Methods and tools for the optimal design of experiments applied to the efficient characterization of enzyme-mediated escape to antimicrobial treatments

Arthur Carcano

Université de Paris

Frontières de l'Innovation en Recherche et Education (E.D. n° 474)

Méthodes expérimentales et computationelles pour la modélisation des processus cellulaires (InBio)

Methods and tools for the optimal design of experiments applied to the efficient characterization of enzyme-mediated escape to antimicrobial treatments

Par Arthur CARCANO

Thèse de doctorat de **Mathématiques et Informatique**

Dirigée par Grégory BATT
Co-supervisée par Jakob RUESS

Présentée et soutenue publiquement à Paris
Le 13 décembre 2021
Devant un jury composé de :

Gregory BATT, DR, Inria Paris & Institut Pasteur	Directeur
Jakob RUESS, CR, Inria Paris & Institut Pasteur	Membre invité
Eva BALSA-CANTO, Research fellow, CSIC	Rapportrice
Béatrice LAROCHE, DR, Inrae Jouy-en-Josas	Rapportrice
Eugenio CINQUEMANI, CR, Inria Grenoble	Examinateur
Olivier TENAILLON, DR, Inserm, UMR 1137	Examinateur

Résumé en français

Au cours des deux dernières décennies, la biologie des systèmes a connu un essor sans précédent, contribuant à asseoir le rôle prépondérant de la modélisation mathématique en biologie. Des modèles d'une complexité croissante sont ainsi publiés quotidiennement, mais leur utilité en tant qu'outils de prédiction reste limitée. L'utilisation de méthodes mathématiques et informatiques pour concevoir un plan expérimental en amont peut cependant aider à maximiser le rendement d'information espéré. Les fondements théoriques de cette conception assistée de plans expérimentaux remontent au vingtième siècle, mais leur application à des cas d'utilisation concrets demeure un défi. Au cours de mon doctorat, je me suis concentré sur l'élaboration d'un plan optimal d'expériences pour un modèle de résistance bactérienne aux traitements antibiotiques. Pour ce faire, j'ai étudié à la fois comment construire ce plan expérimental, mais aussi comment évaluer in silico sa qualité afin de valider notre procédure de conception. Dans cette thèse, je présente ainsi mon étude des méthodes de conception de plans expérimentaux optimaux en dépit de non-identifiabilités. Cette étude s'appuie sur un changement de paradigme, passant d'une tentative d'identification des valeurs des paramètres, à une tentative d'identification des modèles uniquement pour leur pouvoir de prédiction. Pour démontrer la praticité de mon approche, je propose un pipeline pour valider la qualité des plans expérimentaux, ainsi que plusieurs développements logiciels qui furent nécessaire à son développement. Par ailleurs, concernant les stratégies de conception itérative, j'ai étudié l'effet de l'information a priori sur la supériorité de cette approche de conception expérimentale optimale par rapport aux conceptions d'experts.

Ce manuscrit dresse d'abord un bref aperçu de l'histoire du design optimal, dont les prétendants au titre d'inventrice ou d'inventeurs sont: Kirstine Smith (1878 – 1939), Joseph Diaz Gergonne (1771 – 1859) ou plus tard Ronald Fisher (1890 – 1962) ou Gustav Elfving (1908 – 1984). Il illustre également le développement intimement lié de la biologie des systèmes, de la modélisation mathématique en biologie et du design optimal d'expériences.

Ce manuscrit présente ensuite des développements méthodologiques permettant le *design* de plans d'expériences dans un contexte où les non-identifiabilités du système rendent le design bayésien

prohibitivement coûteux du point de vue computationnel et la matrice d'information de Fisher non inversible. Ces développements comprennent notamment une famille d'algorithmes permettant d'identifier localement la ou les axes de non-identifiabilité et de décider si ces non-identifiabilités sont inhérentes au modèle (auquel cas elles peuvent simplement être ignorées puisqu'elles n'impacteront pas la prédiction) ou si elles peuvent être attribuées au plan d'expériences particulier choisi. Durant notre présentation de cette famille d'algorithmes, nous dressons un parallèle avec les stratégies min-max de moteurs de jeux. Nous expliquons également comment ce développement fût guidé par des considérations sur la relation entre l'espace de paramètres, l'espace des comportements du système et les surfaces "iso-comportementales" de l'espace des paramètres. Malheureusement, les méthodes développées ne purent se traduire en une implémentation pratique, puisqu'un taux élevé de faux positif fut observé, que nous attribuons à des instabilités numériques lors de la diagonalisation de matrice.

Nous présentons donc dans un second temps une méthode plus classique, inspirée de la version bayésienne de la borne de Cramér-Rao. Dans cette méthode, les expériences sont planifiées pour identifier les paramètres. Cependant, pour pallier aux angles morts de cette méthode et démontrer sa légitimité, nous proposons également un pipeline computationnel d'évaluation de la qualité des plans expérimentaux proposés par notre méthode de design. Ce pipeline s'appuie sur le pouvoir prédictif obtenu par l'apprentissage de paramètres d'un modèle du système sur des jeux de données obtenus grâces aux plans expérimentaux suscités. Notre méthode de design utilise une distribution *a priori* des paramètres pour calculer un score d'informativité et maximise ce dernier grâce à une routine d'optimisation numérique stochastique nommée CMA-ES. Notre pipeline d'évaluation est plus complexe: pour chaque plan d'expérience à évaluer, on simule les jeux de données correspondant et l'on y *fit* des paramètres, en prenant soin de ne garder que les paramètres donnant un comportement au moins aussi proche du jeu de données que les vrais. Cette sélection implique d'obtenir en amont des milliers de jeux de paramètres, correspondant à autant d'exécutions de la routine d'optimisation et menant à un coût computationnel massif. Une fois ces jeux de paramètres sélectionnés comme expliqué précédemment, ils sont utilisés pour prédire la dynamique du système sur un jeu prédéfini de deux cents vingt expériences de validation. Ces prédictions sont ensuite comparées au comportement réel du système sur les expériences de validation and nous obtenons ainsi un jeu de deux cent vingt distribution de qualité de prédiction pour le protocole expérimental que nous cherchions à évaluer.

Pris de manière isolée, ces scores de qualité de prédiction ne se prêtent pas naturellement à l'interprétation. Cependant, ils peuvent

être utilisés pour comparer deux plans d'expériences. Pour ce faire, nous proposons plusieurs types de visualisations. Tout d'abord, des graphiques comparant les distributions de score de prédictivité pour plusieurs plans expérimentaux peuvent être utilisés. Pour un petit nombre de plan d'expérience, des boîtes à moustache classiques offrent un bon moyen de comparer ces distributions. Pour un plus grand nombre d'expériences, des *ridge plots* sont un excellent choix. Pour comparer deux designs, nous définissons des graphiques en grotte (*cave plots*) qui illustrent l'aspect intrinsèquement probabiliste de nos comparaisons. Enfin, pour comparer la multitude de design générés lors de notre étude de cas, nous présentons les matrices de comparaison, qui synthétisent l'information offerte par les graphiques en grotte.

Afin de rendre possible l'utilisation de ces deux pipelines et de permettre de générer la riche quantité de résultats présentés dans les visualisations suscitées, nous avons également développé un écosystème logiciel. Fwd:AD est une librairie pour le langage de programmation Rust qui permet de faire de la différentiation automatique par accumulation directe, et fondée sur les principes suivants: optimiser l'utilisation de la mémoire et minimiser les copies à l'interface de la librairie. Nous avons également développé des *bindings* Rust pour le solveur d'équation différentielles ordinaires de sundials, cvodes. Ces bindings utilisent Fwd:AD pour calculer les *sensitivities* de la partie droite de l'équation différentielle et tirer parti des fonctionnalités de résolution de *sensitivities* d'équation différentielle de cvodes pour résoudre à la fois le problème à la valeur initiale et le calcul de ses *sensitivities* par rapport aux paramètres. Nous avons également développés un langage spécifique de domaine par le biais d'une macro Rust qui permet de transformer un description ressemblant à du JSON du modèle en du code machine haute performance appelable depuis n'importe quel langage via l'*ABI* C. Enfin, nous avons développé une librairie Python qui s'interface avec le code machine décrit précédemment pour fournir une expérience utilisateur plus haut niveau. Elle permet de facilement simuler le modèle sur différentes entrées, paramètres, avec ou sans calcul des *sensitivities* mais offre également la possibilité de réaliser des graphiques ainsi qu'une interface vers la librairie de machine learning Theano. Grâce à cette interface vers Theano, nous avons pu expérimenter l'inférence Bayésienne et découvrir qu'elle demeurait hors de portée de nos capacités computationelles actuelles. Afin d'orchestrer les très nombreuses tâches qui composent nos pipelines, nous avons également développé Captain, un gestionnaire de tâches haute performance.

Ces contributions méthodologiques nous ont permis de mener notre étude de cas: une application du design optimal de plans d'expériences à la caractérisation chez *E. coli* de l'échappement aux traitement antimicrobiens via la production d'enzymes. Dans cette étude nous montrons tout d'abord que les plans d'expériences obtenus en utilisant

les vrais paramètres de notre système in-silico sont de meilleure qualité que les plans expérimentaux conçus par un expert. Pour expliquer celer, nous avons montré que ces plans expérimentaux peuvent être interprétés a posteriori comme l'exploration concomitante de quatre zones de l'espace des plans expérimentaux possibles, ces quatre zones pouvant être reliées à différent réponses du système aux antibiotiques. Nous nos sommes ensuite intéressés à un scénario plus réaliste, ou plutôt à deux sous scénarios. Dans ces scénario, nous avons utilisé un nombre limité d'expériences d'un expert avec un degré d'information changeant comme première étape d'un processus itératif de conception expérimentale. Les distributions de paramètres obtenues à partir de jeux de données issus de ces expériences ont ensuite été utilisés dans pour la conception de nouvelles expériences. La différence entre les deux sous scénarios tient à la manière dont ces conceptions d'expériences ont été conduites. Dans le premier sous-scénario les plans expérimentaux ont été calculés dans l'optique d'être utilisé seuls. Les expériences de l'expert ont servies à obtenir une distribution a priori mais seuls les plans expérimentai finaux sont évalués dans notre pipeline d'évaluation. Dans le second sous scénario, on cherche à concevoir des expériences pour compléter celles de l'expert, et c'est la combinaison de l'expérience de l'expert et de celles conçues algorithmiquement qui est évaluée. Le premier scénario souligne l'importance capitale d'avoir appris assez des expériences de l'expert pour concevoir des expériences qui aient du sens. Pour l'expert le moins informatif, tous les plans expérimentaux conçus algorithmiquement étaient ainsi de moindre qualité. Cependant, cela souligne néanmoins qu'investir trop dans les premières expériences peut s'avérer être inutile. En effet, une fois que l'on a obtenu un prior suffisamment informatif, chercher à le raffiner encore plus n'apporte qu'une maigre amélioration à la qualité des plans expérimentaux conçus dans la deuxième phase. Concernant le scénario où les deux types d'expériences sont combinés, les conclusions diffèrent quelque peu. Même pour la moins informative des distributions *a priori* obtenues par l'expert, utiliser la conception algorithmique d'expériences peut mener à de meilleurs designs qu'un expert utilisant le même budget expérimental total. En utilisant des distributions a priori plus informatives, cette possibilité devient une certitude: quand la première expérience a utilisé un certain budget (trois puits ou plus), il est toujours préférable d'utiliser le budget restant pour réaliser un plan expérimental conçu algorithmiquement que d'utiliser le budget total pour faire une expérience d'expert.

Enfin, ce manuscrit propose une discussion des résultats présentés et propose des perspectives de recherche future dans la continuité des travaux effectués pendant mon doctorat. Par souci d'exhaustivité, un bref résumé d'un travail de recherche sur un sujet différent mené pendant la durée de mon contrat doctoral est également présenté en annexe.

Acknowledgments

I would like to very warmly thank Eva Balsa Canto and Béatrice Laroche for reviewing my manuscript and Eugenio Cinquemani and Olivier Tenaillon for accepting to be part of my jury. The discussions we had were very interesting, and I am honored by the time and attention they dedicated to my work.

I would also like to thank Jakob Ruess and Gregory Batt who accepted to supervise my PhD work, and guided me through this three year adventure.

The scientific work contained in this manuscript would not have been possible without the support of many: Pasteur donors, administrative staff, open access advocates and open source contributors: thank you!

On a more personal note, I would like to thank all of the people I met while at Inbio. Virgile who paved the way to the PhD, Elise, Sebas and Chetan who walked it with me, and Viktoriia who is marching in our steps. I would also like to thank Elea, Francois, Olivier, Steven, Mariela, Albin, Zach and Davin, Achille, Lorenzo, Hélène and Sara.

A huge thanks goes to the friends I made at Pasteur: Jakub, Luc, Marie, Andrew, Anna and Fred. I am deeply thankful for the many hours I got to spend discussing with you, the sixth flour gang.

Je voudrais aussi remercier ma famille et mes amis qui fûrent à mes côtés pendant ces trois ans. Celles et ceux de l'ENS: Théotime, Auguste, Solène, Clara, Marie, Ramzy, Camille, Jean, Guillaume, Hadrien, Jun et Lucas ; d'AIV: Sophia et Hanifa ; et d'ailleurs: Margot et Julie.

Finally, thank you dear reader, and may the information you seek lies herein.

Table of contents

1. Introduction

1.1. Context

On November, 10th 1990, the University of Washington American footballers were not exactly having the day of their life when they lost to the University of California, Los Angeles 25 to 22 [94]. But in a luxury box, above the playing field, someone else's day was about to turn for the best. Lee Huntsman, then director of the University of Washington center for Bioengineering had struck a conversation with a pretty well-off entrepreneur with an interest for computer science named Bill Gates[26].

During this conversation, Huntsman told Gates how the University of Washington had been trying to recruit a prominent California scientist named Lee Hood for the last ten years. Hood had already acquired a solid reputation by then. He had invented and commercialized scientific instruments such as the first gas phase protein sequencer (1982), a DNA synthesizer (1983), a peptide synthesizer (1984) and an automated DNA sequencer (1984) which had proved capital in the Human Genome Project. But as things stood, the University of Washington could not afford to hire Hood. Huntsman, its director, convinced Gates to meet with Hood over dinner. The dinner apparently went well because Gates went on to donate 12 million US dollars to the university, and Hood proceeded to move to Seattle and started his lab there. In 1992, he established the university's Department of Molecular Biology, and refined his thoughts, preparing – maybe unknowingly – to make a remarked entry in the 21st century.

The University of Washington is located in Seattle, in the state of Washington, and not in Washington city in D.C.

Indeed, in 2000, Hood, together with a chemist named Ruedi Aebersold, and an immunologist named Alan Aderem founded the Institute for Systems Biology, and christened a field in and of itself: *systems biology*. Leroy Hood's conception of what is systems biology can be found in a 2001 paper[49] he co-authored with Trey Ideker (whose PhD

he supervised), and Timothy Galitski (another researcher of the newly founded Institute for Systems Biology). In this paper, systems biology is depicted as a direct consequence of the Human Genome Project, and described as follows:

> Systems biology does not investigate individual genes or proteins one at a time, as has been the highly successful mode of biology for the past 30 years. Rather, it investigates the behavior and relationships of all of the elements in a particular biological system while it is functioning. These data can then be integrated, graphically displayed, and ultimately modeled computationally.

Put differently, systems biologists are not interested solely in the atomic building blocks (e.g. a protein, a gene) of a given system (e.g. a bacteria) but rather in the system as a whole, as an ensemble of the interactions of each of its components. Systems biology is not only complex, it advocates for and embraces complexity, by bringing and linking together different scales of time and size.

A key player in the systems biologist tool-belt is mathematical modeling (what Ideker et al. refer to as "modeled computationally"). A mathematical model is a an abstract representation of a system, using objects, concepts, and methods coming from mathematics. More and more frequently, mathematical modelling also involves techniques and tools from computer science to gain insight on the model (and the modeled system). Mathematically modeling a system can be done for two reasons: to better understand the system, or to simulate the system's behavior faster and cheaper than by running actual physical experiments on it.

In the first case, one tries to match the structure of the system, so that deductions on part of the model will translate to deductions on the corresponding part of the system. For example, if we were interested in bicycles, we could attempt to model how forces ares transmitted from the leg of the operator, to the pedal, to the crank, to the chain, to the back wheel, and on to the road, also taking into account air resistance, and bringing those together via Newton's second law. If we were to model our bike in this way, we could find equations for each relation between components, and put them together to model how energy spent by the cyclist translates into acceleration of the bike. If we were to run experiments, either on the cyclist biking as a whole

or on isolated components, we could then say that we not only know how the cyclist's effort gets converted into bike speed, but we would also have understood how energy is transferred from the bike to the crank, from the chain to the wheel, and how the wind can influence the overall speed. Studying our bike as a whole would have brought us fundamental knowledge of how the bike works.

But sometimes, we are not so much interested in understanding how a system works, but only in how the system will evolve in time. In other words we do not care so much about the fundamental physical laws that explain how our input to the system will affect the system's state, but we rather want to predict what state will be obtained from a given input. This can typically be the case when the model will be used to control the system, i.e.. to bring the system to a given state. As the model provides a way to quickly test many different inputs, one can efficiently search for the input leading to the desired state and apply it. This can also be the case when we try to use models to inform our policy making, when dealing with climate change or a pandemic for example.

A remarkable example of mathematical modeling guiding public health decisions is the debate that took place during the second half of the 18th century regarding the inoculation of smallpox (a process called "variolation"). Usually spread trough the air and infecting the body via the upper respiratory tract, the smallpox virus leads to a milder infection when inoculated via a superficial cut on the skin, a method that still induces immunity. Statistical studies showed that tough not entirely safe, this voluntarily infection still had a favorable benefit-risk ratio, and a heated scientific debate ensued which saw figures such as d'Alembert and Bernoulli [17, 27] weigh in, ultimately resulting in a large spread of the technique in the western world, and Bernoulli's paper which is often described as the first epidemiological model.

1.2. Motivation

No matter the final use of a given model, the ones we will be interested in in this document are based on equations in which quantities appear that can be classified as follows:

- *Inputs* are the quantities fixed by the experimenter or the environment. They are considered to be known exactly.
- *State variables* are the quantities that the model is explaining, they are not known in advance and can vary when the input varies. Some of them are measurable.
- *Parameters* are intrinsic quantities that exist independently of the experiment but that we do not know, or at least not precisely. They are the quantities we try to identify when characterizing the system, whether it is to understand it or to predict its behavior later on.

When simulating a system from its model, fixing the inputs and parameters is enough to predict the state variables (though the exact nature of the predictions may vary depending on whether our model is deterministic or stochastic). Hence, it is no bold claim to say that correctly identifying good parameter values is capital to make a useful model. Parameter values are found by the experimenter by: a) planning an experiment to perform, b) performing said experiment and gathering data and c) finding parameter values so that the simulation matches the data. We can thus extend our claim to say that, if the parameters found depend on the dataset, and the dataset depends on the experimental plan, then identifying a good experimental plan is what is capital to learn a useful model.

The space of possible experimental plans is a large, if not infinite one. Exploration of such a space is well tackled by mathematical and computational methods, a field called “design of experiments”. Because one often tries to achieve the best design possible, design of experiments is often referred to as **optimal design of experiments**. Interestingly, this trend of automation also concerns other steps of the research process: with rising automation of bench work, companies envisioning to offer biology experiments as a service, and analytical pipelines moving to the cloud, it is only natural to try to also enlist the help of machines for the experimental design phase.

In order to study the practical application of experimental design to the characterization of a real-life model, I leveraged the work of a senior PhD student of our lab: Virgile Andreani. During his doctoral work[3], Virgile developed a model of enzyme mediated response to the β-lactam antibiotics and used it to propose a classification of bacteria

(sensitive, tolerant, resilient, and resistant) going beyond the classical sensitive, intermediate, and resistant (SIR) classification. This classification is of high interest, but its practical use is hindered by the fact that the characterization experiments are more involved than those used in the SIR method. Fitting the model requires several optical density (OD) measurements of multiple cell populations exposed to different concentrations of antibiotics, with frequent measurement points. Hence, using design of experiments to reduce the number of experiments needed to well characterize the model could prove key to a wider adoption of this new antibiotic resistance classification.

Moreover this model is of interest because it is not merely a toy example built specifically to apply optimal design methods to, but rather a complex model, constructed to explain actual experimental data. The model has seven state variables, seventeen parameters (two of which are fixed), and consists of ordinary differential equations that have been derived from a partial differential equations model.

As many biological models, our antibiotic resistance model suffers from unidentifiability. This non-identifiability is both:

- Structural: the model's output is strictly equivalent for different parameter values.
- Practical: the model's output hardly changes across different parameter values, often below the threshold that can be perceived through the measurement apparatus.

Non identifiability can also arise only for some value of the true parameters (typically when zeroing a parameters would turn off some upstream mechanism, making the parameters related to what is downstream of this gene non identifiable), it can depend on the chosen input (once again stressing the importance of identifying a good experiment to perform) and it can finally also depend on our measuring capacity (frequency and quantity of measurements, amount of noise, which state variables are measurable).

In summary, I had identified a biological question for which optimal experimental design was a clear next step, but which, because of its complexity would require substantial mathematical and computational developments to fully work out.

1.3. Approach

We thus had our goal: to design optimal characterization experiments for the antibiotic resistance model. We also wanted the methods and software developed along the way to be generic and reusable. Ideally, our optimal design pipeline would become a standard tool used in all of our team's projects and even beyond.

As will be developed later on, non-identifiability presents an enormous challenge for optimal experimental design. To tackle it, we had to go beyond the classical line of thought that a good experimental design is one that identify well some fixed "true parameter" value, and develop an approach where parameter values were seen not so much as an end, but rather as a mean to obtain what we really seek: the ability to simulate well the system and predict its behavior. This change of paradigm guided the entirety of our approach, and inspired many of the proposed developments.

We delved into the Bayesian framework, which naturally support this shift of paradigm. We also studied the usability of the Fisher Information Matrix (FIM) in a non-identifiable setting. More precisely we explored possible mathematical methods to identify linear local approximations to classes of parameters resulting in equivalent behaviors. Nonetheless, the approach we finally proposed and used is an hybrid one, where we design for parameter identification in an iterative setting but assess the quality of the computed designs for their prediction power.

