_0 = (10^8, 0.1, 1.0, 0.3, 1.0)$ to calculate the FIM. [WAS SIND GENAU DIE ZU OPTIMIERENDEN GRÖSSEN? BITTE HIER AUCH BESCHREIBEN.]

Other optimisation arguments were taken the same as in previous example. to get the new Optimal Experimental Design for shown in Fig. 6.



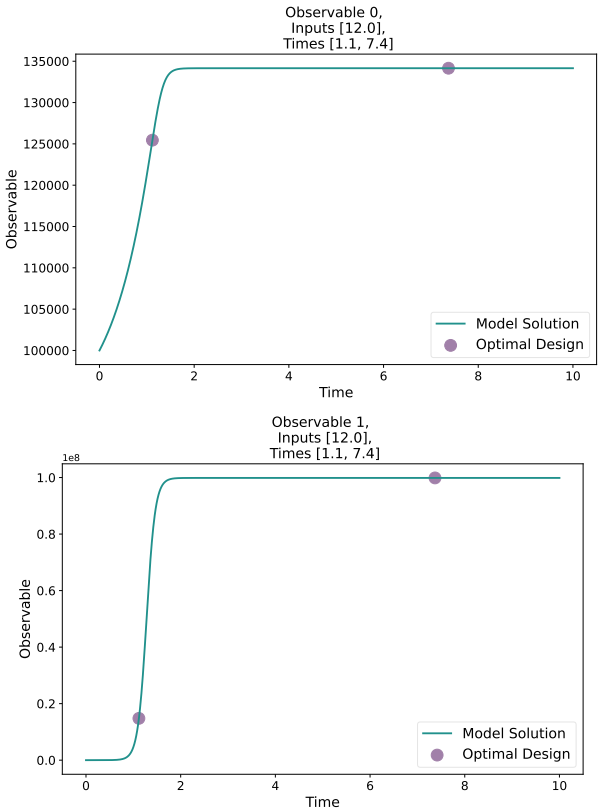


Figure 6. The Optimal Experimental Design for the Baranyi and Roberts model with two different species. The line plot presents the model solution for the observable, and the scatter plot determines the time points chosen by Experimental Design. [SEHT MEINE KOMMENTARE BEI DEN ANDEREN BILDERN.]

The presented Design shows that the least needed number of experiments is two with temperatures of 4.6 and 12.0. Here in each experiment two observables were measured that correspond to the concentrations of two bacteria types. For the first temperature one time-point is needed while for the second one at least two measurement times were chosen. With these experimental conditions the system is identifiable so the data is sufficient to estimate the parameters.

[WAS FEHLT IST DEN EFFEKT DES EXPERIMENTELLEN DESIGNS ZU ZEIGEN. DAZU MÜSST IHR FOLGENDES MACHEN: NEHMT EINEN PARAMETERVEKTOR ALS WAHR AN. STARTET MIT EINEM VEKTOR NICHT ZU WEIT WEG VON DIESEM. PRODUZIERT NUN IN SILICO DATEN UND SCHÄTZT DIE PARAMETER. ZEIGT DIE UNSICHERHEITEN AUF DEN PARAMETERN, AM BESTEN BENUTZT IHR DAZU DIE PROFIL-LIKELIHOOD.]

Supporting information

462

References

[1] A. C. Atkinson. Developments in the Design of Experiments, Correspondent Paper. *International Statistical Review / Revue Internationale de Statistique*, 50(2):161–177, 1982.

[2] BALSA-CANTO, E. BANGA, J.R. COMPUTING OPTIMAL DYNAMIC EXPERIMENTS FOR MODEL CALIBRATION IN PREDICTIVE MICROBIOLOGY. *Journal of Food Process Engineering*, 31, 2008.

[3] J. R. Banga, E. Balsa-Canto, C. G. Moles, and A. A. Alonso. Improving food processing using modern optimization methods. *Trends in Food Science & Technology*, 14(4):131–144, 2003.

[4] J. R. Banga, E. Balsa-Canto, C. G. Moles, and A. A. Alonso. Dynamic optimization of bioprocesses: Efficient and robust numerical strategies. *Journal of Biotechnology*, 117(4):407–419, 2005.