Our design and assessment procedures are both computationally intensive. To enable our research we thus designed large scale automated pipelines, and researched questions pertaining to software engineering, to offer tools that are both high-performance and user-friendly. Finally, we validated the relevance of our method and implementation by a case-study of optimal designs for the characterization of the antibiotic resistance model. In this case-study, we compared optimal designs to those of an expert to assess their quality. We did so by first computing designs using the true parameters of our in silico system to validate that a perfectly informed optimal design can outperform the expert. We then simulated realistic iterative learning scenarios and concluded by an analysis of the tradeoffs implied by doing iterative design on a fixed total experimental budget.

1.4. Contributions

During my PhD I made the following contributions. I proposed a mathematical way to obtain a matrix akin to a FIM but relating to model behavior and not parameter values. I propose a discussion of the relevance of the Bayesian Cramér-Rao bound, also known as one of the Van Trees inequalities to iterative optimal design, and its computational intractability when priors only exist in the form of samples. I also developed two pipelines, one for the design of experiments, and one for the assessment of their predictive power as a measure of their quality. To implement these pipelines, I developed the following software:

- Fwd:AD, a Rust library for automatic differentiation in forward-mode. This library was engineered from the ground-up with one founding principle: optimize memory usage, and minimize memory copy at the interface of the library.
- Bindings to sundial's ODE solver cvodes, using Fwd:AD to compute the sensitivities of the right-hand side and leverage sundials sensitivities capabilities to integrate both the initial value problem solution and its derivative with respect to the parameters.
- A Rust macro implementing a domain specific language allowing to transform a user-friendly JSON-like description of an ODE model into high-performance machine code callable from any language using the two libraries above.
- A python library, interfacing with the machine code generated above to provide an higher-level user-friendly experience, allowing to easily simulate the model on various inputs, parameters, with or without the sensitivities, but also offering plotting capabilities as well as bindings to the theano machine-learning library for the model, to allow experimenting on Bayesian inference.
- An high-performance job manager to orchestrate the millions of jobs that have been run on the cluster.

More specific to the case-study at hand, I also developed structured representations of experimental protocols, fitting results, and datasets required by the pipeline.

Finally, I demonstrated the relevance of optimal design for the characterization of the antibiotic resistance model. I showed that – when given perfect knowledge of the system's behavior – optimal design algo-

rithm based on the Fisher Information Matrix can propose experimental plans that outperform those of an expert. I also showed that optimal design maintains its advantage when performed in an realistic iterative setting. To this end, I illustrated how considerations on the intuition behind optimal designs can guide such demonstration, and proposed visualizations and methods that can be of general use when assessing the quality of experimental designs.

1.5. Outline

The rest of this document is organized as follows.

Chapter 2 presents a state of the art on optimal design, highlighting the creation of the field at the beginning of the twentieth century and current topics regarding optimal design in systems biology. It also proposed a brief general state of the art on computational methods used in systems biology.

Chapter 3 presents the scientific background on which my PhD work was built, introducing first mathematical considerations, and then computational ones.

Chapter 4 presents the methodological developments I contributed. First, a mathematical method to design while embracing the non-invertibility of the FIM in a context of non-identifiability is presented. Then, the method we used in practice is introduced, where designing experiments is done for parameter identification, but the designs are assessed for their predictive power. Finally, the software produced is extensively presented.

Chapter 5 presents our case study. It first introduces the work done by Virgile Andreani to develop the model as well as some experimental context, and then demonstrates the potential that optimal design has to improve over an expert's experiment design.

Chapter 6 presents the realistic instantiation of said case-study, where optimal design is used to in an iterative fashion. It notably studies, in a setting where two sets of experiments are to be performed iteratively, how the repartition of a fixed budget between the first and second set of experiments affects the overall quality of experiments.

Finally, Chapter 7 concludes this manuscript by presenting a summary of the work, discussing how it articulates with the existing literature, and presenting perspectives for future research on the topic.

2. State of the Art

This chapter presents an history of optimal experimental design, and reviews current topics regarding optimal design in systems biology. In addition, a brief general state of the art on computational methods used in systems biology is provided.

2.1. Optimal design of experiments

2.1.1. History

Kirstine Smith
(1878 – 1939)

Joseph Diaz Gergonne
(1771 – 1859)

Stigler gave his name to Stigler's law of eponymy, which states that a scientific discovery is never named after its original inventor… Quite fitting for our discussion on optimal design's discovery!

In a 1916 letter [72] to Ronald Fisher, Karl Pearson described a certain "Frøken K. Smith" as "one of the most brilliant of the younger Danish statisticians". The Frøken Smith in question was Kirstine Smith, a Danish statistician who went on to complete her doctoral degree under his supervision at the University of London in 1918. Her thesis [87], entitled *On the Standard Deviations of Adjusted and Interpolated Values of an Observed Polynomial Function and its Constants and the Guidance They Give Towards a Proper Choice of the Distribution of Observations* laid the first stone of the field that later became optimal experimental design.

Other claims to the title of inventor of optimal design I have found during my research include: Fisher (1936) [32] and Elfving (1952) [31], whose publications are all posterior to Smith's thesis. A serious candidate to the title of inventor of optimal design would be Joseph Diaz Gergonne, whose 1815 article [36, 37] was rediscovered by Stephen Mack Stigler in 1974 [90]. However, if Gergonne's paper contains considerations on optimal design, he does not – to the best of my knowledge – report a solution or mathematical result for optimal design.

Kirsten Smith work is, among other things, remarkable for her definition of optimal design and her description of how, while some designs

are intuitively optimal, other imply balancing various sources of measurement noise. Indeed, the very beginning of her introduction, still very much relevant today, reads as follows:

> In all sorts of experiments which are not simple repetitions but have at least one varying essential circumstance or indefinite variate the experimentalist is confronted with a choice in regard to the values of that variate. If the experiments be quite simple the question may be without great importance; but when their requirements as to time or expenditure come into account the problem arises, how the observations should be chosen in order that a limited number of them may give the maximum amount of knowledge. [...]
>
> When we deal with, for example, a linear function which it is possible to observe with the same accuracy for all values of the indefinite variate we should not hesitate to put the observations in two equally big groups as far apart from each other as feasible. But if the standard deviation of the observations be a function of the indefinite variate and increases with the distance from the middle of the range, where is then the point in which the advantage of removing the two groups of observations from each other just counterbalances the disadvantages of increasing the error of observations? The problem becomes very complicated for functions of higher degrees.

This seminal work of Smith is focused on G-optimal (see 3.1.2.2.3) designs for polynomial functions, of up to order 6 [42]. In that case optimal experiments can be computed analytically and a general solution proposed. This ease of computation can be generalized to numerous applications of optimal design, including to linear or polynomial models (see Elfving, 1952 [31]), and introductory books to "optimal design" are in fact implicitly restrained to such cases. Other common applications of optimal design where computations can be carried out without too much pain are experiments where the user input can only take a finite discrete state of values. They are studied as *Combinatorial designs* [91] (or *factional designs* when the input space remains finite but becomes too big to be exhaustively explored). Clinical trials [73] are a prototypical examples of combinatorial design.

However these applications of optimal design are all rather orthogonal to ours. Applications to more complex, non-analytically solvable

models of dynamical systems are more recent, with the first review being published in 1974 [62]. An high-quality 2008 review of this more advanced optimal design techniques by Franceschini et al. is provided in [35]. Outside of systems biology, optimal design has been applied a wide range of topics, including the drying of rice (Goujot et al., 2012) [40], chemistry (Bauer et al., 2000) [16] or thermal degradation of food nutrients (Balsa-Canto et al., 2007) [6]. Regardless of the domain of application, studying the robustness of iterative design is an active research topic [55, 96].

2.1.2. Optimal design in systems biology

However, our work is not the first to focus on applying optimal design methods to questions coming from systems biology, as is shown by the 2008 methodological review of Banga et al. [14] Even though the complexity of our model (and its non identifiabilities) prevented us from resorting to Bayesian optimal design (cf. sections 3.1.1.3, 3.2.2 and 4.3.5), prior uses of it for biological systems influenced my research. Liepe, 2013 [58] for example pushes the use of a complete Bayesian Framework to the computational limits. In it, experiment design is in fact limited to experiment selection from five predefined protocols, but because of its entirely Bayesian nature, it allows for a natural way of optimizing for prediction power. Other works on Bayesian optimal design include those of Vanlier et al. 2012 [95], Ryan et al., 2016 [84] or Pauwels et al., 2014 [71].

Another group of studies that are somewhat similar to ours are the ones where optimal design is performed for stochastic models, often for experiments at the single cell level. In Fox, 2019 [33, 34], the authors devise a way to compute the FIM from a finite state projection (FSP) approximation of the reaction system, and propose an in silico study as well as an “experimental” one, reusing previously collected data. In Ruess, 2013 [82], the authors were interested in optimal design for a stochastic model where the kinetic rates themselves were random processes. In Zimmer, 2016 [101], the author uses a method mixing multiple shooting for stochastic systems and the linear noise approximation to compute the FIM of non-linear stochastic systems, in line with work done in Komorowski 2011 [54] on the use of the linear noise approximation for FIM computations. Also related to our work is

optimal design for the sake of model selection. For example Bandiera et al. 2020 [13] designs experiments to discriminate between two models, and hence search for the input on which the two models' predictions differ the most.

That being said, many studies use a more classical setting, even more similar to ours, where the FIM is used with a deterministic ODE model. In *Iterative experiment design guides the characterization of a light-inducible gene expression circuit* (Ruess et al., 2015) [83] for example, the authors present an application of optimal design to a protein production system. The system is modeled via six reactions, involving three bio-chemical species (the mRNA, the dark protein and its fluorescent version), and a total of nine parameters. Interestingly, the authors decided to derive moment equations up to order four for this originally modestly sized system, which leads them to sixty-five ordinary differential equations. Model-wise, they are thus interested in a model which has less parameters than the one presented in our application (fifteen versus nine), but much more equations (sixty-five versus seven). This large difference in equations to parameters ratio may be the cause of the main difference between their application and ours: identifiability. Indeed, there is no statement, in the main-text or supplementary material they published of unidentifiabilities in their model, while dealing with non-identifiability is a central point of this thesis. Another difference is that our study presents a quantitative assessment of the tradeoff between investing on preliminary experiments, or saving experimental budget for designs created with more information. A remarkable point of their study is the application to actual biological experiments at the bench, where the superior quality of their design over both an expert's and random designs was demonstrated. Another noteworthy aspect is their use of an hybrid method, between Bayesian inference and frequentist design.

In *Optimal Experimental Design for Parameter Estimation of a Cell Signaling Model* [9], Bandara et al. also propose a study of optimal design on a deterministic model with at-the-bench demonstration. Their paper pinpoint early on non identifiability as a general issue in systems biology, both for parameter identification, and for the use of the model for prediction. However, they forgo the FIM an adopt a succinctly described custom method also relying on the sensitivity of the model's output to the parameters to produce a variance matrix, to which they then apply

criterions similar to those used for the FIM. Their model remains modest in size, as it consists of six parameters and four state variables. On top of the discussion regarding non-identifiability, another echoes the points developed in this manuscript: the sometimes counter-intuitive nature of the designed proposed by the authors' algorithms, and their later rationalization, which completes our discussion section 5.2.3 and reminisces us of research on interpretable machine learning [66]. Finally, Bandara et al. stress the importance of ensuring the feasibility of the proposed designs to achieve practical applications on optimal experimental design.

Lastly, in *On-Line Optimal Input Design Increases the Efficiency and Accuracy of the Modelling of an Inducible Synthetic Promoter*, Bandiera et al. [12] extends their work of 2018 [11] and propose an in-silico study of optimal experimental design for an inducible promoter in *s. cerevisiae*. They propose an assessment of the quality of experimental protocols, and a cross comparison, based on the spread of the inferred parameters, which as discussed hereafter is not suited to systems with non-identifiability such as ours. Design-wise, they use a FIM based method very similar to ours, once again save for the fact that their FIM is invertible. Noteworthy, they tackle online optimal design in a way which we do not. Overall, their study is close in spirit to ours, tackling issues such as the intuition behind experimental designs, and comparing experimental designs to one another. However, our embraces non-identifiabilities through computations of predictive power at a very large scale.

2.2. Computational aspects

Regarding computational aspects of our study, prior art is also abundant. Quite a number of papers have been dedicated to methods for non-linear optimization, both for model fitting or experiment informativeness maximization. Given that we do not discuss further model fitting (apart from our attempts at Bayesian inference discussed in section 4.3.5) or experimental input space exploration, I will simply give a few pointers to relevant publications. Recent reviews are provided by Raue et al., 2013 [77], Reali et al., 2017 [78] and Villaverde et al, 2018 [97]. Recent methods include CMA-ES (Hansen, 2016) [43], the "hy-

brid global method" (Balsa-Canto et al., 2008) [8], enhanced scatter search (Egea et al., 2009) [28]. Toolboxes wrapping different optimization algorithms, and sometimes heuristics to select one have also been developed, such as MEIGO (2014) [29]. More general toolboxes, trying to offer a turnkey experience, where the user inputs a high level model description and most fitting (and sometimes optimal design) routines are provided have been released, including AMIGO2 (Balsa -Canto et al., 2016) [7] and Data2Dynamics (Raue et al., 2015) [75].

Another aspect of this thesis is the work presented on automatic-differentiation. Hoffman, 2015 [48] cites Wengert, 1964 [100] as the first article on automatic differentiation, which is there presented in forward mode. Griewank 2012 [41] states that many contributions in the late sixties and early seventies converged towards the backward-mode of automatic differentiation, but the first explicit statements can be found in Linnainmaa, 1970, a Finnish master thesis [60] (English translation in [59]), with an independent nearly simultaneous discovery in Ostrovkii et al., 1971 [68] (in German). A general introduction to automatic differentiation can be found in Hoffman, 2015 [48]. More recently, more and more libraries for automatic differentiation have been released, both generic ones (ADOL-C [99], CasADI [2], ad [57]) and some dedicated to more specific problems (the MC Stan math library [89] is used by MC Stan, a Bayesian inference library; pytorch [70], tensorflow [1] and theano [79] are all example of machine-learning libraries that implement an automatic differentiation engine). Automatic differentiation is identified as a tool for optimal design of experiments around 2000 by Bauer et al. [15, 16], but there is still not an unifying toolbox proposing automatic differentiation capacities for the ODE models usually found in systems biology. An example of such attemps would be casiopeia (Buerger, 2017) [20], which is unfortunately unmaintained. However, despite the abundance of automatic differentiation solutions, those interfacing properly with ODE solvers remain few [61]. The JuliaDiff [51] set of packages can be cited as a solution to ODE solving with automatic differentiation capabilities, but Julia's notoriously long "time-to-first-plot" [69] prohibited us from using it in our large-scale pipeline. Outside of automatic differentiation IDEAS (Munoz-Tamayo) [67] relies on symbolic differentiation to provide a toolbox with FIM computing capabilities. In my experiences, symbolic differentiation on our model lead to too big equations which were practically unusable.

3. Background

This chapter introduces concepts from mathematics and computer science required to understand the contributions presented in the following chapters.

3.1. Mathematical considerations

I'll first begin by exposing some mathematical aspects of modelling in systems biology. I will present how ODE models are used in systems biology, how they relate to generated datasets, and the formalism underlying experimental design. Through pedagogical examples of optimal experimental design, I will introduce the question of non-identifiability which is essential to the work presented in the thesis.

3.1.1. Models, likelihood and Bayesian inference

3.1.1.1. Ordinary differential equations based models

Our study focuses on the modeling of biological systems by systems of ordinary differential equations of the form:

$$x'(t, \theta, u(t)) = f(t, x(t, \theta, u(t)), \theta, u(t))$$

where x is the vector of state variables, θ is a constant vector of parameters and $u(t)$ is an imposed exterior input to the system (either modelling a control the experimenter has on the system or environmental variations). Oftentimes, we will consider u to be fixed and be interested in x as a function of the space of parameters θ to the space of functions of time $t \mapsto x(t, \theta, u(t))$. Often, we will omit the assumed

fixed input, and simply write $x(t, \theta)$. When writing about experimental design, the state variables will sometimes be made dependent on an experimental plan ξ rather than an input u, to stress the fact that environmental conditions are assumed to be fixed.

Practically, such models are used by obtaining datasets, that is a collection of pairs (t_i, y_i) where we assume that the y_i are random variates distributed according to a probability distribution parametrized by $x(t, \theta)$, called the noise model and denoted $\mathcal{D}_{x(t,\theta)}$. This noise model is assumed to have a mean, equal to $g(x(t, \theta))$ where g is the observation function. The observation function is typically "simple", representing the fact that we can often only observe some of the state variables, and sometimes only a quantity derived from them (for example the logarithm, or an affine rescaling). Moreover, we assume that the noise model admits a probability density function, denoted $p_{x(t,\theta)}$, and that all y_i are independent realization of the same noise model, up to its parametrization by $x(t, \theta)$.

3.1.1.2. The likelihood of a dataset

When characterizing the system, we will be mostly interested in one quantity: the likelihood. Denoted l, the likelihood is a function which associates the probability of the (fixed) dataset we have observed (with a given fixed input) to a parameter vector θ. Put differently, it represents how well this parameter vector would explain the dataset. It is computed as:

$$l(\theta) = \prod_i p_{x(t_i,\theta)}(y_i)$$

A common way to use this likelihood is to try to maximize it: for a given dataset that we obtained with a given input, we can try to find the parameter vector that maximizes the likelihood. This parameter vector that maximizes the likelihood is called the Maximum Likelihood Estimator (MLE). Let's denote θ^* the MLE for some fixed dataset. We can then consider that the system behaves as simulated by our model instantiated with $\theta = \theta^*$.

3.1.1.3. Bayesian inference

Finally, another approach to model characterization is the Bayesian one. Bayesian statistics formalize the intuitive way in which we reason about a given parameter being more or less "likely" to be the correct one, by putting probability distributions on parameters vectors. We can then go from a *prior* distribution whose density is denoted $p(\theta)$ to a *posterior* distribution learned on the data and denoted $p(\theta|y)$. This is done using Bayes' formula:

$$p(\theta|y) = \frac{l(\theta)p(\theta)}{\int_\iota l(\iota)p(\iota)}$$

which shows that the posterior is the product of the likelihood and a prior, normalized to integrate to one so as to be a proper probability distribution.

3.1.2. Optimal design of experiments

Whether using a Bayesian or Frequentist approach, there is a clear link between the dataset and the learned parameter values. To obtain high-quality datasets, rich in information about our system, we can improve our choice of system dynamics to observe. Hence, as put by Kirsten Smith "the problem arises, how [...] a limited number of observations may give the maximum amount of knowledge". Guiding this choice mathematically is a field in and of itself, the field of optimal design of experiments.

3.1.2.1. Bayesian optimal design

Optimal design is the act of designing experimental plans that yields a maximal amount of information about the model. The go-to method to quantify the information we have on a model's parameters is to use Bayesian statistics. Bayesian statistics formalize the intuitive way in which we reason about a given parameter being more or less "likely" to be the correct one. Let's say that we have two experiments, A and B, that we carry out, and that we perform Bayesian inference on datasets A and B. If A tells us that a few parameters are very likely to be the good ones, but B tells us that more parameters may be the good ones (but

are so with less probability), we can intuitively think that A is a better experiment than B (cf Figure 3.1).

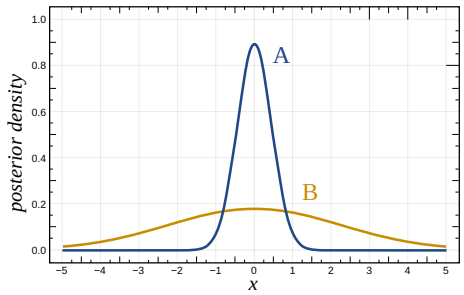


Figure 3.1.: Two posterior distributions, coming from two experiments A and B. The one coming from A is narrower, and intuitively better, because less parameters are plausible.

With such a framework, and in an ideal setting, quantifying whether a given experimental design is of interest is near straightforward: you simulate the outcome of the experimental design, if needed several times to account for variability in the dataset realization, you compute the posterior distribution on parameters and use some measure of the spread of this posterior (covariance, entropy…) to assess its quality. Noteworthy is the fact that this measure may or may not need to be quantitative depending on the intended use: inside an automated algorithm optimizing information, it needs to be quite quantitative, and probably to be a single number that can serve as the scoring function of a minimizing or maximizing algorithm. On the other hand, if you're trying to analyze and compare several designs by hand, simply looking at the shapes of the posteriors may be enough.

Bayesian experimental design abstracts from the idea of simulation and reasoning only about probability distributions. For some utility function U that associates some real-valued score of informativeness of this distribution (e.g. gain in Shannon information from the prior, Kullback-Leibler divergence with the prior) to a posterior probability density, the quality Q_ξ of an experimental design is:

$$Q_\xi = \int \underbrace{U\left(\underbrace{\theta \mapsto p(\theta|y,\xi)}_{\text{posterior pdf}}\right)}_{\text{measure of posterior quality}} \underbrace{p(y|\xi)}_{\text{a priori probability of } y} \, dy$$

where

$$p(y|\xi) = \int p(\theta)p(y|\theta,\xi)d\theta$$

and

$$\theta \mapsto p(\theta|y,\xi) = \frac{p(y|\theta,\xi)p(\theta)}{p(y|\xi)}$$

3.1.2.2. Frequentist optimal design using the Fisher Information Matrix

Unfortunately, computing such Bayesian informativeness score is often too costly, if at all feasible, and another metric needs to be used. The Fisher Information is often picked.

3.1.2.2.1. Definition

The Fisher Information Matrix I of a random variate $X(\theta)$ with n parameters θ_1 to θ_n, and probability density $f(X|\theta)$ is defined element-wise as :

$$\mathcal{I}_{i,j} = \mathrm{E}_{X|\theta}\left[\left(\frac{\partial}{\partial\theta_i}\log f(X;\theta)\right)\left(\frac{\partial}{\partial\theta_j}\log f(X;\theta)\right)\right]$$

where $\mathrm{E}_{X|\theta}\left[\cdots\right]$ means that we take the expectation over the random variate X, with parameters θ fixed.

Under certain regularity conditions, this is equal to

$$\mathcal{I}_{i,j} = -\mathrm{E}_{X|\theta}\left[\frac{\partial^2}{\partial\theta_i\partial\theta_j}\log f(X;\theta)\right]$$

The Fisher Information Matrix (FIM) can be interpreted in two ways, justifying its use as a proxy for informativeness.

1. In its second form, the FIM can be interpreted as the expected curvature of the log-likelihood for a given realization of the data. Hence, if the likelihood has a local maximum, the FIM is an expectation of the first non-null term of order greater than 1 around that point, and the higher it will be, the more difference in likelihood there will be between our estimate and its neighbors.
2. The Cramér–Rao bound is a theorem which states that, for a random variate X parametrized by a parameter vector θ, and having

an *invertible* FIM I, any *unbiased* estimator of θ has a covariance matrix greater or equal to I^{-1}.

The matrix inequality $A \geq B$ is understood to mean that the matrix $A - B$ is positive-semidefinite, which is called the Loewner order.

3.1.2.2.2. Interpreting Fisher information matrices through their eigen-decomposition

Fisher information matrices – like covariance matrices – are always symmetric (and hence diagonalizable in an orthonormal basis) and positive semi-definite. As matrices, they canonically define a linear map, and because of the aforementioned properties, the image of the unit circle by this map will always be an ellipse. Fisher information matrices (like covariance matrices) are thus entirely defined by their eigen-decomposition, as illustrated in Figure 3.2. Interestingly, positive semi-definite matrices are invertible if and only if none of their eigenvalues is zero. In that case, their inverse is also positive semi-definite and:

- The eigenvectors of the inverse are the same as the one of the original matrix.
- The eigenvalues of the inverse are the inverse of the eigenvalues of the original matrix.

These properties are especially useful when reasoning about Fisher information matrices and their inverse as covariance matrices (see the Cramer-Rao bound).

3.1.2.2.3. Going from a Matrix to a scalar

Fisher information matrices are intrinsically multidimensional beasts, representing the information we have on each parameters and some of their relationship. Yet, to compare Fisher information matrices to one another, especially in optimization routines, it is often needed to summarize them as a single scalar value. As we have seen before, Fisher information matrices are best understood through their representation as ellipses, and *criterions* to summarize a matrix into a scalar are often inspired by this representation. A few common criterions are:

- **D-optimality:** Minimizing the determinant of the inverse of the FIM. This is equivalent to maximizing the determinant of the FIM itself. Geometrically, this is equivalent to maximizing/minimizing the volume of the corresponding ellipsoid. This is the criterion I'll use in this manuscript, if not specified otherwise.

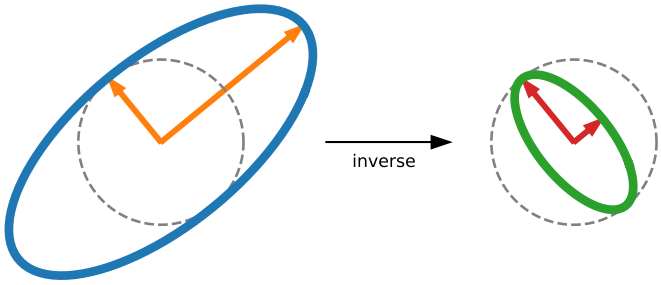


Figure 3.2.: **Left:** Image (blue solid line) of the unit circle (gray dotted line) by a positive-semidefinite matrix M. The eigenvectors are drawn in orange, and with a norm equal to the corresponding eigenvalue. **Right:** Image of unit circle (gray dotted line) by the *inverse* of M. The eigen vectors (in red) have the same direction, but their corresponding eigenvalues are the inverse of the the corresponding eigenvalues for M.

- **A-optimality:** Minimizing the trace of the inverse of the FIM ("A" stands for average). Contrary to the D-optimality criterion, this criterion requires to compute the inverse of the FIM (trace and inverse do not commute). Geometrically, this is equivalent to minimizing the sum of the length of the axes of the covariance ellipsoid.
- **G-optimality:** Minimizing the maximum diagonal entry of the so-called hat matrix $S\left(S^TS\right)^{-1}S^T$ where S is the sensitivity matrix. This is akin to assuming additive Gaussian noise and minimizing the maximum variance of trying to re-predict the same experiment that was used to learn. It was the first ever proposed optimality criterion (see 2.1.1).

3.1.2.3. Examples of information quantification using the FIM

3.1.2.3.1. Gaussian distribution

Both the usability and shortcomings of the FIM can be illustrated on the model-organism of probabilities: the Gaussian distribution. Let's consider a random variate X, that we know to be normally distributed, with mean μ and standard deviation σ. Its probability density function (PDF) knowing these two parameters – noted $f(X|\mu,\sigma)$ – is given by:

$$f(X|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

We are trying to simultaneously identify the parameters μ and σ by sampling n independent realizations x_1 to x_n of X, and looking for the maximum likelihood estimate for μ and σ, i.e.. the values of the two parameters that maximizes $\prod_{i=1}^{n} f(x_i|\mu,\sigma)$. It is noteworthy to remark that there is no MLE if $n = 1$, as the optimum value for μ is x_1 but the likelihood strictly increases towards infinity when σ goes to zero. For n greater than one the MLE for μ is the sample mean, and the MLE for σ is the (biased) sample standard deviation.

The log of the probability density function of a Normal distribution is

$$\log f(X|\mu,\sigma) = -\log\left(\sigma\sqrt{2\pi}\right) - \frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2$$

For a simple distribution like the Normal one, we can obtain the sensitivity of this log-pdf with respect to each of the two parameters analytically. These sensitivities can then be multiplied pair-wise, and integrated with respect to the probability distribution of X, according to the FIM definition, resulting in:

$$\mathcal{I}(\mu,\sigma) = \begin{pmatrix} 1/\sigma^2 & 0 \\ 0 & 2/\sigma^2 \end{pmatrix}$$

The Fisher information obtained from independent samples is the sum of the information of each sample, so the total information from our n samples is $n\mathcal{I}(\mu,\sigma)$.

The FIM allows us to make the following deductions:

1. The information expected to be obtained decreases when σ increases, which matches the intuition that noise in data hinders learning.
2. It is invertible, so our system is locally identifiable[81].

3.1.2.3.2. A growth experiment

The previous example remained very theoretic. This next example is a bit closer to actual uses of the FIM as a tool to design experiment.

Let's consider a population of bacteria, whose growth is governed by a logistic equation. We want to take only two measurements of the population size (maybe because we have to perform tedious colony forming units measurements) to fit our model of logistic growth below:

$$n(t) = \frac{k}{1 + \frac{k-n_0}{n_0} e^{-rt}}$$

Here, n is the size of the population, and the sole state variable, n_0 is the (known) initial population size, and r and k are two parameters representing respectively the growth rate and the carrying capacity of the model. Logistic growth is a model where the population undergoes an initial quasi-exponential growth of rate r and then progressively saturates the medium, at which point its growth slows down to asymptotically reach the carrying capacity k. Both the time-dynamic and its sensitivities to both parameters can be computed analytically (see Fig. 3.3).

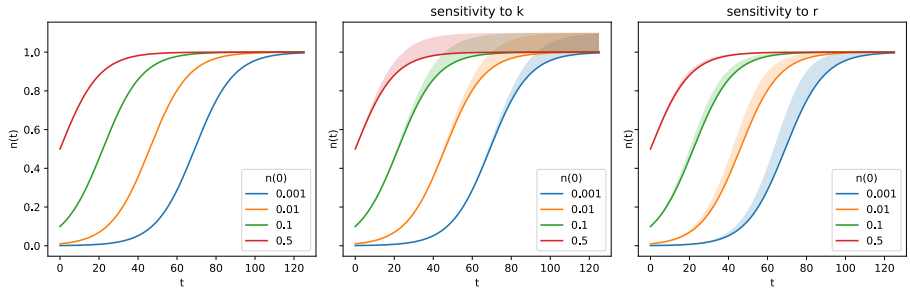


Figure 3.3.: Left: logistic growth of a population for different initial conditions. **Center and right:** sensitivity of this time dynamic to the parameters: the area over the curve represent the sensitivity of that time-point to a change in carrying capacity k (center) or growth-rate r (right).

To optimally choose our two measurement time, we can already gain some intuition from the sensitivities:

- k is better measured when the population size is close to the carrying capacity
- however, r, the exponential rate, must be measured when the absolute population size is large to reduce relative error, and before the carrying capacity is reached, at which point the rate with which the population got there does not really matter.

Assuming additive Gaussian noise on $n(t)$ of standard deviation σ, the Fisher information matrix of $n(t)$ is:

$$I_{n(t)}(k,r) = \frac{1}{\sigma^2}\begin{pmatrix} \left(\frac{\partial n(t)}{\partial k}\right)^2 & \frac{\partial n(t)}{\partial k}\frac{\partial n(t)}{\partial r} \\ \frac{\partial n(t)}{\partial k}\frac{\partial n(t)}{\partial r} & \left(\frac{\partial n(t)}{\partial r}\right)^2 \end{pmatrix}$$

using the fact that Fisher information matrices are additive, the total information from making two measurement at t_1 and t_2 will be: $I_{n(t_1)}(k,r) + I_{n(t_2)}(k,r)$. We allow ourselves to take measurements between 0 and 120 minutes, and choose the D-optimality criterion (see 3.1.2.2.3). We can then use a numerical optimization routine (here CMAES, see 3.2.3) to find the two sampling times that maximize the expected information yield from the experiments, marked by a white star on Figure 3.4.

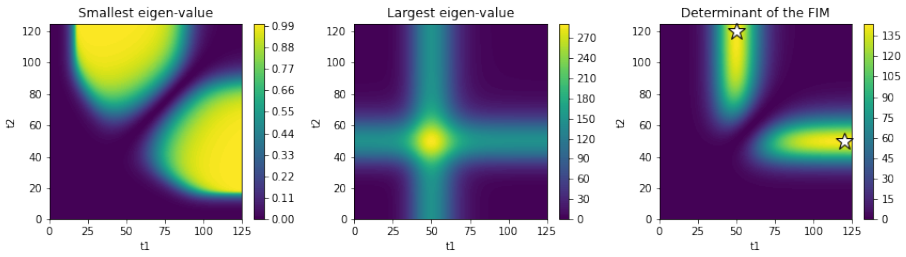


Figure 3.4.: Left and center: Smallest and largest eigenvalue of the FIM, for varying sampling times t_1 and t_2. **Right:** Product of the eigenvalues (also the determinant of the FIM). White stars indicate the (symmetrical) maximum, the corresponding sampling times (50min and 120min) maximize the information (according to the D-criterion)

3.1.3. Non identifiability

I have so far set aside the question of the identifiability of the model, but our growth-rate case study would be a typical example of a system with non-identifiability, provided we made only one time point of measurement. Indeed, even knowing the initial population size, a single measurement does not allow us to discern between a fast growing population with a low carrying capacity, and a slow growing population with a high carrying capacity, as illustrated Figure 3.5.

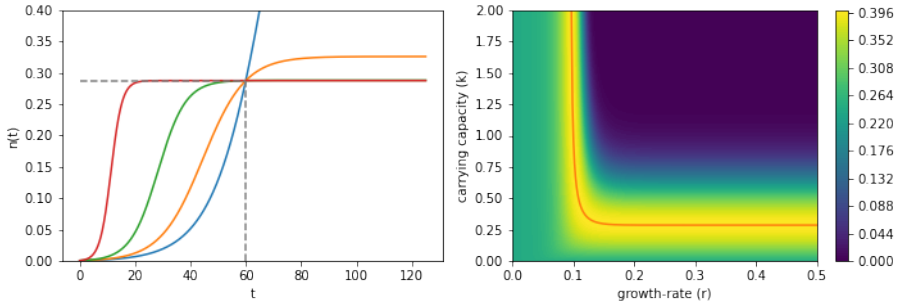


Figure 3.5.: Left: Multiple parameters vectors can give dynamics which cannot be distinguished by only looking at the population size at t=60. Each curve represents a different set of parameters r and k, but all have the same value n^* at t=60. **Right:** 2-D plot of the likelihood function, for the same observation of $n(t = 60) = n^*$. The noise model is multiplicative Gaussian, with $\sigma = 1$. The red curve represent the set of all maximum likelihood estimates.

3.1.3.1. Practical and structural non-identifiability

More generally, unidentifiability can be formally defined as follows.

- Structural non identifiability happens when there exist two different parameter vectors θ_1 and θ_2, such that for all t and u, $\mathcal{D}_{x(t,\theta_1,u(t))} = \mathcal{D}_{x(t,\theta_2,u(t))}$. It is inherent to the model, and the observability of the state, but does not depend on the realization of the noise (hence its other name: *prior* identifiability analysis): even an infintie amount of observations would not differentiate between the two parameter vectors.
- Practical identifiability is more loosely defined. It is the kind of non identifiability that is linked to a given dataset (hence the name *posterior* identifiability analysis), or at least a precise experimental protocol. Existing definitions found in the literature can be:
  - Based on frequentist confidence intervals, or likelihood-based confidence region. In any case the system is considered practically unidentifiable if the interval or region has an infinite (Lebesgue) measure [76].
  - Based on sensitivity analysis, by trying to detect a local continuous non uniqueness of maxima. [85]
  - Gontier et al. [39] propose a definition based on model selec-

tion. Though very original, and with the massive advantage of being independent of data realization, its complexity and reliance on the comparison between two existing models cloud its potential for use.

- Miao et al. [64] suggest an analysis based on Monte-Carlo simulations of the realization of the noise model, each fitted (via a mean left undefined). The average error in parameter estimation would then indicate the non-identifiability of the system.

A key difference between structural and practical non-identifiability, as identified by Chis et al. in [24] is the "capacity to recover identifiability":

> If some parameters turn out not to be structurally identifiable, numerical approaches will not be able to find reliable values for them. In those situations, the only possibilities for a successful model building will be i) to reformulate the model (reducing the number of states and parameters), ii) to fix some parameter values (for example, those which are less relevant to model predictions) or iii) to design new experiments by adding measured quantities (if technically possible). Lack of practical identifiability will be in general terms solvable, providing the experimental constraints allow designing sufficiently rich experiments. In this regard, recent works suggest the use of model based (optimal) experimental design to iteratively improve the quality of parameter estimates.

3.1.3.2. A continuum of recoverability

This leads me to suggest that the cut between practical and structural non identifiability is not as clear as presented in the literature, but that there is rather a continuum of "recoverability" of non identifiability:

1. The most recoverable is non identifiability due to a specific realization of the noise. There can even be one unique maximum likelihood estimate, but a very large or infinite space of parameters whose likelihood is close to the MLE.
2. An intermediate case is the one where there is some apparently structural non-identifiability, but this non identifiability can be

solved by adding measurements at different times, or measuring other state variables.
3. Finally the other extreme is a structural non identifiability that is intrinsic to the model, and cannot be solved even if we have access to the continuous noiseless observation of the full state for all possible inputs.

3.1.3.3. Non-identifiability hampers optimal experimental design

As far as optimal design is concerned, non-identifiability is a massive hindrance. Many methods try to find an experimental plan that would reduce parameter uncertainty to its minimum, and a non identifiability direction will often lead to the informativeness score being dominated by the non identifiability, making it hard to explore the space of possible designs. Moreover, non identifiability will make the FIM non-invertible (singular), making it impossible to compute the criterion mentioned in 3.1.2.2.3. Overcoming these limitations will be a recurring challenge in our work, and is more specifically tackled in section 4.1.

3.1.4. The chicken and egg problem: iterative design

Another issue one may have spotted in the examples above is the chicken and egg problem to which we are constantly confronted in optimal design. When designing an experiment, we always assume that we have some knowledge about the parameters of our model, whether by using a prior in the Bayesian case, or a point estimate in the FIM context. But we would not be trying to design an experimental plan if we knew all there is to know about the parameters. This paradox is inherent to optimal design, and is mitigated by focusing on iterative design. Iterative design is the idea and process where, starting with little information, we design a first experiment. We perform it and gain some, but likely imprecise, knowledge of the system. This knowledge is imprecise because we had little prior information, and our choice of experimental design was thus poorly informed. Yet, we can now use this newly acquired knowledge to design a second experiment that will allow us to learn a more accurate model of our system. Iterating this process, we increasingly refine our learned model, and reach a higher level of accuracy than we would have gotten to if we had randomly designed

the experiments. To my knowledge, there exists no mathematical proof that in the general case, this process will gradually lead us toward the best possible parameters of our system, but some specific cases have been treated theoretically [21] and published practical applications [83] are promising.

3.2. Computational and software considerations

Aside from the mathematical framework, this PhD also presents work on computational aspect enabling the large-scale and automated pipelines presented in 4.2. In the hope of helping all readers fully understand the following chapters, I thus introduce here optimization algorithms, Bayesian inference algorithms, and a bit more in depth automatic differentiation.

3.2.1. Automatic differentiation

Automatic differentiation is maybe the less common topic of those presented in this chapter, and certainly was the most novel I discovered during my PhD. Because my implementation (presented in section 4.3.1) required a solid understanding of it and of the trade-offs between the forward and backward modes, I hereafter present a somewhat deep introduction to automatic differentiation.

Automatic differentiation comes from the idea that it is much easier to differentiate programs than mathematical formulas: programs boil down to a series of base operations (addition, multiplication, etc…) whose effect on the derivative of the quantities are know, for example: $\frac{\partial x*y}{\partial \theta} = \frac{\partial x}{\partial \theta} * y + x * \frac{\partial y}{\partial \theta}$. By replacing each of the operations on variables by an operation on the variables and their derivatives with respect to some quantity, we transform all of the program into something that can operate on both variables and their derivatives. Automatic differentiation can be done in two modes: forward or backward (the latter is also called backpropagation), which can be seen as unrolling the chain-rule of differentiation in one way or the other.

To illustrate automatic differentiation, we will walk through the automatic differentiation of a simple example: a function that compute the Euclidean norm of a two dimensional vector:

```
f(x,y):
  return sqrt(x**2 + y**2)
```

This function can be decomposed in more atomic steps as follows: (I consider here that the automatic differentiation library or compiler is aware of the "square" operation and does not need to decompose it into the product of a variable with itself):

```
f(x,y):
  x_square = x**2
  y_square = y**2
  norm_square = x_square + y_square
  norm = sqrt(norm_square)
  return norm
```

3.2.1.1. Forward mode

We can first differentiate each expression using the chain rule and assuming that the differential of each term of the expression has already been computed. This is the forward mode. For a variable `var`, I will define `d_var = {.x = <the differential of var with respect to x>, .y = <the differential of var with respect to y>}` and access it as `d_var.x` or `d_var.y`.

```
f(x,y):
  x_square = x**2
  d_x_square = {.x = 2*x, .y = 0}
  y_square = y**2
  d_y_square = {.x = 0, .y = 2*y}
  norm_square = x_square + y_square
  d_norm_square = {
    .x = d_x_square.x + d_y_square.x,
    .y = d_x_square.y + d_y_square.y
```

```
  }
  norm = sqrt(norm_square)
  d_norm = {
    .x = 1/(2 * sqrt(norm_square)) * d_norm_square.x,
    .y = 1/(2 * sqrt(norm_square)) * d_norm_square.y
  }
  return norm, d_norm
```

which, after propagating the constants becomes:

```
f(x,y):
  x_square = x**2
  d_x_square = {.x = 2*x, .y = 0}
  y_square = y**2
  d_y_square = {.x = 0, .y = 2*y}
  norm_square = x_square + y_square
  d_norm_square = {.x = d_x_square.x, .y = d_y_square.y }
  norm = sqrt(norm_square)
  d_norm = {
    .x = 1/(2 * sqrt(norm_square)) * d_norm_square.x,
    .y = 1/(2 * sqrt(norm_square)) * d_norm_square.y
  }
  return norm, d_norm
```

Note that here, we have differentiated the function only with respect to its inputs, but the inputs of the program as a whole cannot be known when looking at the individual functions. Hence, we must rather differentiate with respect to an unknown vector of "parameters". To this end, we will now consider that each function does not only have for input the variable as written by the programmer, but also a vector of n derivatives with respect to some unknown quantity. To indicate the values set for such vectors, I'll use the notation `d_var = [.i = <the derivative with respect to the ith parameter>]`. Our function f becomes:

```
f(x, y, d_x, d_y):
  x_square = x**2
  d_x_square = [.i = 2*x*d_x.i]
  y_square = y**2
```

```
d_y_square = [.i = 2*y*d_y.i]
norm_square = x_square + y_square
d_norm_square = [.i = d_x_square.i + d_y_square.i]
norm = sqrt(norm_square)
d_norm = [.i = 1/(2*sqrt(norm_square))*d_norm_square.i]
return norm, d_norm
```

Introduced this way, automatic differentiation in forward mode may seem to be necessarily implemented as a tool taking source code of a program, transforming it, and returning a new source code, augmented with automatic differentiation capabilities. Tough it is the historic way things were done [19], source code transformation tools are often cumbersome to use. Recent advances in programming languages, allowing expressive libraries have led to a more pleasant experience. Automatic differentiation can now be implemented as objects implementing the usual mathematical operations and exposing a similar API to a floating point value, but actually containing the value itself and the derivative vector. Implemented this way forward mode automatic differentiation has a very good optimization potential for the compiler of the host language: operations only have a very local influence on the control flow of the program and the numerical computation optimization machinery of the host compiler can be relied upon and not implemented in the automatic differentiation tool itself.

Overall, forward mode automatic differentiation is thus conceptually simple, and a basic working implementation can be obtained somewhat easily. However, it suffers from a flaw that can be fatal to some use cases: the number of executed assembly instructions scales linearly with the number of parameters with respect to which a derivative is computed. If one is trying to derive a function $f : \mathbb{R}^n \longmapsto \mathbb{R}^m$, (where n is the numbers of parameters with respect to which a derivative is computed, and m the number of output variables whose derivative is computed) forward mode scales with n. Amongst the use cases that are not well suited to forward mode, the most well known is certainly machine learning, and more specifically neural networks. A stereotypical classifier will have thousands of parameters for only one output: it is hence no surprise that research on perceptrons, ancestors of the actual deep neural networks, independently discovered backward mode automatic differentiation, which they called “backpropagation”, that

scales linearly with the number of outputs (m in the formula above).