[5] J. Baranyi and T. A. Roberts. A dynamic approach to predicting bacterial growth in food. *International Journal of Food Microbiology*, 23(3-4):277–294, Nov. 1994.

[6] K. BERNAERTS, E. DENS, K. VEREECKEN, A. H. GEERAERD, A. R. STANDAERT, F. DEVLIEGHERE, J. DEBEVERE, and J. F. VAN IMPE. Concepts and Tools for Predictive Modeling of Microbial Dynamics. *Journal of Food Protection*, 67(9):2041–2052, Sept. 2004.

[7] E. V. Derlinden, L. Mertens, and J. F. V. Impe. The impact of experiment design on the parameter estimation of cardinal parameter models in predictive microbiology. *Food Control*, 29(2):300–308, 2013.

[8] G. Franceschini and S. Macchietto. Model-based design of experiments for parameter precision: State of the art. *Chemical Engineering Science*, 63(19):4846–4872, 2008.

[9] R. Frieden and R. A. Gatenby. *Exploratory Data Analysis Using Fisher Information*. Springer Science & Business Media, London, May 2010.

[10] A. Gábor and J. R. Banga. Robust and efficient parameter estimation in dynamic models of biological systems. *BMC Systems Biology*, 9(1):74, Dec. 2015.

[11] M. R. García, C. Vilas, J. R. Herrera, M. Bernárdez, E. Balsa-Canto, and A. A. Alonso. Quality and shelf-life prediction for retail fresh hake (Merluccius merluccius). *International Journal of Food Microbiology*, 208:65–74, 2015.

[12] C. Kreutz and J. Timmer. Systems biology: Experimental design. *The FEBS journal*, 276(4):923–942, Feb. 2009.

[13] F. Logist, B. Houska, M. Diehl, and J. F. V. Impe. Robust multi-objective optimal control of uncertain (bio)chemical processes. *Chemical Engineering Science*, 66(20):4670–4682, 2011.

[14] A. Ly, M. Marsman, J. Verhagen, R. G. J. of Mathematical, and 2017. A tutorial on Fisher information. *Elsevier*, 80:40–55, Oct. 2017.

[15] I. Stamati, S. Akkermans, F. Logist, E. Noriega, and J. V. Impe. Optimal experimental design for discriminating between microbial growth models as function of suboptimal temperature: From in silico to in vivo. *Food Research International*, 89:689–700, 2016.

[16] J. D. Stigter, D. Joubert, and J. Molenaar. Observability of Complex Systems: Finding the Gap. *Scientific Reports*, pages 1–9, Nov. 2017.

[17] R. Storn and K. Price. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 4(11):341–359, 1997.

[18] J. Sun, J. M. Garibaldi, and C. Hodgman. Parameter Estimation Using Meta-Heuristics in Systems Biology: A Comprehensive Review. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, Mar. 2011.

[19] J. Team. Jupyter Notebook. https://jupyter.org.

[20] D. Telen, F. Logist, E. V. Derlinden, I. Tack, and J. V. Impe. Optimal experiment design for dynamic bioprocesses: A multi-objective approach. *Chemical Engineering Science*, 78:82–97, 2012.

[21] G. van Rossum and F. L. Drake. *The Python Language Reference*. Number Pt. 2 in Python Documentation Manual / Guido van Rossum; Fred L. Drake [Ed.]. Python Software Foundation, Hampton, NH, release 3.0.1 [repr.] edition, 2010.

[22] K. J. Versyck, K. Bernaerts, A. H. Geeraerd, and J. F. Van Impe. Introducing optimal experimental design in predictive modeling: A motivating example. *International Journal of Food Microbiology*, 51(1):39–51, Oct. 1999.

[23] C. Vilas, A. Arias-Méndez, M. R. García, A. A. Alonso, and E. Balsa-Canto. Toward predictive food process models: A protocol for parameter estimation. *Critical Reviews in Food Science and Nutrition*, 27:1–14, May 2016.

[24] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore,

J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, and P. van Mulbregt. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, Mar. 2020.

[25] D. J. Wales and J. P. K. Doye. Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, July 1997.