3.2.1.2. Backward mode

The idea behind backpropagation is that instead of computing the derivative in the same pass as the original program, the latter is executed first, memorizing the values of each variable, and the computation graph is then walked backwards.

To our function:

```
f(x,y):
  x_square = x**2
  y_square = y**2
  norm_square = x_square + y_square
  norm = sqrt(norm_square)
  return norm
```

we can now associate another function: `f_diff(x,y, x_square, y_square, norm_square, norm, d_norm)`. `f_diff` is a function which takes the derivative of the program's output with respect to the output of `f` and returns the derivatives of the program's output with respect to each input of `f`. Its arguments `x` and `y` are the same as in `f`, `x_square`, `y_square`, `norm_square`, and `norm`, are the intermediate values computed in `f` and `d_norm` is a vector of derivatives. However, because we are now doing backpropagation, `d_norm` is now an input to our function representing the derivative of the whole program's output with respect to `f`'s result. Our notation `d_var` must be adapted and its definition changes to become: `d_var = [.i = <the derivative of the i-th program output with respect to var>]`, and `f_diff` is defined as:

```
f_diff(x,y, x_square, y_square, norm_square, norm, d_norm):
  d_norm_sq = [.i = 1/(sqrt(2)*norm)*d_norm.i]
  d_x_square = [.i = d_norm_sq.i]
  d_y_square = [.i = d_norm_sq.i]
  d_x = [.i = 2*x*d_x_square]
  d_y = [.i = 2*y*d_y_square]
  return d_x, d_y
```

Unfortunately, this two passes leads to a harder implementation and optimization. Contrary to the forward mode where transformations of the original source are very local, it now becomes necessary to store intermediate computation, and implement the second backward pass. Often, when this is done in a library, this will require some kind of runtime machinery to stack all performed operations and unstack them when derivatives are needed [100].

3.2.2. Bayesian Inference

Though well and rather simply defined mathematically, Bayesian Inference remains a computational challenge. Indeed, in Bayes formula, $p(\theta|y) = \frac{p(y|\theta)p(\theta)}{\int_\iota p(y|\theta=\iota)p(\theta=\iota)}$, the denominator is often prohibitively expensive to compute because it requires integrating over the whole parameter space. Computer scientists have quickly identified this pitfall and have design methods needing only the numerator of Bayes' formula (more generally, only a function proportional to the posterior, which the numerator is given that the denominator is merely a normalization constant). A prominent family of such algorithm are Monte-Carlo Markov chains (MCMC) methods. MCMC methods all implement a Markov Chain designed in such a way that it has an equilibrium distribution which is equal to the posterior distribution. A state of the art method is Hamiltonian Monte Carlo (HMC) [18], augmented with No U-Turn Sampling (NUTS) [47]. As a testament to the pervasiveness of numerical differentiation, HMC also requires to have access to the derivatives of the likelihood and the prior with respect to the parameter. Section 4.3.5 discusses our attempt at implementing and using Bayesian inference on our model.

3.2.3. Optimization algorithms

As Bayesian inference has its computational tools, so does the Frequentist inference of a maximum likelihood estimate. These tools however, belong to a more general class: numerical optimization methods. In our work, numerical optimization tools are used both for model fitting and for optimizing informativeness criterion. Presenting all numerical optimization methods is beyond the scope of this section, but I nonetheless provide a very brief overview of the field. More details can be found

in [14, 80, 93]. Numerical optimization methods can be characterized by 1) their locality and 2) whether or not they require to have access to the gradient of the target function.

Local optimization routine only attempt to find a local optimum and often require an initial starting point which will be in the "drainage basin" of the final result. Global methods on the other hand attempt to identify a global optimum.

Gradient-based routines are based on the gradient of the function being optimized, which they try to follow to an optimum. Black-box methods on the other hand require no information on the derivative of the function, and rely on the ordering of its values at several point.

Given that no computational method can reliably identify the global optimum, which is often what we are after, randomness is commonly added and the method run several times, to try to maximize the chances of having indeed identified the minimum in question. Randomness can be added either by the method itself, or by running several times the same deterministic method but with a random input (for example a vanilla gradient descent can be iterated with a random starting point).

In this work, the optimization routine I will use the most is the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [43] in its python implementation [44]. CMA-ES is a stochastic, derivative-free, global optimization method, where a population, represented by a multivariate Gaussian is evolved to converge towards a point distribution centered on the minimum. Its initial input, apart from the function to be optimized are an initial mean and a scalar spread indicator. It can also be given bounds to restrict the search space.

4. Methods

This chapter presents the methodological developments I contributed during my PhD. First, a mathematical method to design experiments while embracing the non-invertibility of the FIM in a context of non-identifiability is presented. This method relies on the shift of paradigm introduced in the first chapter: designing for prediction power rather than parameter identification. We note that this method is not usable in practice because of numerical issues. Hence, we introduce the method we used in practice, where designing experiments is done for parameter identification, but the designs are assessed for their predictive power. Finally, the software produced is extensively presented, as well as our foray into Bayesian inference.

4.1. Designing experiments in the face of non-identifiability by designing for prediction power

In this section, I first present how viewing the parameter space through the prism of model behavior can help us tackle the issue of designing experiments while embracing a model non-identifiabilities, and then elaborate on two mathematical methods that I developed in that spirit.

4.1.1. Prediction abstracts away from parametrization

We have discussed in the introduction how data can be used to identify the parameters of a model, and how these parameters can either have some intrinsic physical or biological meaning, or are more simply abstract knobs and cursors to move around in the space of models.

When faced with non identifiability, the first of these endeavors becomes foolish: we will not manage to give a physical meaning to a parameter if this parameter can take infinitely many values. Hence, parameters in our case can be seen as a mere parametrization of an abstract model space in which we we are only looking for a model which will be an efficient prediction tool.

For a given model, the associated parameter space can be equipped with an equivalence relation, relating all parameter vectors that give the same behavior. On the parameter space quotiented out by this equivalence relation, the model is, by definition, identifiable again. We switch from trying to identify parameter values, to identify a given behavior. With this change in point of view in mind, I tried to identify mathematical methods that could allow us to design experiments that maximize not the information learnt on parameter values, but the predictive power of the fitted model.

4.1.2. Irrelevant Component Analysis

In this section, I propose a mathematical method to use the FIM in the face of non identifiability. The idea is that even though some parameter relations may be non identifiable, identifying them may be irrelevant to use the model for prediction, as illustrated in Figures 4.1 and 4.2. Let's hence consider that we have a current estimate θ^* of the real parameter vector. Let's consider a set of inputs on which we will test the model's ability to predict the behavior of the system: $\mathcal{P}red$. Finally let ξ be the experimental plan whose quality we are trying to assess, and $\mathcal{I}$ be its Fisher Information Matrix.

Given that the FIM is symmetric, it is possible to decompose it as $I = BDB^T$, where B is orthonormal, and D is a diagonal matrix whose entries are the eigenvalues of I. The geometric interpretation of this is that in the B basis, the eigenvalues of I indicate how much information we have on each axis. Moreover, if I is invertible, $I^{-1} = BD^{-1}B^T$: the variance on each axis in the B basis will be at least the inverse of the information we have on this axis, per the Cramér-Rao bound. D^{-1} is a diagonal matrix whose i-th entry is the inverse of the i-th entry of D. Hence, abusing notations, we can generalize the fact that $I^{-1} = BD^{-1}B^T$, even to the case where D has null entries (and hence I is non

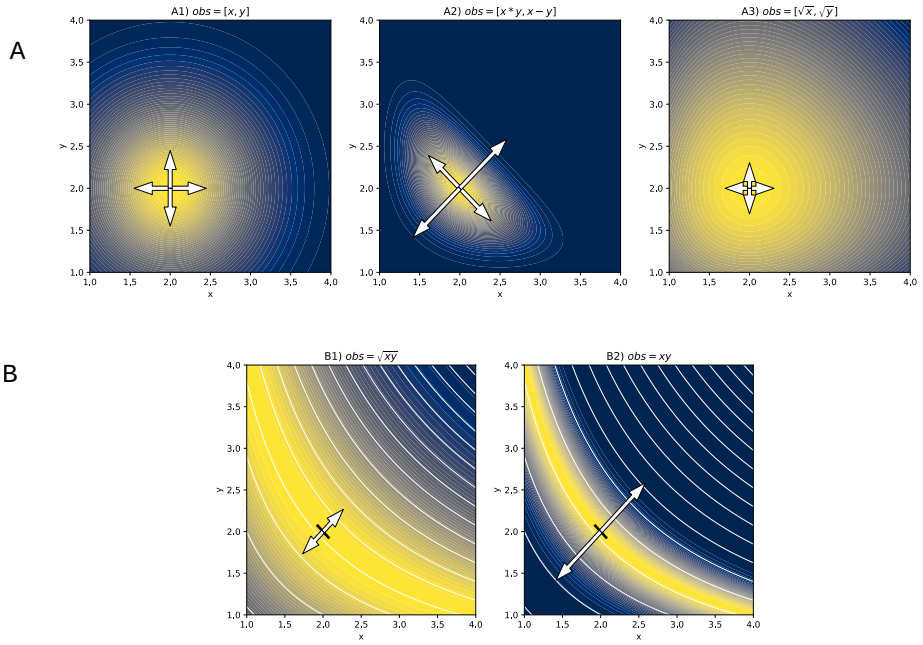


Figure 4.1.: Different observation functions (subplots titles) to identify two parameters x and y, and their respective likelihood functions and Fisher Information Matrix's eigenspace. For each observation, the likelihood (for additive gaussian noise) of the parameters is plotted as a contour function (where yellow is high likelihood and blue low). The likelihood has been computed with respect to a noiseless realization of the observation. The FIM are represented via their eigenvectors. If the corresponding eigenvalue is not zero, the eigen vector is a white arrow, of length proportional to the eigen value. If the eigen value is zero, the eigen vector is represented by a black bar. **Row A** represents fully identifiable systems (all eigen values are non zero), though with different amounts of information. **Row B** represents systems where a direction is not identifiable (only the product, xy is). Even if a direction is not identifiable, we can still intuitively find that **B2** has more information than **B1**.

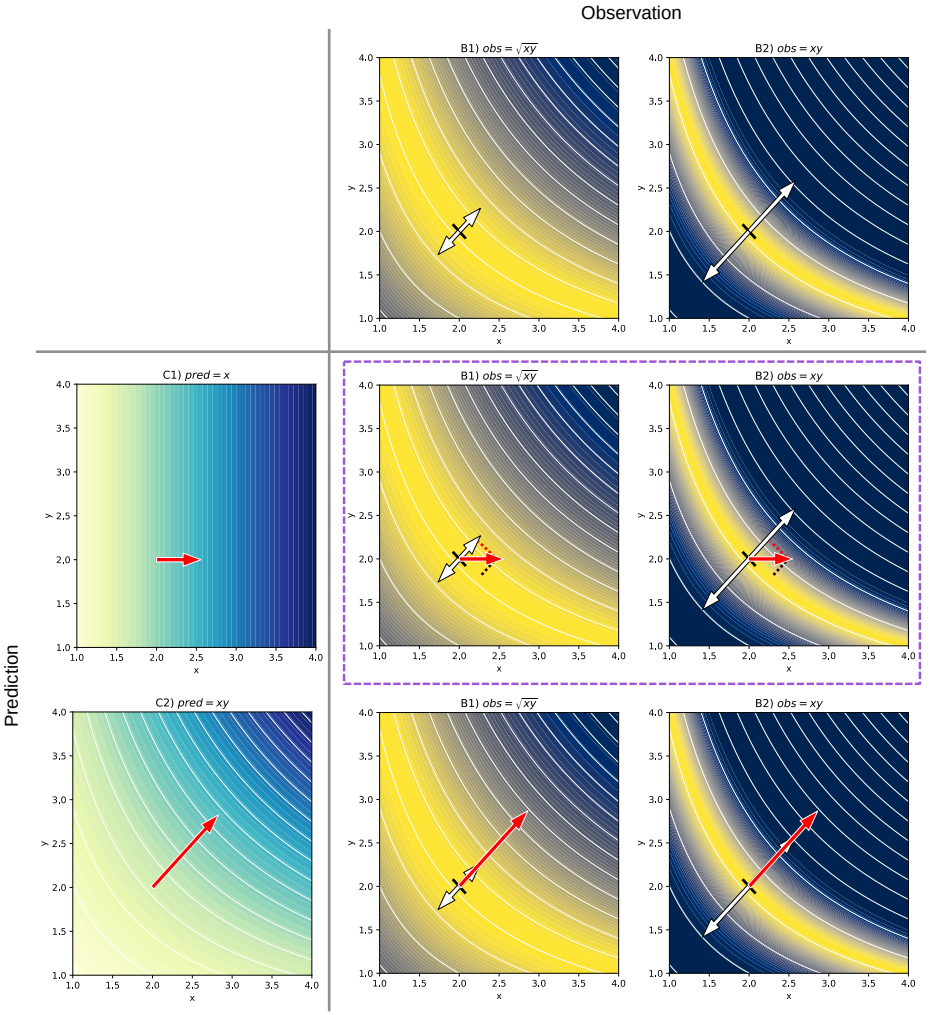


Figure 4.2.: Depending on what we are trying to predict (the rows), we may be fine with the non identifiability of the observations (the columns), and still want to select the most informative experimental plan. The prediction gradient with respect to the parameters is represented by a red arrow, and its projection on the eigen vectors of the FIM by dotted lines, either red (non-zero eigen value) or black. In the purple-dash-boxed figures, the non-identifiability will prevent prediction, because the prediction gradient has a component on a null eigenvector. On the last row, the prediction gradient is colinear to an identifiable axis, and we can thus predict after learning from the two observations. We can also observe that the bottom right observation gives the most information.

invertible), by setting the corresponding entry in D^{-1} to ∞, taking care of never trying to actually carry-out the matrix multiplication $BD^{-1}B^T$. The geometric interpretation of this would be that because we have zero information on some axis, our estimator can take any value on it, hence having infinite variance.

Starting from there, I had the idea to try to determine whether a particular axis had an impact on the dynamics: we consider the testing experiments in $\mathcal{P}red$ which hopefully cover a lot of the possible dynamics, and compute at θ^* (our current estimate) their sensitivity to the parameter values, stacking the sensitivity matrices vertically in a matrix S (S will be a "tall" matrix, having $dim(\theta)$ columns and as many rows as the cumulated number of measurement points in $\mathcal{P}red$). In the B base, this sensitivity matrix is SB.

Using this we can have a heuristic to determine whether a singularity in the FIM is coming from the system itself, in which case it will be there for any experiment (including both our design ξ and our experiments to predict $\mathcal{P}red$), or if the precise experimental design from which the FIM was computed fails to reveal enough information to correctly predict the outcome of one of the testing experiments. To do this:

1. Start with the experimental design ξ
2. Compute its FIM I
3. Diagonalize I in an orthonormal basis B, also finding its eigenvalues λ_1 to λ_n
4. Compute SB where S is the sensitivity matrix of the testing experiments.
5. Learning on ξ should allow to predict the experiments in $\mathcal{P}red$ if for all i so that λ_i is null, the i-th column of SB is also null.
6. If the above condition is satisfied, we can then remove the columns of B and rows and columns of D corresponding to zero eigenvalues, noting $B_>$ and $D_>$ these new matrices, and proceed as if $B_>D_>B_>^T$ was the (invertible) FIM.
7. If wanted, we can also use $SB_>D_>^{-1}B_>^TS^T$ as an estimation of the covariance matrix of our predictions.

Indeed, the i so that λ_i is null are all the directions on which the system is locally non identifiable that make the FIM non invertible.

This method may seem highly involved. Indeed, the classical Cramér-Rao bound tells us that, for an estimator of a function $\phi(\theta)$ of the pa-

rameters θ, the variance will be at least $\frac{\partial \phi}{\partial \theta} I(\theta)^{-1} \left(\frac{\partial \phi}{\partial \theta}\right)^T$. Thus if ϕ was our prediction, we get that the variance of the prediction is at least $SBD^{-1}B^TS^T$, and the method above may seem like a tortuous way to reach a similar conclusion. Except that this "prediction" Cramér-Rao bound still requires the FIM to be invertible. Had the term been $\left(\frac{\partial \phi}{\partial \theta} I(\theta) \left(\frac{\partial \phi}{\partial \theta}\right)^T\right)^{-1}$, our result would indeed be useless, but here, our method circumvent singularity of the FIM in a way that the classical one does not achieve.

4.1.3. Adversarial design: playing against Murphy

In the method discussed above, the set of experiments $\mathcal{P}red$ is only a proxy for "all possible experiments that one could ever want to predict with the model". Strictly speaking, it would only be after ensuring that the non-identifiabilities do not get carried through to the predicted behavior for every possible input that we could elect to ignore them. However, checking every possible input is far from feasible, and hence proxies have to be found. The one presented above is the case where we take a fixed sample of possible inputs, that we hope is representative of prediction power in general. But one could instead decide to identify, for any model, the experiment that this model predicts the worst. Instead of having a fix set of validation experiments, we could use a nested optimization loop to try to find, for each learning experiment, a validation experiment on which the prediction power is the worst.

Going back to the FIM analysis, the steps would be slightly modified. For a given experimental design ξ, we would (changed items highlighted):

1. Compute its FIM I
2. Diagonalize I in an orthonormal basis B, also finding its eigenvalues λ_1 to λ_n
3. Find an input u_{pred} that maximizes the following function:
    1. Compute SB^T where S is the sensitivity matrix of the model's output for input u_{pred}.
    2. Compute the set N of indices i so that the i-th column of SB^T is non-null if and only if $i \in S$.
    3. Consider the set $\{\lambda_i | i \in N\}$. If any of these eigen values is null (or below a certain pre-decided threshold), the non-

identifiability remaining after learning from ξ will prevent the prediction of the system's behavior on u_{pred}. Return $\inf$.

4. Else, remove the columns of B and rows and columns of D corresponding to zero eigenvalues, noting $B_>$ and $D_>$ these new matrices, apply the criterion of your choice to the matrix $SB_>D_>^{-1}B_>^TS^T$ (an estimation of the covariance matrix of predictions on u_{pred}). For example for D-optimality, return its determinant.

4. The u_{pred} that maximizes the function above is the one for which we will predict the worst.

The procedure outlines above associates to each design, its worst prediction "score". By including it in a minimization loop, we can find the design that minimizes its maximal prediction error. This is reminiscent of the minimax algorithm for game engines: during a design step, the experimenter attempts to minimize their possible loss of a worst case scenario during the prediction phase. It is as if the experimenter was playing their turn by finding parameters, for which an hypothetic adversary of theirs would then try to find an experiment that is only poorly predicted.

I implemented and tested this method, but unfortunately step 3 is notoriously prone to numerical error for ill-conditioned matrices, and the rate of misclassified cases was such that research in this direction is currently on hold.

4.2. Designing in spite of non-identifiability: designing for parameter identification and quantifying the resulting prediction power

Having set aside the method presented above, we devise another one, where experiments are designed for parameter identification but their quality is assessed for their predictive power. To introduce our design pipeline, I first present the Bayesian Cramér-Rao bound and discuss how it influenced our use of the FIM in an iterative design context, and then present the pipeline itself. Afterwards, to introduce our assessment

pipeline, I present a general framework in which experimental plans can be assessed and then our particular instantiation of this framework.

4.2.1. Designing for parameter identification

4.2.1.1. The Bayesian Cramér–Rao bound

The van Trees inequality, also known as a Bayesian Cramér-Rao Bound (BCRB) [52, 38] are a sort of hybrid beast between the frequentist Cramér-Rao based design and the full-fledged Bayesian experimental design. They are very similar to the Cramér-Rao bound, except that a prior Π (with pdf π) is placed on the parameter space. With this prior, the Bayesian Cramér-Rao bound is:

As highlighted in [52], a less interesting inequality sometimes also called the Bayesian Cramér-Rao bound can be directly derived from the vanilla Cramér-Rao bound by convexity of $x \mapsto x^{-1}$ on the space of positive-definite matrices:

$$\mathrm{E}\left(\mathit{Var}(\hat{\theta})\right) \geq \mathrm{E}\left(\mathcal{I}(\theta)^{-1}\right)$$

$$\mathrm{E}\left(\mathrm{Var}(\hat{\theta})\right) \geq \left(\mathrm{E}\left(\mathcal{I}(\theta)\right) + \mathcal{I}_{prior}\right)^{-1}$$

where the expectation is taken over the prior Π, and $\mathcal{I}_{prior}$ is a term representing the information contained in Π, quite similar to a Fisher Information Matrix and defined as:

$$\mathcal{I}_{prior} = \mathrm{E}\left[\left(\frac{\partial \log \pi(\theta)}{\partial \theta}\right)^T \left(\frac{\partial \log \pi(\theta)}{\partial \theta}\right)\right]$$

4.2.1.2. Adapting an information criterion from the BCRB

The computational tractability of this expression depends on the nature of the prior. If our prior is defined analytically, as a “nice” distribution (typically a multivariate normal), then $\mathcal{I}_{prior}$ can be computed, and $\mathrm{E}\left(\mathcal{I}(\theta)\right)$ can be integrated by Monte-Carlo methods.

In our pipeline, priors are most likely to come in the form of samples. We are performing iterative design and the prior of step $n+1$ is the posterior of step n. Let’s consider that we have access to samples from actual Bayesian inference (which is not the case, as for the assessment pipeline discussed in 4.2.2). These samples are from a prior which admits a probability density function, but this function’s analytical form is unknown. One could think of using Kernel Density Estimation methods to retrieve a probability density function from the samples, but

KDE comes with a bandwidth parameter, which would actually appear in $\mathcal{I}_{prior}$ for all practical kernel choices (see proof in appendix A, page 103).

Thus, without a way to convincingly give an analytical pdf to Π, we have to give up on computing $\mathcal{I}_{prior}$. Intuitively though, the role of I_{prior} is not entirely clear: the estimator to which variance we are giving a lower bound is a purely frequentist one, with no notion of prior, but the Bayesian Cramér Rao bound somehow links its variance to the information contained in Π. Indeed, previous work [82] proposed a similar formula, except without I_{prior}, which is justified even further by the fact that a term standing for the Fisher Information Matrix of the experimental plans already carried-out can be added to $\mathrm{E}\left(\mathcal{I}(\theta)\right)$. Hence, we settle on using $\mathrm{E}\left(\mathcal{I}(\theta)\right) + I_{prev}$ as our hybrid Fisher Information Matrix, where I_{prev} is the Fisher Information Matrix of previous experimental plans.

4.2.1.3. Using this information criterion in a design loop

To design an experimental plan, I engineered the following pipeline (see Figure 4.3). I have experimental design done iteratively (see 3.1.4), with priors represented by samples in mind. I assume that there exists a parametrization of the design space in $\mathbb{R}^n$, and so by abuse of notation, I will assume that the design ξ is in $\mathbb{R}^n$. I define a score function which, associates associates a pseudo-FIM $\mathrm{E}\left(\mathcal{I}(\theta)\right) + I_{prev}$ to a design, as detailed above. If we name λ_i the eigenvalues of that matrix, we associate a score $\sum \max\left(\log_{10}(\lambda_i), -8\right)$ (the eigenvalue computation is unreliable for small values, hence the clipping at 10^{-8}). Using CMA-ES (see 3.2.3) I can then maximize the score function. Because CMA-ES is a stochastic algorithm, it needs to be run several times to increase the odds of finding an actual global optimum.

4.2.2. Assessing prediction power

4.2.2.1. A generic framework

Except for the virtually impossible Bayesian experimental design, all the methods outlined above embed approximations and/or hypothesis. For example the FIM is a second order approximation, and relies on a point estimate of the true parameters, which in the vast majority of

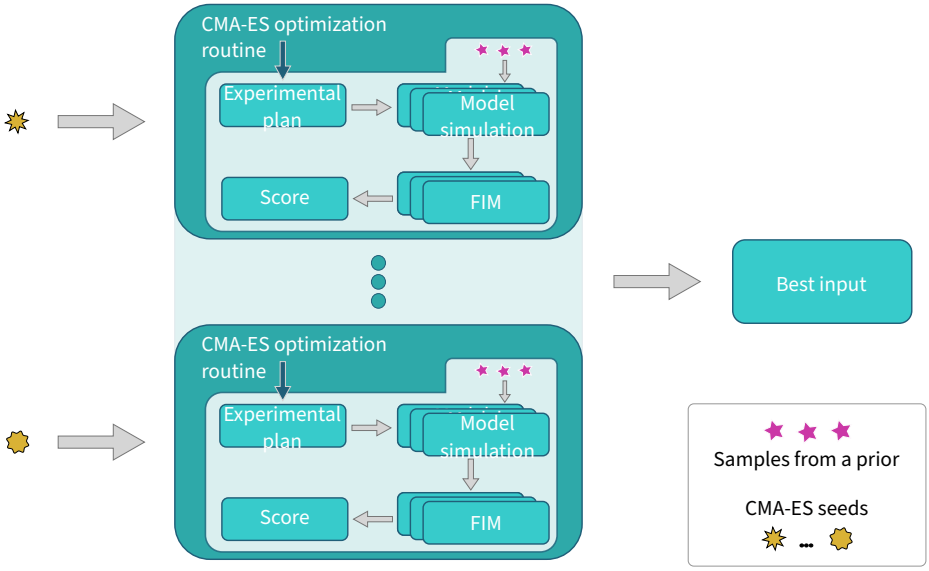


Figure 4.3.: Schematic representation of our design pipeline. Several instances of the CMA-ES optimization routine are launched, each with an different initial seed (dark yellow shapes). Each CMA-ES routine computes an input that (tentatively) maximizes the information, and the best of those is then selected. To compute an information score, the expectation of the FIM on the parameter samples (pink stars) is computed, and its eigenvalues are then used to compute a score (cf main-text).

real use cases won't be exact. Hence, it is necessary to devise a way to assess the quality of a given experimental design.

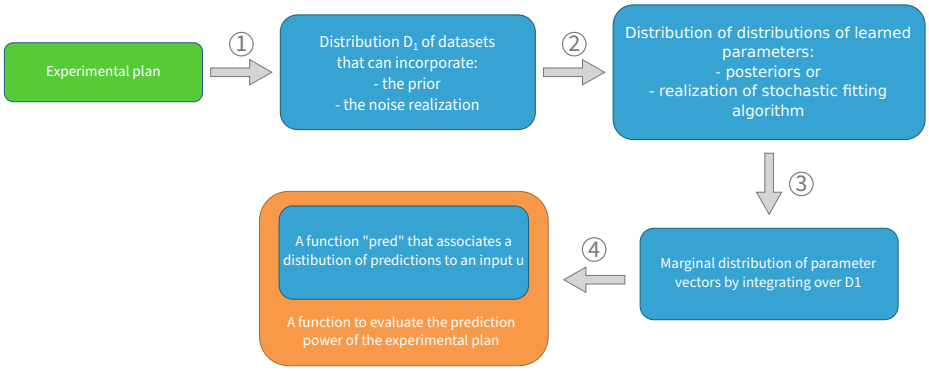


Figure 4.4.: A flow chart representing the general process of assessing the quality of an experimental plan. **1:** The experimental plan is used to simulate datasets. The datasets can vary because we currently have a distribution of estimates of the true parameters of the system, or because of different realizations of the noise stochastic process. **2:** For each of these datasets, we can use a (to-be-defined) learning method to obtain a corresponding distribution of parameters. Because we had a distribution of datasets, we obtain a distribution of distributions of parameters. **3:** Me marginalize the previous distribution to obtain a distribution of parameters. **4:** We now have a distribution of parameters that could be learned from the experimental plan, and we want to assess its prediction power. To each input u, we can associate a distribution of prediction of the system's dynamic corresponding to each parameter of our distribution from step 3. We can then use this function to compute some numerical score of prediction power.

As for the design of the mathematical methods above, we base our reasoning (represented in Figure 4.4) on the importance of prediction power. In all generality, we will see an experimental design as a way to generate some dataset, or rather a distribution of possible datasets, conditioned to a vector of parameters and a noise realization. This distribution defines a distribution of distributions of learned parameters: to each dataset, our fitting method associates a distribution of parameters, whether it is a posterior, a distribution representing the estimates of a stochastic optimization routine trying to find the MLE, or a point-distribution where all the mass is concentrated on a single estimate of MLE. We can take this distribution of distributions and marginalize it into a distribution of parameter vectors (forgoing the correspondence

to a given dataset). Finally this gives us a function "pred" which to any input u associates a distribution of predictions, that is the image of distribution of parameters by the model. This function pred can be used in a metric that evaluates the accuracy of the prediction.

This very general framework implies that certain choices can be made. Amongst them:

- The prior of parameters used to generate the datasets (step 1)
- The noise model used (step 1)
- The method used to obtain parameters from datasets (step 2). This could for example be Bayesian inference, a maximum likelihood estimate, or a distribution of realizations of a stochastic fitting algorithm.
- A function to evaluate the prediction power of the experimental plan (step 4), which requires several specific choices as well:
  - On which input shall we test the accuracy? We can either fix a set of representative inputs as in Irrelevant Component Analysis (see 4.1.2) or find the input on which the prediction is the worst as in Adversarial Design (see 4.1.3).
  - How should the error be measured? We can either compare the predictions to a ground truth, or rather interest ourselves in their diversity, for example by measuring their standard deviation.

Finally, it is remarkable that this general pipeline is very close to the Bayesian experimental design introduced in 3.1.2.1. Bayesian design would be obtained by having D_1 be exactly the marginal on the datasets after integrating the prior and noise realization, marginalizing a Bayesian posterior to get the distribution of parameters, and using a quality metric on this distribution of parameters and not the prediction.

4.2.2.2. Our proposed pipeline

In our case, we instantiate the generic pipeline outlined above as follows and as illustrated in Figure 4.5. The experimental protocol to be assessed is used as input, along with the standard deviation of a multiplicative log-normal noise and a noise seed. For each of these noise-level/noise-seed combination a dataset will be generated. We then distinguish two cases, a Bayesian setting, computationally intractable,

and our actual pipeline. For both sub-cases, a fixed set of 220 inputs to be predicted will be used, and often called the validation set of inputs. To each of these inputs, we can associate a true behavior by simulating it with the same parameters used to obtain the dataset.

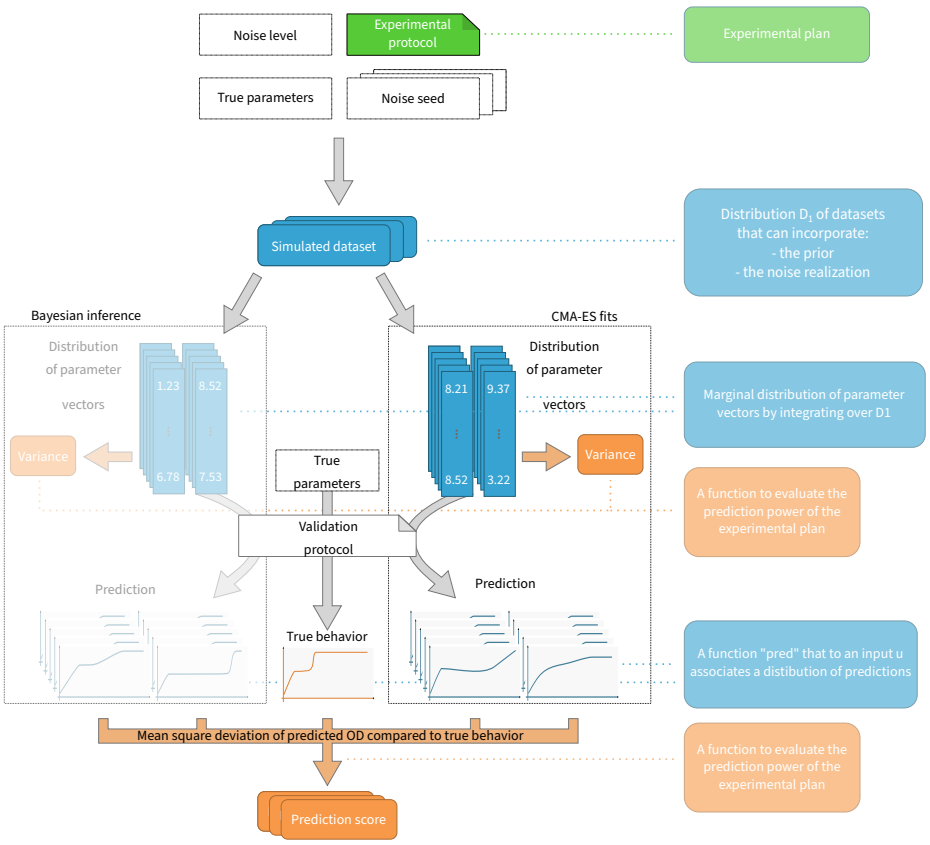


Figure 4.5.: Our pipeline to assess experimental protocols. All orange boxes can be taken as a measure of quality; but we focus on the bottom one. The left black dotted box represents the “Bayesian way” (see main text) while the right one represent the one we actually use. Because we finally do not use the Bayesian way, it is represented as faded. Transparent items on the right refer to Figure 4.4.

In the Bayesian sub-case, Bayesian inference is performed on the dataset, which gives samples from the posterior distribution. Then, for every pair of samples parameter vector and input (from the 220 inputs of the validation set), we simulate the behavior of the system, and compare it using mean-square deviation to the true behavior of the observable quantities. Unfortunately our model does not lend itself well to computational Bayesian inference (see 3.2.2). We thus replace it with an inference step, using multiple CMA-ES (see 3.2.3) runs, all

initialized with a different seed. The set of parameter vectors obtained is then used as would be a set of samples from the posterior.

Regardless of which way (Bayesian, or CMA-ES) we perform, we finally obtain, for each experimental protocol and each validation experiment, a set of prediction error (a scalar number) coming from parameter samples , which we ultimately want to use to compare designs to one another (see Figure 4.6). Let's consider two experimental designs ξ_a and ξ_b and their respective set of prediction error (for a given validation experiment), S_A and S_B. Let's define A (resp. B) to be random variates, taking uniformly distributed values out of the set S_A (resp. S_B). A first idea would be to consider $\mathcal{P}\left(A \leq B\right)$, the probability that learning on ξ_a gives a better prediction than learning on ξ_b. This idea makes a lot of sense intuitively, and I first tried it out, but it suffers from a major issue: even if two parameter set are identified by the human operator as giving the same prediction as the ground-truth, tiny and negligible differences in the numerically integrated behavior are contributing to the prediction error, and will thus bias the aforementioned probability (see 4.8).

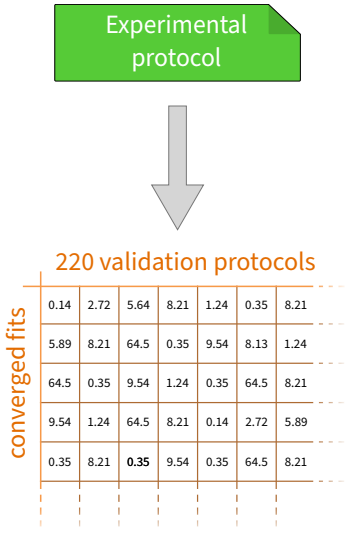


Figure 4.6.: The predictions scores obtained for one evaluated protocol.

Instead, we should rather identify, for each comparison between a realization a of A and b of B, which of the following proposition is true:

a. Both predictions have a negligible error (they are below the blue line in Figure 4.8), gray area in Figure 4.9.
b. At least one prediction is making a non-negligible error, and ξ_a is making a smaller error, red area in Figure 4.9.
c. At least one prediction is making a non-negligible error, and ξ_b is making a smaller error, green area in Figure 4.9.

This allow us to propose, to compare two experimental designs, the visualization presented Figure 4.10, that I name "cave plots", because of its resemblance to a cave system with its stalagmites and stalactites.

It is worth noting that care should be taken when implementing the comparison procedure. If we consider that for each design, we have n prediction error samples, the naive implementations would be in $O(n^2)$. However, it is possible to implement it in $O(n\log(n))$, as illustrated in Algorithm 1 in appendix B, page 107. It should also be noted that this relation, which identify when a random variate will be greater than another one with probability greater than 0.5 is not transitive, and should

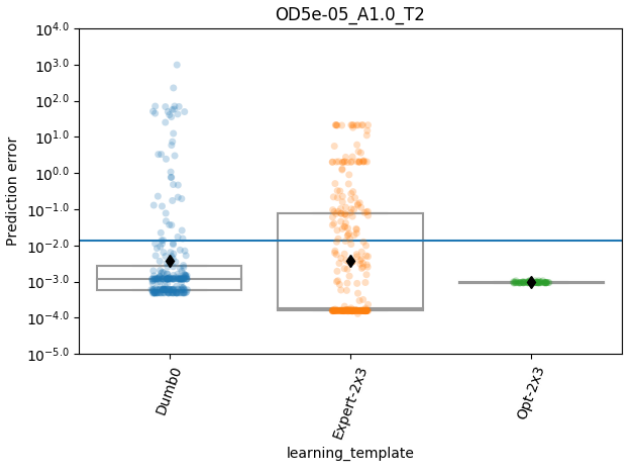


Figure 4.8.: Prediction error of fifty fits on three different designs ("learning_template"), named "Dumb0", "Expert-2x3" and "Opt-2x3", for *one* validation experiment. Each point is a fit, done on the design corresponding to its x-axis position, with a prediction error on the validation experiment equal to its y-axis position. The horizontal blue line indicates the value below which error is not visible to the human operator: all points below this line can be considered equivalent. If all points are considered, the "Opt-2x3" design has a worth distribution of prediction error than "Expert-2x3", whereas if we only consider all point below the blue line to be equal, we reach the opposite conclusion that "Expert-2x3" is worse than "Opt-2x3".

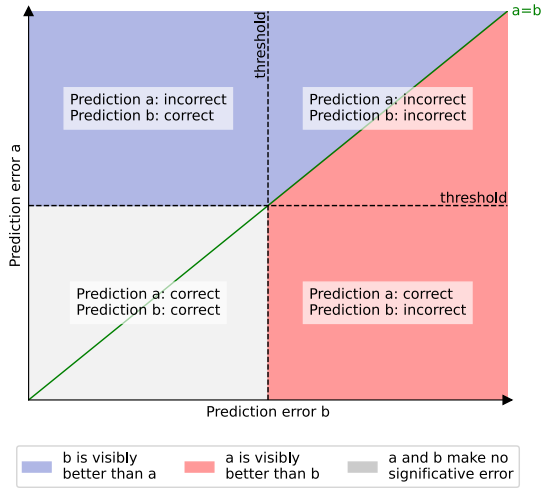


Figure 4.9.: An illustration of the three cases we distinguish. The x-axis represents the prediction error of b, and the y-axis the prediction error of a. The dashed black lines represent the threshold below which the error is neglected (the blue line in Figure 4.8). They delimit four regions, described in the four semi-transparent white background, text boxes. The green first diagonal represents the line where both a and b have the same prediction error. Above this line, a leads to a worse prediction, while below this line b leads to a worse prediction. The red and blue areas illustrate that only the non-negligible differences are taken into account.

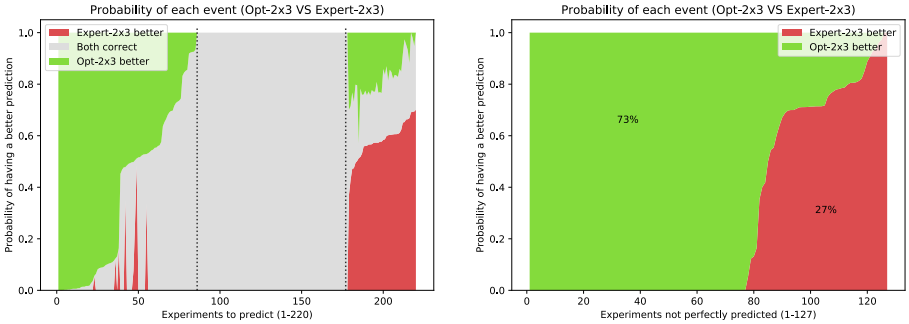


Figure 4.10.: Cave plot for two designs, “Expert-2x3” and “Opt-2x3”, probabilities of three events: **Expert-2x3 better:** at least one prediction is making a non-negligible error, and “Expert-2x3” is making a smaller error, **Both correct:** both predictions have a negligible error, and **Opt-2x3 better:** at least one prediction is making a non-negligible error, and “Opt-2x3” is making a smaller error. The x-axis is the 220 validation experiments equally spread apart, the y axis are probabilities, and the curves are stacked onto each other so that they sum to one. Dark gray dashed vertical lines delimit the region of validation experiments that are perfectly predicted by both designs with probability one. The sub-figure on the right represents the same probabilities but knowing that not both are correctly predicted (i.e.. “removing” the gray from the left panel).

not be considered a proper order relation. This paradox is known as the non-transitive dice paradox [86].

4.3. Software implementation

Because we wanted to be able to test many different models experimental designs, and to extensively assess them, we needed software able to deliver high performances at large scale. Several development were needed, which are detailed below. A graphical summary is provided Figure 4.13}.

4.3.1. Fwd:AD (automatic differentiation in Rust)

Fwd:AD implements **automatic differentiation** (with forward propagation). Given the trade-offs exposed in 3.2.1, I choose to rather go for automatic differentiation in forward mode, to benefit from the existing compiler automation, have no runtime, and because our n and m are of comparable magnitude.

Implementation-wise, Fwd:AD aims for efficiency by:

1. Never copying memory in its functions, leaving it up to the user to be explicit as to when copying should happen. An option is provided to trade-off a bit of performance guarantee for ease of use by instead having Fwd:AD copy memory implicitly. Whether the implicit copy option is turned on or off, all the checks and decisions to copy are made at compile time thanks to Rust's powerful type system.
2. Allowing the user to store all values on the stack if the number of variables with respect to which derivatives are needed is known in advance, thus preventing memory allocation.

It is distributed on *crates.io*, the Rust library repository (similar to PyPI) at the following address: https://crates.io/crates/fwd_ad.

4.3.2. cvode-wrap

Integrating ordinary differential equations and their sensitivities is conceptually simple. If we have a system of differential equations on n

state variables, and we want the sensitivities to p parameter variables, we have np equations. We can integrate these np equations as if they had no relations to each other, and were a differential equation system with $n(p+1)$ state variables. However, these equations are actually related, and these relations can be leveraged to gain better performances. If the numerical integration scheme requires the Jacobian of the right hand side (as is often the case for schemes used for stiff systems), then it can be greatly optimized using the fact that the initial ODE system has the **same Jacobian** as each of the p sensitivity systems. Moreover using a solver that clearly identifies sensitivities as such and not as "np other state variables" allows for finer grained error control and step-size choice method. More information on this subject can be found in [45].

I thus was looking for an ODE solver with the following characteristics:

- high-performance (preliminary code-bases were using scipy [98] solvers and a pre-compiled Rust routine for the right-hand side but were too slow)
- sensitivities as first-class citizens
- Rust interface or C interface (to be able to call through Rust's foreign function interface)

I hence settled my choice on cvode(s) from the sundials suite [46]. A good review of existing solutions is available in [74]. The remaining challenge was to make its functionality available from Rust.

The Rust language emphasizes the memory-safety of programs (safeguarding from out-of-bounds accesses, use after-free, and null-pointer dereference), which C does not provide, hence there are usually two steps to make functionalities from a C library available to Rust programmers. The first one is to simply make the C symbols (the function names) available to Rust. This work had already been done for sundials but the crate (Rust library) was unmaintained. I contacted the maintainer and took over, updating it and getting it to work with the latest release of sundials. The second step – called "wrapping" and more complex – is making the C functionality available to Rust programmers via an interface that guarantees the memory safety. I have designed, written, and published a set of wrappers[22], available on the crates.io archive.

4.3.3. Odegen: efficient ODE solving

Odegen is a Rust macro allowing one to specify an ODE model of a system in a JSON like format. This model is then compiled to efficient code to integrate the ODE (using the CVodeS solver from the sundials suite) and its sensitivities to the parameters (using Fwd:AD), and C bindings are exported allowing one to call this efficient code from virtually any language through foreign function interface (in our case, it is called from Python). A code example can be found Figure 4.11.

```
model! {
    nvars = 2;
    nparams = 4;
    ninputs = 0;
    nobservables = 0;
    state = x, y;
    parameters = alpha, beta, gamma, delta;
    inputs = ;
    observables = ;
    definitions = {};
    equations = {
        Return {
            x : alpha * x - beta * x * y,
            y : delta * x * y - gamma * y,
        }
    };
    observables_equations = {
        no_observables!()
    };
}
```

Figure 4.11.: Example of an JSON-like Lokta-Voltera prey-predator model that can be used as input for odegen.

4.3.4. Adoi: the user-friendly interface to efficient code

Adoi is a Python library implementing a user-friendly interface to call code generated by odegen. It includes:

- An interface to define changing inputs to the system (for example light-signal to an opto-genetic system) or instantaneous actions (injection of chemicals, dilution…), adapting the state variables and restarting the integration as needed,

- Plotting helpers to study the result of an ODE integration.

4.3.5. Bayesian inference on fallible functions

During my PhD, I spent a substantial amount of time trying to perform Bayesian inference on the model, with unfortunately little success. The first attempt consisted in trying to use Stan-MC [92], with which I had prior experience. Stan is a standalone language, dedicated to Bayesian inference, using HMC+NUTS (see 3.2.2). It comes with its own automatic differentiation library. Unfortunately, numerical differential equation solving remains an underdeveloped part of Stan. In fact, numerical differentiation of differential equation solvers remains a hard problem to solve generally, and a blind-spot of many Bayesian inference tools: when ODE solving is offered, it is often in a simple form, with poor user interfacing, making it difficult to well use with our general experiment description representation.

The second attempt was to use a less integrated solution: instead of looking for a tool where I would only need to specify the model, I would look for a tool which would rely on its user to compute the likelihood and its derivative. After all, I already had code to compute a model's behavior and its derivatives. I thus choose to interface with Theano, an academic Python machine learning library, which included an implementation of HMC+NUTS. Once I had a working interface between Theano and my model simulation code adoi, I launched the HMC routine and… realized that I was still far from done. The throughput (measured in posterior samples per minute) was terribly low, and the routine would often end up crashing, with abstruse errors originating from the deep internal machinery. I tested with another HMC code I found online with similar results. My current understanding of this issue is that HMC+NUTS is not well suited to perform inference on a model which numerical ODE solvers will often fail to integrate, as was the case for ours. The difficulty to solve a system of ODE numerically is something quite subjective, or at least implementation dependant. Indeed, across the three years of my PhD, and numerous iteration of the model's code, I have gradually made its implementation more kind to ODE solving routines, sometimes with tremendous impact on performances. However, I had to take the decision to not pursue further Bayesian inference attempts, and consider as sunk cost the time I had

invested in it.

Basic MCMC algorithms were also investigated, but while they ran without crashes, the quality of their samples was too poor to be usable, as expected in a high dimensional space, with a complex-shape prior.

4.3.6. Captain: an ad-hoc workflow manager

Large pipelines such as our, where many small tasks depend on each other, and need to be run efficiently, and only when needed, are cumbersome to manage by hand. Fortunately, there exist programs that are entirely devoted to this task: make [56], snakemake [65], nextflow [25], Apache Airflow, Spotify's Luigi, etc… Each of these programs has its own way of defining the tasks and their dependencies, and their specific options and capabilities. Following common practice in the department, I initially started using snakemake, which is focused on scientific computing workflows, and natively support the job submission daemon of our cluster: SLURM [50]. Unfortunately, as my work was advancing and the task graph was growing, snakemake started scaling poorly. In the end, the time to first job submission on the cluster was more than fifteen minutes, and the RAM used in the graph building preliminary phase was more than 50GB, requiring out of the ordinary resources allocation.

To resolve what had become a major bottleneck, I had three possibilities: debug what is likely to be a memory leak in the snakemake codebase and fill an issue with the project maintainers, switch to a new workflow manager, or implement my own ad-hoc solution. My command of advanced Python debugging and profiling was not enough to efficiently identify the potential culprit in snakemake's codebase, and switching to a new workflow manager would have meant taking the risk of learning how to use it only to realise that a capital feature was missing, or that performance was lacking. Hence, I opted for the third solution and implemented my own workflow manager: Captain.

Captain is a highly-concurrent workflow manager, providing three backends: dry-run, locally parallelized, and job submission to a slurm daemon. It it written with performance in mind, and contrary to snakemake, its time to first job submission is on the order of a second for my largest work graphs, with memory in the order of hundreds of MB.

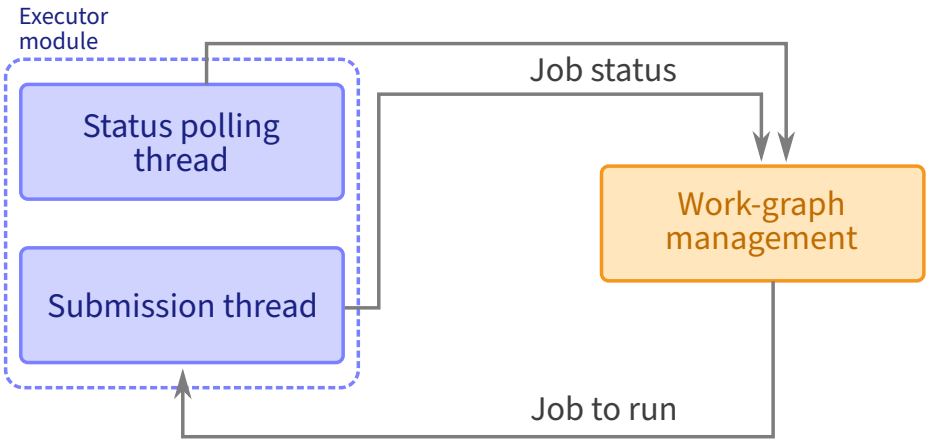


Figure 4.12.: A simplified representation of Captain's structure.

It is structured as a main module, managing the work-graph, which communicates with executors. Different executors allow to schedule jobs on different platforms (for now slurm, locally, and as a dry-run), and I will from now on focus on the slurm executor. slurm does not (readily) offer a way to receive a message (for example on a Unix socket) once a job is done running. Hence, the slurm executor module is composed of two threads: the submission thread and the status polling thread. The status polling thread regularly queries the state of all the currently submitted jobs, detects those that are finished, and signals them to the main work-graph management thread. The Submission thread receives jobs to launch on the cluster and submit them to the Slurm engine. The main thread receives messages signaling that jobs are finished, and sends the jobs that depended on them to be launched. It makes use of `crossbeam` a Rust message passing library, but also of concurrent atomics when possible, to guarantee strong performances.

In all rigorousness, there are actually several submission threads, to compensate for the lag of slurm's blocking interface.

4.4. Consideration on the use of a cluster

Institut Pasteur hosts a computing cluster. Following its renewal in 2020, all nodes now have the following configuration: 2 AMD EPYC 7552 (Rome) CPUS (96 cores, 1 thread per core), 512GB of RAM. The team has to its disposal 192 cores we own, 3648 which are shared, and 6912 which are owned by other teams on which we have very low priority access. This massive computing power was instrumental in helping me carry out the in silico study presented afterwards. However, using

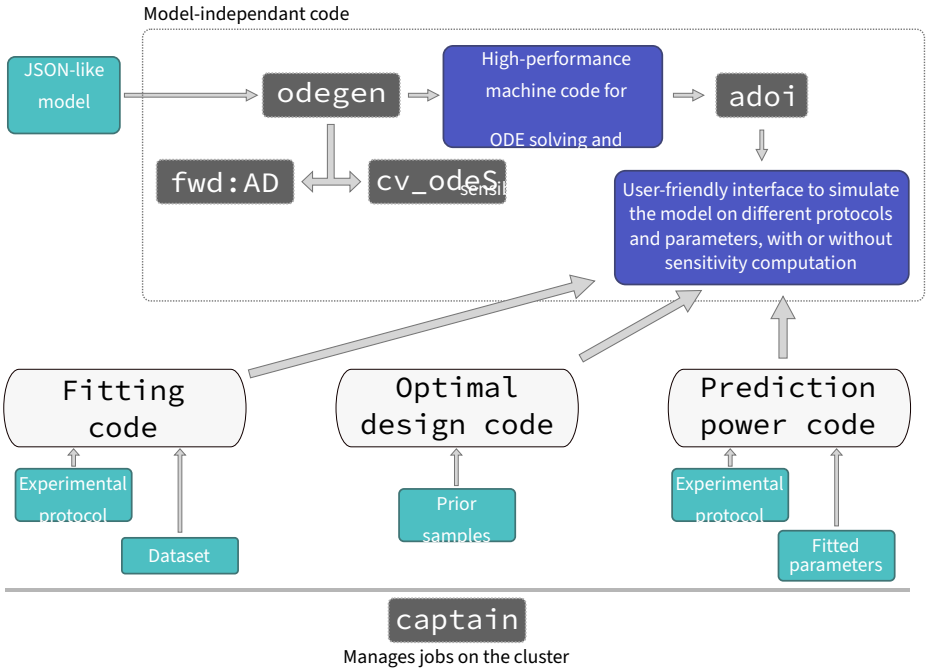


Figure 4.13.: General overview of the different pieces of software and their interactions. **Top:** Four libraries interact to transform a JSON-like description of an ODE model into code that offers both high performances and a user friendly Python interface to simulate the model on different inputs and parameters, with or without sensitivity computations. **Bottom:** Captain orchestrates on the cluster thousands of jobs performing either model fitting, optimal design or assessment of design. All these scripts rely on the user-friendly interface to the model.

the cluster also imposed many software engineering constraints and entailed bugs such as:

1. The cluster uses a shared file-system (NFS) where performances degrade dreadfully when as few as a thousands files are present in a given directory. One must hence pay attention to the way the file hierarchy is organized, especially when using tools such as make or snakemake, which relies on the directory structure to identify the task that must be run, making it cumbersome to change.
2. Not all libraries have been designed with large-scale parallelism in mind. For example the Python implementation of CMA-ES (see 3.2.3) tries by default to log into a file, whose name doesn't change across runs. Hence, thousands of processes ended up competing for write-access to the same file on the networked file-system, with massive impact on performances.
3. Running very large numbers of jobs means that low-probability events will probably occur. An example of that is the arbitrary termination of a job while its result file is being written to, leading to its corruption. Defending against this is done by making all write-operations on persistent files atomic, using the appropriate library in each language.
4. When the Rust code implementing the model has changed, it is automatically recompiled before being used. This can happen at any place in the pipeline where the Rust model is loaded in the python code. Cargo, the Rust package-manager ensures that a given source code is not being compiled by two different process in parallel by using file locks, which are unfortunately bogus on some NFS file-system, such as the cluster's one. Hence, some processes would randomly crash because two of them were trying to recompile simultaneously code that had been updated.
5. Sharing a cluster with other users implies that resources are scarce. The cluster's job submission engine, SLURM, implements a policy where short-duration jobs are given a priority advantage. On top of giving it a higher priority, having a short-lived job reduces the risk that it is stopped by the scheduler to run a higher priority one. Given that most of the long run jobs are iteration of an algorithm, it is highly interesting to sub-divide the iterations into smaller units, which will run for less time. This

implies that intermediate states must be saved to and loaded from disk, adding yet another level of complexity and multiplying the number of jobs in the whole pipeline.

Given the technical difficulties that come with using a cluster, one may wonder whether it is actually needed to use one. As of the final version of our design and assessment pipeline, the cumulated run times were as indicated in table 4.1, making for an overall duration-CPU of more that sixteen years.

Table 4.1.: Several statistics on the duration of the design optimization loop and the fitting loop of the assessment pipeline.

	Design	Assess (fit)
Number of jobs run	9 057	705 114
Mean duration	4h 46m	8m 52s
Total duration	4.9 years	11.9 years

5. Can optimal design help with the characterization of a model of enzyme-mediated escape of antibiotic treatment?

5.1. A model of enzyme-mediated escape of antibiotic treatment

5.1.1. β-lactam antibiotics

β-lactam antibiotics – amongst which penicillin was the first discovered – are a class of antibiotics widely used to fight Gram-negative and Gram-positive pathogens. In 2003 they had a share of 65% of the worldwide market for antibiotics [30] and in 2019, 64.5% of antibiotics consumed in French health establishments were β-lactams [88]. The most well known to the public are amoxicillin (sometimes combined with clavulanic acid as in Augmentin) and Cefpodoxime. Having been exposed to β-lactam antibiotics for the last 80 years, many pathogens have developed resistance to those, notably through the production of enzymes that hydrolyze the antibiotic: β-lactamases. In Gram-negative bacteria these enzymes are stored in the periplasmic space where they degrade the antibiotic that passes the outer membrane, and make the individual cell *resistant* to β-lactam antibiotics. As depicted in Figure 5.1, *resilience* happens because bacteria release β-lactamases when their cell wall is broken by the β-lactam antibiotic. Once released into the environment, the enzyme starts hydrolyzing the antibiotic, hence

reducing the peril to which the remaining bacteria are exposed. For more detail on resistance and resilience see [63].

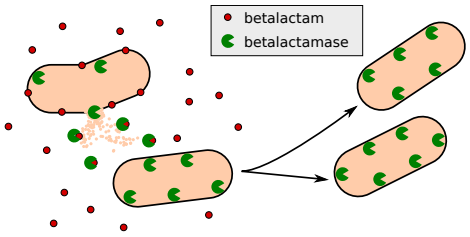


Figure 5.1.: Collective antibiotic tolerance in bacteria producing β-lactamases. The top left cell dies and releases β-lactamases which depollute the environment and allow the bottom left cell to duplicate unharmed. Figure by V. Andreani. [3]

Resistance and resilience combined lead to a complex behavior, where an initial dose of antibiotic can lead to a sharp decrease in the number of viable cells, which in turns implies that a lot of β-lactamases has been released, and that the environment is being actively "cleaned" (from a pathogen's point of view), allowing for a later rebirth of the bacterial population. Current characterization of bacterial strains as either sensitive (S), intermediate (I) or resistant (R) is based on the Minimal Inhibitory Concentration (MIC), the minimal concentration of antibiotic that prevents growth after a given time. This classifications is too coarse to help clinicians fully understand the resistance and resilience capacity of the infection they are fighting, as represented in Figure 5.2A.

5.1.2. How a fellow PhD student modeled antibiotic resistance

The work presented in this section has been done by V. Andreani during his PhD in the InBio team. Presentation of original work resumes in 5.2.

5.1.2.1. Overview

A new characterization technique beyond the SIR classes for enzyme mediated resistance to antibiotic is precisely what Virgile Andreani – a senior PhD student of our team – designed during his PhD [3, 4]. This method, represented Figure 5.2B, consists of a bench-to-silico pipeline where experiments are performed using an automated platform he developed and a model is fitted to the resulting dataset, allowing us to obtain parameters and uncertainties on the fits. These parameter

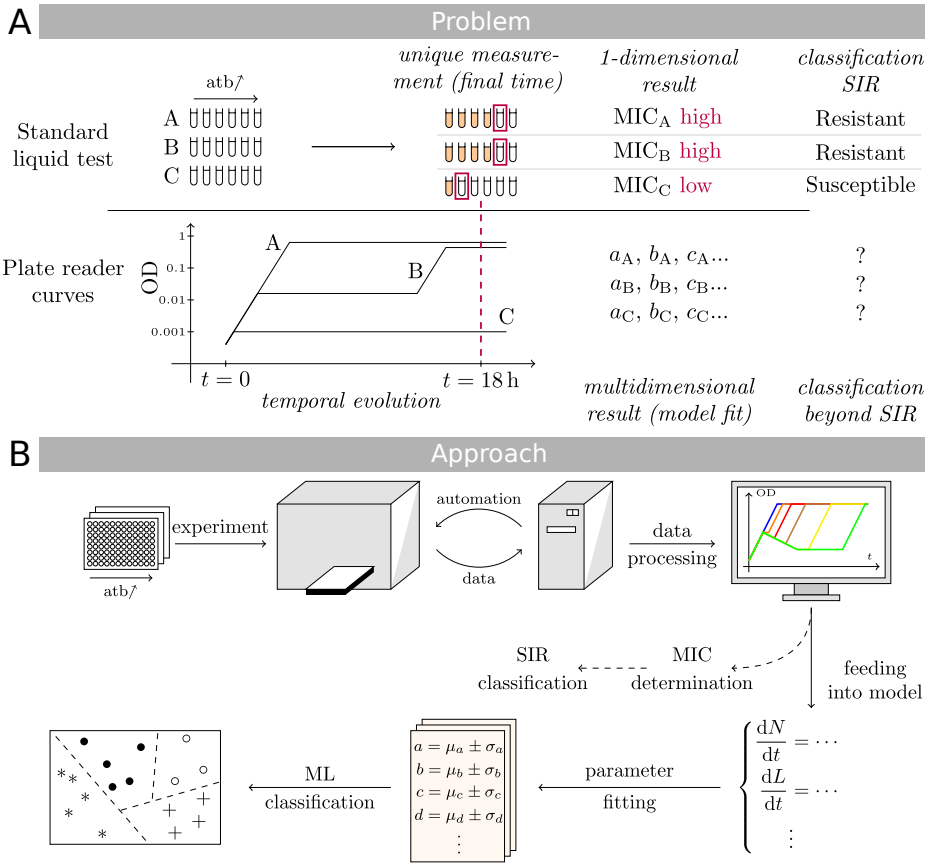


Figure 5.2.: Introduction Figure from [4], representing the problem and the authors' approach to solve it. **A, Problem** Top: Minimal inhibitory concentrations (MICs) are measured with a unique time point at 18 hours. Isolates A and B have a high MIC, whereas isolate C has low MIC. Bottom: Following the optical density (OD) of the cultures during the experiment allows one to unveil that strains A and B behave differently despite having the same MIC. One can qualify A as highly resistant, and B as poorly resistant but tolerant and resilient. Strain C is neither resistant nor tolerant. **B.** Model calibration approach. A 96-well plate is inoculated with the isolates to be tested, and a range of antibiotic concentrations is applied. The OD is read by an automated plate reader and processed automatically. The data is then fed into the model, for which numerical parameters are estimated, along with their uncertainties, for each isolate. On the basis of these parameters, a machine learning approach can be used to classify the strains into clusters.

uncertainties are then used to classify the strains in three categories: sensitive, tolerant/resilient and resistant.

This work was set in an heavily automated environment, with many computer-to-apparatus interactions. Given that the data collection and analysis already benefited of much computational help, it made sense to try to automate the experimental plan as well, potentially with interleaved phases of learning from the already-performed experiments and designing the next experimental plans. Moreover, given the key-role played by both parameters values and their uncertainty in Virgile's classification, offering an *optimal* characterization of the antibiotic resistance and ensuring that uncertainties in parameters came from the biological system itself and not a poor choice of observations could prove decisive to the usability of the pipeline. I thus set out to try to apply experimental design methods to Virgile's model, as a real-world case-study, where the optimal design would be both highly beneficial, and challenging.

5.1.2.2. The mathematical model

Virgile's model is represented in Figure 5.3, and incorporates the following processes. Without antibiotic, *e. coli* individuals reproduce by replicating their DNA, elongating to twice their size, moving genetic material to both ends of their now elongated self, and dividing through a ring-like structure at its middle that gradually tightens. β-lactam antibiotics work by binding to proteins called Penicillin Binding Proteins (PBP) involved in building the cell wall during the elongation and division. More specifically, the model focuses on the action of the antibiotic on PBP1 and PBP3. PBP3 inhibition prevents the cell from dividing, leading it to elongate way past its nominal size. These abnormally long cells are more fragile, and once they reach a critical size, their cell wall will start to break and they will die in an exponential process. When the antibiotic doses are high enough to also inhibit PBP1, the critical length at which the death process starts is reduced, leading to an earlier cell death. Each time a cell dies, it releases an amount of the resistance enzyme β-lactamase proportional to its length into the environment.

Though it is not needed to fully grasp the model to understand what follows, its mathematical definition is included for reference:

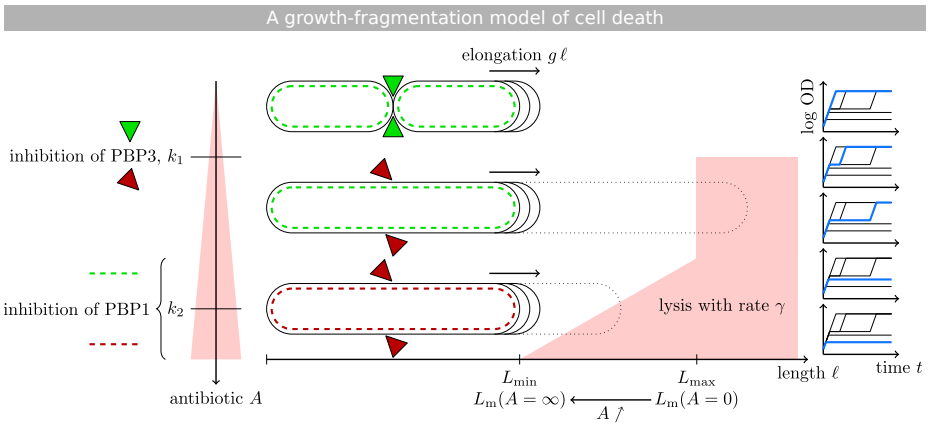


Figure 5.3.: Model representation figure from [4]: Cells elongate exponentially at a rate g that depends on nutrients but is not affected by the antibiotic. Above a concentration k_1, the antibiotic inhibits PBP3, which deprives the cells of their capacity to divide. Cells that cannot divide filament until they reach a critical length $L_{\max}$ from where they experience a cell death with rate γ. Higher concentrations of antibiotics, above k_2, inhibit the PBP1s which weakens the cells and reduces the critical length, such that cells die earlier. On the right are pictured schematic representations of the typical response of the cell culture to an initial antibiotic treatment.

$$\frac{\mathrm{d}N}{\mathrm{d}t} = N\left[f\left(\frac{L}{\ln 2} - 1\right) - \gamma\, Y_{>}\right]$$
$$\frac{\mathrm{d}L}{\mathrm{d}t} = L\left[g - f\left(\frac{L}{\ln 2} - 1\right)\right] - \gamma(L_{>} - L\, Y_{>})$$

$$\frac{\mathrm{d}s}{\mathrm{d}t} = -\frac{g}{\lambda}\, N\, L \qquad g = \mu\frac{s}{K_s + s} \qquad \nu = \frac{\beta}{g}$$
$$\frac{\mathrm{d}a}{\mathrm{d}t} = -k_{\mathrm{b}}\, b\, a - d_{\mathrm{a}}\, a \qquad f = \frac{\beta}{1 + \left(\frac{a}{k_1}\right)^{h_1}}$$
$$\frac{\mathrm{d}b}{\mathrm{d}t} = \gamma\, B_{\mathrm{in}}\, N\, L_{>} - d_{\mathrm{b}}\, b \qquad L_{\mathrm{m}} = L_{\min} + \frac{L_{\max} - L_{\min}}{1 + \frac{a}{k_2}}$$
$$\frac{\mathrm{d}c}{\mathrm{d}t} = \gamma\,(1 - p_{\mathrm{c}}) N\, L_{>} - d_{\mathrm{c}}\, c \qquad L_0 = \left(1 + \frac{\mu}{\beta}\right)\ln 2$$
$$\frac{\mathrm{d}c_{\mathrm{r}}}{\mathrm{d}t} = \gamma\, p_{\mathrm{c}}\, N\, L_{>} \qquad OD = \eta(N\, L + c(t) + c_{\mathrm{r}}(t))$$

$$Y_{>}\left(x = \frac{L}{L_0 L_{\mathrm{m}}}\right) = \begin{cases} \frac{x}{\nu}\left(x^{\nu} - \left(\frac{x}{2}\right)^{\nu}\right) & \text{for } x \le 1 \\ x - 1 + \frac{x}{\nu}\left(1 - \left(\frac{x}{2}\right)^{\nu}\right) & \text{for } 1 \le x \le 2 \\ 1 & \text{for } 2 \le x \end{cases}$$

$$\frac{L_{>}}{L_0 L_{\mathrm{m}}}\left(x = \frac{L}{L_0 L_{\mathrm{m}}}\right) = \begin{cases} \frac{x}{\nu \ln 2}\left(x^{\nu} - \left(\frac{x}{2}\right)^{\nu}\right) & \text{for } x \le 1 \\ x\frac{\ln x}{\ln 2} + \frac{x}{\nu \ln 2}\left(1 - \left(\frac{x}{2}\right)^{\nu}\right) & \text{for } 1 \le x \le 2 \\ x & \text{for } 2 \le x \end{cases}$$

Variable	Unit	Comment
t	h	Time
$L(t)$	1	Average cell length of the population [au]
$N(t)$	1	Number of individuals in the population
$s(t)$	g/L	Concentration of nutrients
$a(t)$	mg/L	Concentration of antibiotics
$b(t)$	mg/L	Concentration of β-lactamase (the enzyme)
$c(t)$	1	Dead degradable biomass

Variable	Unit	Comment
$c_{\mathrm{r}}(t)$	1	Dead non-degradable biomass

Parameter	Unit	Comment
γ	$1/\mathrm{h}$	Death rate
β	$1/\mathrm{h}$	Maximal division rate
μ	$1/\mathrm{h}$	Maximal growth rate
K_{s}	$\mathrm{g/L}$	Half-velocity constant of nutrients
λ	$\mathrm{L/g}$	Conversion factor from nutrients
k_1	$\mathrm{mg/L}$	Concentration of antibiotics needed to stop cell division
h_1	1	Hill coefficient of this antibiotic action
k_2	$\mathrm{mg/L}$	Concentration of antibiotics needed to stop defect repair
B_{in}	$\mathrm{mg/L}$	Concentration of β-lactamase released by a cell of length 1
k_{b}	$\mathrm{L/mg/h}$	Activity rate of β-lactamase
d_{a}	$1/\mathrm{h}$	Degradation rate of antibiotics
d_{b}	$1/\mathrm{h}$	Degradation rate of β-lactamase
d_{c}	$1/\mathrm{h}$	Elimination rate of dead biomass
p_{c}	1	Proportion of non-degradable dead biomass
L_{min}	1	Minimal cell length where lysis can occur
L_{max}	1	Maximal viable cell length
η	1	Conversion between biomass and OD

5.1.2.3. Experimental observations for model fitting

To correctly and efficiently implement optimal designs methods, it is essential to well understand how our experimental plan becomes a dataset. To do so, we must ask ourselves what is it that our experiment setup allows us to measure. Virgile had two ways to collect information on the model:

1. Via Optical Density (OD) measurements. These measurements are the easiest, and well automated. To perform them, we used a multi-mode micro-plate reader (model Spark, Tecan Group Ltd., Austria) allowing frequent measurement for up to 60 hours. The commonly held belief is that OD measurements are dependent on the number of individuals in a bacterial population. However, as shown by Koch in the 1960's [53], OD actually depends on the biomass of bacteria. When individual bacteria have equal masses, the total biomass only depends on the number of cells, but in our case, where cells will elongate and thus have widely varying lengths at each instant, OD is more a measure of their added lengths (assuming bacteria have a cylindrical shape and constant density). In the ODE model of Virgile, where the quantity L stands for the average length of a cell, and N for the number of cells, OD is thus dependent on the product NL.
2. Via Colony Forming Units (CFU). CFUs are a way to measure the number of individual cells in a given volume of solution. To do so, a Petri Dish is prepared, filled with a nutritive agar. The solution is spread on top of the plates, which are then incubated. During the incubation, each microscopic cell will turn into a macroscopic circular colony on the agar. If the concentration of cells in the volume was not too important, colonies will be well separated, and each of them will correspond to a single colony forming cell. These circles can then be counted to deduce the number of bacteria that were put on the plate, and hence the concentration in bacteria/mL of the original solution. This process is very tedious and does not lend itself well to automation, but it would allow us to identify a quantify dependent on N only.

Because our work on optimal experimental design was done with the aim of being applicable to the platform in an automated way, we decided to forgo CFU measurements and consider that only OD measurements are available to us.

5.1.2.4. Identifiability of the model

The model suffers from several non identifiabilities, some structural and some practical.

First, we cannot measure the amount of β-lactamase enzymes B, hence, we cannot differentiate between the case where cell would produce a large amount of poorly efficient enzyme, or few highly efficient ones. We only have (indirect) access to a kind of "units of efficiency at destroying the antibiotic" produced per cell. Mathematically, we can see that the the ratio of B_in and k_b is not identifiable. Hence, in our numeric explorations, B_{in} and η will be fixed. Another structural non-identifiability comes from the fact that we only measure OD, and not CFUs, making it impossible to identify the η parameter, which will hence also be fixed.

Besides these structural (and solvable) non identifiabilities, the model is also ripe with practical non-identifiabilities. A detailed study of those is provided section 4.2 page 107 of [3]. In fact, non-identifiabilities are so present, and dependant on the "true" parameters of the system generating the data, that Virgile used them later in his work to classify the strains in different categories based on what axes of the parameter space were non-identifiable.

5.2. Demonstrating the potential of optimal design for the characterization of the antibiotic resistance model

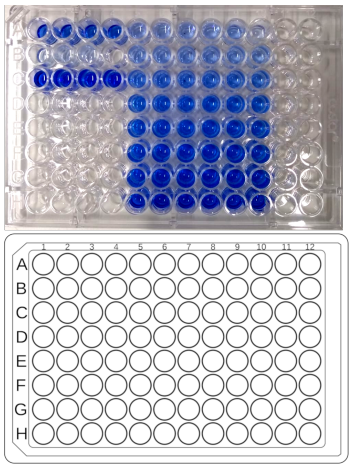
Presentation of original work resumes here.

Virgile's model is an excellent candidate to apply optimal design to. It is a complex model, with non-identifiabilities making it challenging, but also one that has not been specifically engineered for the sole purpose of demonstrating optimal design in silico. In this section, I will demonstrate that optimal design can yield experimental plans that are more informative than those of an expert. Because the goal is only to demonstrate the potential of optimal design, this study is carried out in artificial conditions where the design is computed using the true parameters of the in silico system. A realistic scenario is studied in the next chapter.

5.2.1. Defining a design space

To define a space of possible experiments that must be explored, I drew inspirations from the experiments that Virgile actually carried out. These experiments were time series measurements of optical density data, taken across up to 48 hours at a frequency of about 20 OD reads per hour. Most of the characterization experiments were done while injecting antibiotic only at the initial time. The initial concentration of nutrients in the media was fixed, leaving two variables to define the experiment: the initial optical density and antibiotic concentration.

In the rest of this work, the constraints of design will be as follows. k initial OD values are chosen. Those are imposed and their values is not part of the design space. For each initial OD, the designer (human or machine) is allowed to pick n initial antibiotic concentrations. In this $k \times n$ design space, there are hence kn degrees of freedom. Because at the bench, these kn parallel experiments are undertaken in kn wells of a microplate, I will often refer to them as "wells" (see margin Figure).



Top: picture of a 96 well microplate, with some of the wells filled with a gradient of blue dye. Bottom: Schematic of a 96-well plate [23].

5.2.2. Instantiating our validation experiments set

Our assessment pipeline (cf. 4.2.2.2) requires a set of validation experiments. They are all considered done in $200\ \mu\mathrm{L}$, for 48 hours, with OD measured every 10 minute. The following combination are possible:

1. An initial OD equal to either $5 \cdot 10^{-5}$ or $5 \cdot 10^{-4}$ is chosen. (two possibilities)
2. An initial dose of antibiotic equal to either 0 or 2^i with i an integer, $-10 \leq i \leq 11$. (twenty-two possibilities)
3. Either no re-injection is chosen, or a re-injection time equal to 2,4,6 or 8 hours is chosen. If a reaction time t is chosen, then at time t 2 µL of solution concentrated to a hundred times the initial dose are injected. (five possibilities)

Overall, this leaves us with 220 possibilities. It is noteworthy that the validation experiments include inputs that cannot be seen during the learning phase, given than 80% of them include a re-injection of antibiotic, as detailed below.

5.2.3. Gaining intuition on the model

A first step was to better understand if optimal design was needed at all. Indeed for some biological systems, the most informative experiment can be trivial, typically when it is obtained by minimizing or maximizing the input. However, in our study of antibiotic resistance, the input (the initial dose of antibiotic) must be carefully chosen.

When exposed to antibiotics, the population of bacteria can be in three phases:

- Biomass production: biomass is produced, leading to an exponential growth of OD. This can either be because cells are elongating, or because they are multiplying (or a combination of both)
- Cell-death and depollution: cells have elongated past their critical length and are now dying. Their death releases enzymes that degrade the antibiotic. The OD decreases, but much more slowly than it increased, because remaining bio-material still contributes turbidity.
- Population extinction: all cells have died, and the population won't be able to resume growth.

An ideal experiment should allow us to explore both the exponential growth phase, and the depollution phase, without killing off the population which would destroy our proxy measurement of the antibiotic depollution via resumption of exponential growth. Too little antibiotic and cells will grow as if there was none, too much antibiotic and cells will quickly die off, and we won't be able to identify when the environment was depolluted.

Let's consider the case where k and n are equal to one (a single well at fixed OD), making the space of possible experiments one-dimensional. We can divide this space in three regions (see Figure 5.4.):

- The "low antibiotic" one, where there is either no antibiotic, or so little antibiotic that there is no observable effect on the OD behavior during the whole experiment.
- The "medium antibiotic" one, where the initial antibiotic dose is such that growth will stop and bacteria will die, but the enzyme-mediated depollution will manage to clean enough of it and growth will resume.

- The “high antibiotic” one, where the antibiotic dose is so high that once growth has stopped, the depollution will not be enough to remove antibiotic (in the experiment duration), and the OD will plateau where it stopped. Note that the cut between the “medium” and “high” region can be dependent on the chosen duration T of experiments, although some doses of antibiotic will be high enough to have killed the whole population before the environment is depolluted, nd hence will prevent regrowth even after an infinite amount of time.

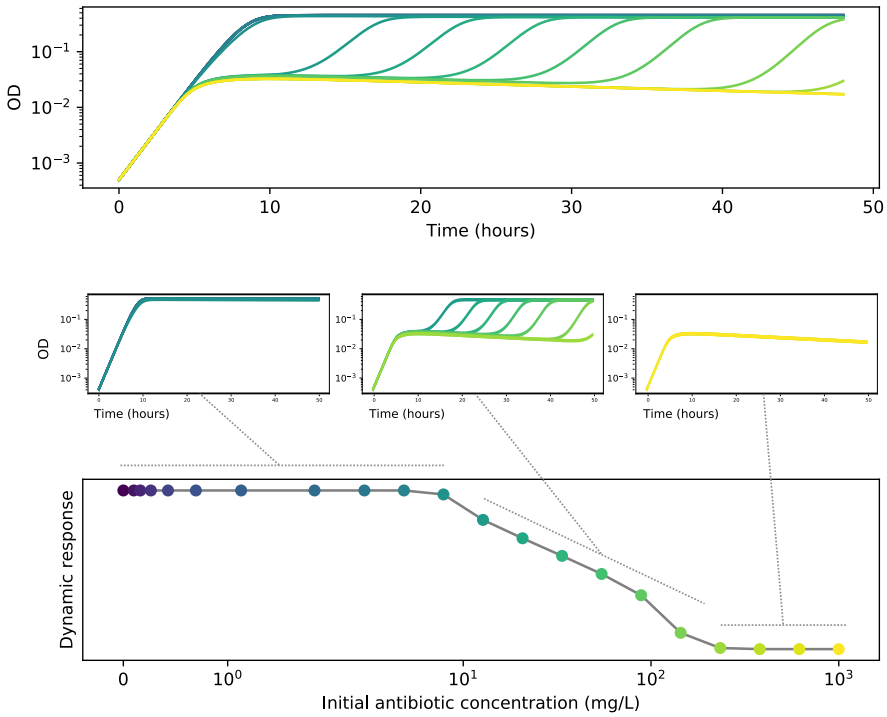


Figure 5.4.: **Top:** simulated behavior for various initial concentrations of antibiotic. **Bottom:** A *dynamic response* plot. The two extreme regimes are shown as plateaus left and right. The intermediate regime is in between. Colored points represent the dynamics from the top panel. **Middle:** Three inserts, each representing a region in the antibiotic space, from left to right: no effect of the antibiotic, an antibiotic dose that shows regrowth, an antibiotic dose high enough to kill off the population

This led me to try to identify a quantity that could be used to quickly and visually assess what an informative dose of antibiotic would be. To find it, I observed that, starting from the OD curve with no antibiotic, gradually increasing the initial antibiotic dose has a graphical effect

akin to running a squeegee on the curve and gradually flattening it out. A candidate quantity would then be to compute the time at which growth resumes, if any, but this requires deciding upon a threshold in advance, and can be tedious to do. Instead, we can remark that the area under the curve will diminish when it is "flattened" as the antibiotic increases. Hence, the integral: $\int_0^T \log(\text{OD}(t))$ is a good measure. I name this quantity the "dynamic response" of the system to antibiotic. Mathematically, for a given model, it is a function of the parameters, the initial OD, and the initial amount of antibiotic.

5.2.4. Defining an expert

As discussed in 4.2.2, prediction power can be used to compare the quality of two designs. To determine whether or not our optimal experimental design is of good quality, we thus need to compare it to the design that a human expert of the field would have come up with. Through discussion with Virgile Andreani, and based on his PhD work, I determined an algorithm that the expert would follow to choose the antibiotic doses in the $n \times k$ space. For each OD, a culture is grown without antibiotic. The $n-1$ remaining doses are chosen geometrically spaced, between 0.5 and 512 mg/L, as illustrated Figure 5.5 and 5.6.

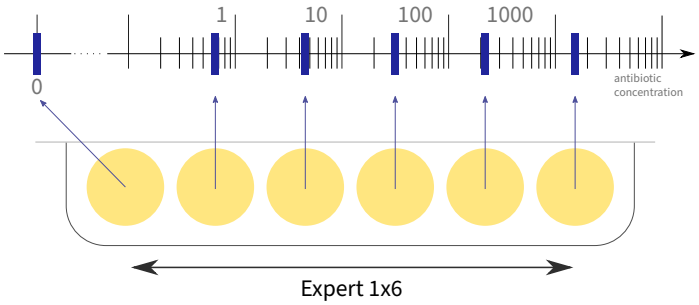


Figure 5.5.: The Expert design method: for each OD, a culture is grown without antibiotic. The $n-1$ remaining doses are chosen geometrically spaced, between 0.5 and 512 mg/L.

5.2.5. Experimental design with the true parameters

Before trying to apply a realistic iterative design process, we first need to check that if we were to design while already knowing the

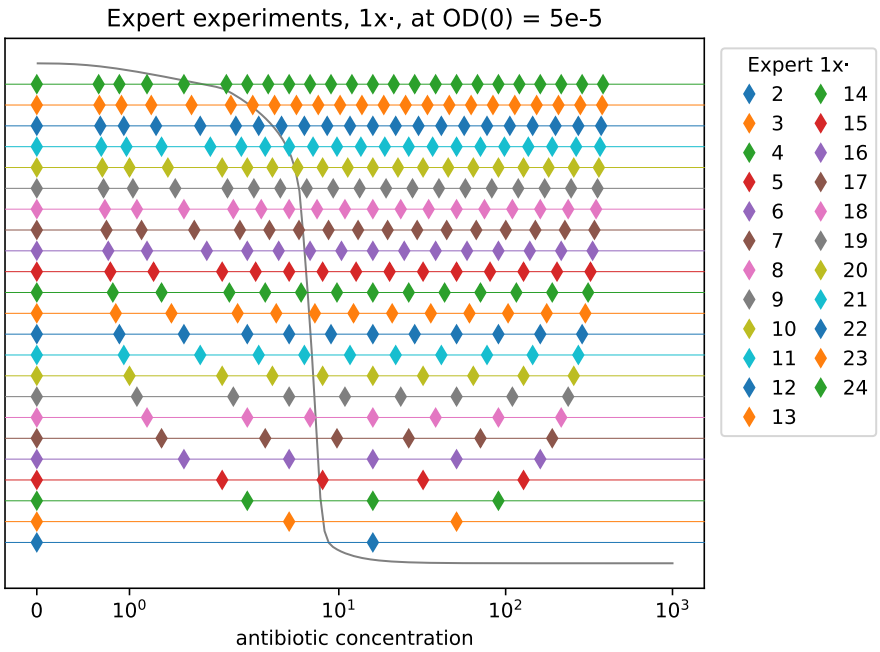


Figure 5.6.: Doses of antibiotics chosen by the expert for one initial OD (equal to $5 \cdot 10^{-5}$), and between 2 and 24 wells per OD. Each horizontal line represents one value of k. On each line, the diamonds represent the chosen antibiotic doses. The x-axis is in symlog scale (i.e. it is log-scaled, except close to zero where it is linear). The y axis has no meaning with respect to antibiotic doses. The gray line represent the dynamic response.

true parameters, our experimental design would prove to be better that the one the expert came up with. We thus proceed to design an experimental plan using two to twenty-two wells, using our FIM based method with a point estimate equal to the true parameters, illustrated in Figure 5.7.

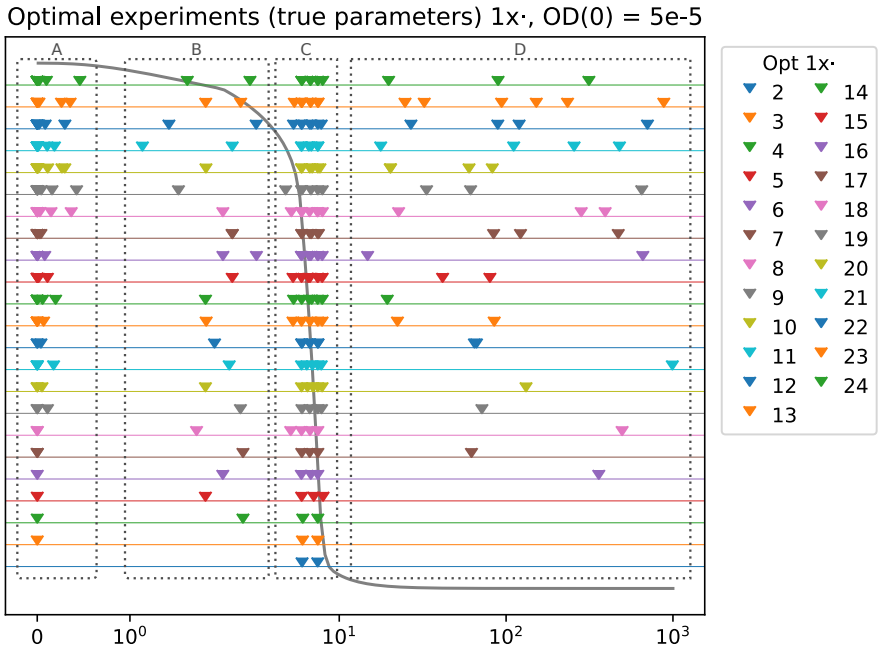


Figure 5.7.: Doses of antibiotics for the optimal design done with the true parameters, for one initial OD (equal to $5 \cdot 10^{-5}$), and between $k = 2$ and $k = 24$ wells per OD. Each horizontal line represents one value of k. On each line, the diamonds represent the chosen antibiotic doses. The x-axis is in symlog scale, and the y axis has no meaning with respect to antibiotic doses. The gray line represent the dynamic response. The doses chosen by the algorithm can be separated in four categories which have been boxed with dotted black lines, lettered A through D, and are discussed in the main text.

The resulting family of designs exhibits several interesting features. The designs follow a similar pattern: doses from the design for n experiments are roughly included as they are in the set of doses for $n + 1$ experiments. Even though all experiments have been designed in parallel, the results are similar to the ones that would have been obtained with a serial design. Doses are first put in the area identified as C in Figure 5.7 – in the slope region of the dynamic response, then a zero dose is added (area A), then a dose in the high plateau of the dynamic

response (area B) and finally a dose in the low plateau of the dynamic response (area D).

The four areas can be understood as:

- **A** is the area where the (nearly) zero doses are. These doses are identified as interesting by the optimal design, which follow our intuition that observing undisturbed cell growth, though not very informative on its own, is a key part of a more complex experimental plan.
- **B** and **D** are pretty similar in that they both are the doses for the respectively high and low plateau of the dynamic response. The algorithm explores them when it can afford (i.e.. when the total number of experiments is high enough) but nonetheless favors A and C.
- Finally, **C** is at the crux of our intuitive reasoning. We discussed in 5.2.3 how the sloped part of the dynamic response was likely to be where doses of antibiotic leading to informative behaviors were, and it is indeed where our algorithmic design put a good number of them.

Overall, we thus consider that the optimal design has converged towards a solution which match with our intuition of what an informative design would be for the characterization of Virgile's model. To further demonstrate the relevance of our method, we will next compare its prediction power to the one of the expert's designs.

5.2.6. Assessing the quality of our tentatively optimal design

We introduced in 4.2.2.2 and Figure 4.10 a method to compare two experimental designs: the cave plot. Cave plots offer an exhaustive view and comparison of the predictive power of two different designs, but in our case, they scale poorly to offer a one-glance visualization: both our (would-be) optimal and expert-like designs methods are algorithms that can be parametrized by the total number of wells. To compare the distribution of prediction errors (marginalized over every validation experiments) at equal well number we use ridge plots, such as the one illustrated in Figure 5.8.

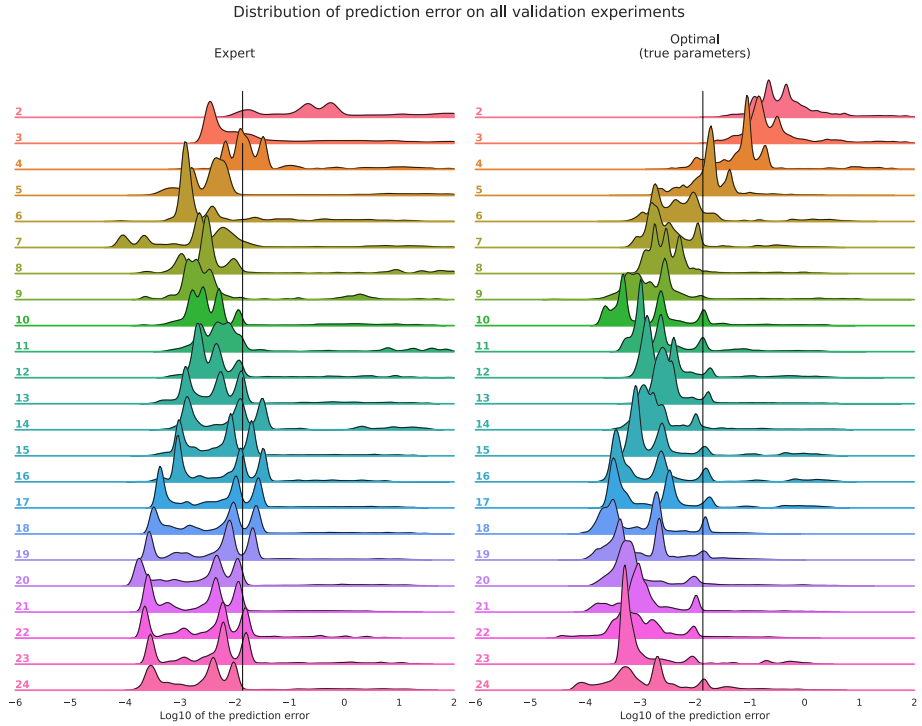


Figure 5.8.: Distributions of the log10 of the prediction error across all validation experiments, for the expert designs and the optimal designs computed with the true parameters. The number of parallel experiments is indicated on the left (from 2 to 24). The solid black vertical line indicate the threshold below which the error is considered negligible.

This comparison is interesting, but even more so is to compare the designs quality across the number of parallel experiments that are allowed. To this end, I introduce comparison matrices as the ones presented Figure 5.9. These matrices allow one to quickly picture the variation of probability of an event such as "the experts prediction will be better". In them, each cell represents, for a uniformly chosen validation experiment, a uniformly chosen parameter sample learnt on the optimal design, and a uniformly chosen parameter sample learnt on the expert design, the probability of the event indicated in the title and legend. Events looked at can be:

- The probability that the optimal design leads to a visibly better prediction: that is at least one of the prediction is making an error higher than the fixed threshold, and the error of the optimal design is less (Figure 5.9-Left).
- The probability that both experiments will correctly predict the validation experiment (Figure 5.9-Center).
- The probability of union of the (mutually exclusive) events above (Figure 5.9-Right).

Figures 5.8 and 5.9 allow us to make the following observations.

1. In the comparison matrices, a remarkable threshold effect is observed, where the quality of the design increases massively from six wells on. This effect is strong in all three matrices, and shows that for six wells or more, the optimal design is strictly better than any expert design, even with 24 wells.
2. This cut-off can be identified in the ridge plots, where most of the prediction error coming from the optimal design becomes negligible once there are six wells or more.
3. For the expert design, the ridge plot shows an oscillation pattern of the error when the number of wells varies, which can also tenuously be seen in the matrices. This is because the expert will only have doses in the interesting sloped response dynamic by "chance": the more wells there are the more likely it becomes that some doses will be well placed (explaining the global trend of error decrease) but nonetheless, oscillations in the number of doses close to the interesting zone will occur, leading to the oscillations in prediction error.

It could be argued, especially after observing the ridge plot, that

Figure 5.9.: Comparison matrices of the predictive power of optimal designs obtained using the true parameters (y-axis) and expert designs (x-axis). **Left:** The value of each cell is the probability (as a percentage) that the prediction coming from the optimal design is visibly better. Cells are colored according to their value, red being a probability < 0.5 and blue > 0.5. **Center:** Probability that both prediction are correct. **Right:** Probability that the optimal design's prediction is not visibly worse than the expert design's.

the advantage of one design over the other is massively dependant on the choice of the negligibility threshold. However, I generated the comparison matrices corresponding to no threshold (Figure 5.10) and though the advantage is less clear cut, the optimal design remains better than every expert design for six wells and more.

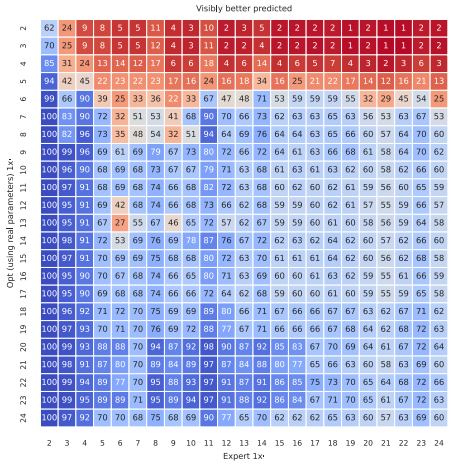


Figure 5.10.: Same as Figure 5.9-Left, but no error is considered negligible. Still, the optimal design beats the expert once afforded enough wells.

Hence, we have shown that when performing optimal design with the true parameters known, we can find designs that are more informative than those of an expert. Unfortunately, true parameters will not be known in advance: all we will have access to are estimates of the true parameters. Our study has so far demonstrated that there is room for improvement comparing to the expert's design, but it remains on the table whether or not a better design can actually be achieved in a realistic scenario.

6. Optimal design of characterization experiments in a realistic scenario

6.1. Our iterative setup

To demonstrate our method in a realistic setting, we study the following scenario of iterative optimal design.

In our iterative design (see section 4.2) a distribution is used as input to the design algorithm, representing our current knowledge of the parameters. Thus the question of which distribution shall be used as initial estimate arises. A common choice would be to have some generic distribution, typically a Normal one, that covers the set of plausible parameters values. However, as discussed in section 4.1.1, the geometry of the parameter space is actually pretty different from the geometry of the behavior space, and because our method is computationally limited in the number of samples from the parameter distribution it can use, it becomes very hard to identify a distribution of parameters that represent well a non-informative prior. Another possibility would be to use different parameter estimates from a library of different strains of bacteria, and use these as "samples" for the initial design. This solution is appealing because it has strong guarantees that our prior would cover plausible behaviors of the system. Unfortunately as of the realization of our study, libraries of strain do exist but their characterization remains to be done, and only a few datasets of satisfying quality are available.

If we take a step back and reflect on how the prior affects the optimal design, we see that it is by inducing a distribution of positions for

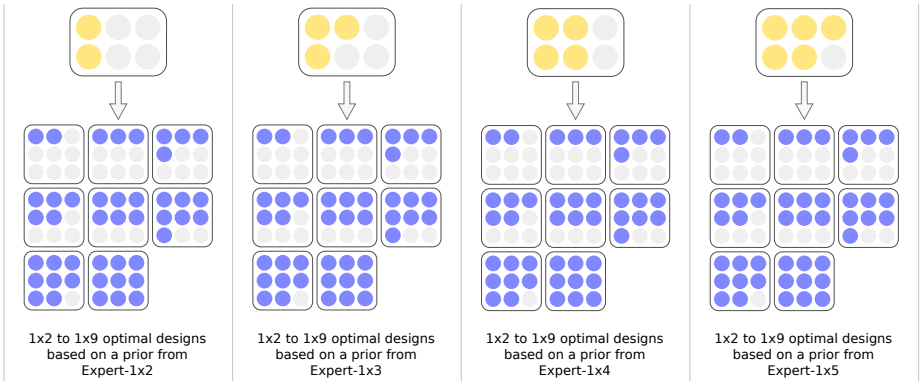


Figure 6.1.: Graphical summary of the iterative design setup: four expert designs can be used as a first learning step, and parameters obtained by learning on them are then used to design eight optimal designs.

the slope of the dynamic response, and hence the interesting region discussed in 5.2.3, and illustrated afterwards in Figure 6.3. An uninformative prior would imply that the dynamic response can start and end its dive down at virtually any dose of antibiotic. Hence, we posit that, for such prior, the optimal design would actually be close to the one of the expert: a wide range of doses spanning all reasonable positions for the dynamic response slope. Thus we elect to use design of the expert as initial designs.

These first experiments are used to obtain a set of estimates of the parameters, which is then used as a prior to our designing algorithm detailed in 4.2.1.3. We choose to use as first experiments expert designs with one initial OD and two to five wells, and for each of these four initial designs, we create eight optimal ones, with one fixed OD and two to nine wells, as illustrated Figure 6.1. In total, we will thus have $4 \times 8 = 32$ possible combinations of experiments.

To well understand the process of iterative design, I choose to take two different steps. First a prior is indeed used, but to design experiments that should be informative on their own. As in the section were we designed with the true parameters, these “standalone” experiments will be assessed by themselves. In a next step, experiments will be designed to be used in conjunction with the expert experiments which originated the design prior, and the combination of the two will be assessed. These two possibilities are illustrated Figure 6.2.

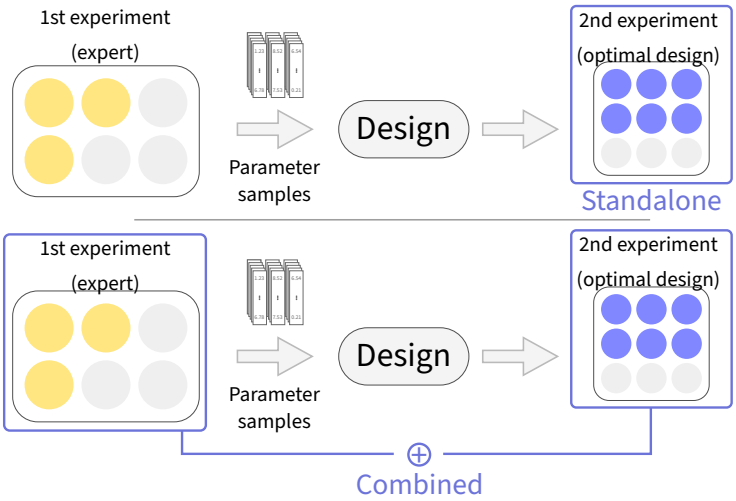


Figure 6.2.: Our two kinds of designs, either **Top:** standalone or **Bottom:** combined. The blue box indicates what experiments are used for the final fit.

6.2. Priors resulting from the expert

Before presenting the resulting designs, it is worthy to check how precise are the priors. To do so, for each of the four expert experiments, I have simulated fit samples (i.e. samples from the prior) and plotted their corresponding dynamic response in Figure 6.3.

We clearly see on these plots that as the number of wells increases, the distribution of dynamic responses narrows down. Indeed, for the five-wells expert, the dynamic responses of the learned parameters nearly align with the one of the true parameters. However, we have seen in the previous section that the expert designs can be beaten by optimal ones, meaning it is not perfect. A possible explanation for this apparent paradox is that our validation set is much more comprehensive, especially as it consist of 80% of experiments with re-injection, and of 50% of experiments at another OD, which our dynamic response plot does not cover. Another noteworthy fact is that for all four plots, the dynamic responses are all crossing where the antibiotic doses were, which is explained by the fact that it is at these doses that we observed the system, and hence where all our fits will agree. This even further consolidates our intuition: new experiments should be designed by putting antibiotic doses at concentrations for which the dynamic response induced by the prior varies the most, because measuring the

behavior at these points will “resolve” all this variation to one point.

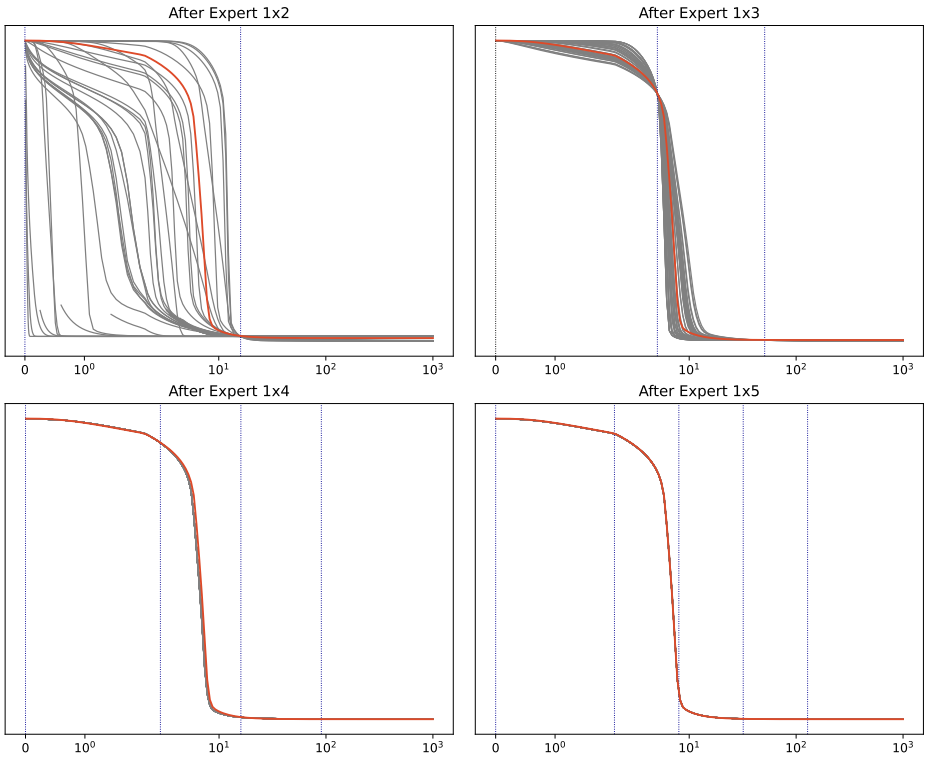


Figure 6.3.: Dynamic response of parameter samples learned from the expert (with 2,3,4 and 5 wells) experiments. Gray solid curves represent the dynamic response, the solid red curve represents the dynamic response of the true parameters. Dotted vertical blue lines represent the antibiotic doses of the corresponding expert experiment.

6.3. Designing with the priors

These designs are presented in Figure 6.4. The following observations can be made:

- When the prior comes from the two-wells expert, it is so wide that the design algorithm, when given six wells or less places all the doses around the expected slope of the dynamic response, and does not afford exploring elsewhere. it is important here to stress once again that the design here is being done for experiments meant to be carried out from scratch: the rest of the dose space is unexplored not because it has been seen previously by the expert’s design, but because the design algorithm finds it better

to maximizes its odds of having an experiment well in the dynamic response slope.

- For the prior coming from the expert with three wells, the algorithm goes back to its behavior of the previous chapter, except that the slope is a bit wider, leading to doses of this area being a bit more spread apart, and without affording to explore the two plateaus of the dynamic response.
- Finally for priors close to the true parameters (four and five wells), the computed designs highly resemble those discussed in the previous chapter.

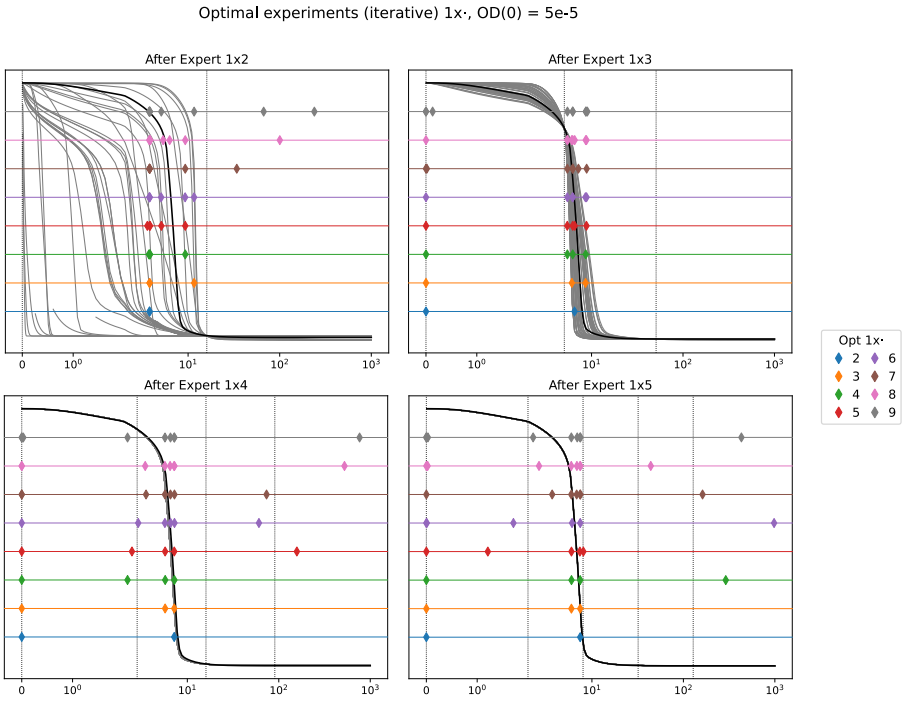


Figure 6.4.: Doses of antibiotic from the standalone optimal designs (one initial OD – equal to $5 \cdot 10^{-5}$ – and between $k = 2$ and $k = 9$ wells per OD). Each horizontal line represents one value of k. On each line, the diamonds represent the chosen antibiotic doses. The x-axis is in symlog scale, and the y axis has no meaning with respect to antibiotic doses. In the background, a gray-scale version of Figure 6.3.

Regarding prediction power (cf Figure 6.5), our designs from priors clearly improve as the number of wells used for the prior increases. Concretely, designing from the poor prior of the two wells expert appears nearly pointless, as all designs will be of less quality than those of the expert, no matter the number of wells. For higher number of wells, we

find again a pattern of clear cut, where once afforded six wells or more, the optimal designs surpass virtually all of the expert's designs.

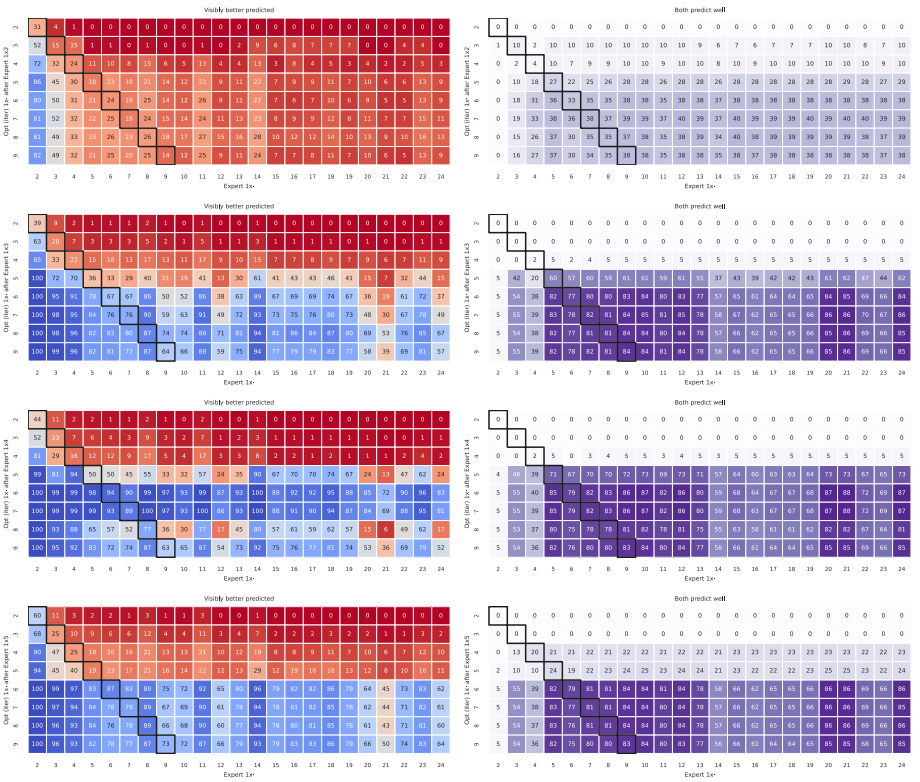


Figure 6.5.: Comparison matrices of the predictive power of standalone optimal designs obtained using priors from the expert (y-axis) and expert designs (x-axis). **Left:** The value of each cell is the probability (as a percentage) that the prediction coming from the optimal design is visibly better. Cells are colored according to their value, red being a probability < 0.5 and blue > 0.5. **Right:** Probability that both prediction are correct.

6.4. Designing with the priors to add to our knowledge

We have now shown that priors can be effectively used to compute new designs. To conclude our study, we demonstrate a realistic iterative learning scenario where experiments from the expert's design would be first carried out to produce a first set of plausible parameter vectors. This parameter vectors are then used as prior to design new experiments

that complete the one already carried out (cf. Figure 6.2-Bottom). The priors are still the one presented in above and in Figure 6.3, and the computed designs are presented Figure 6.6.

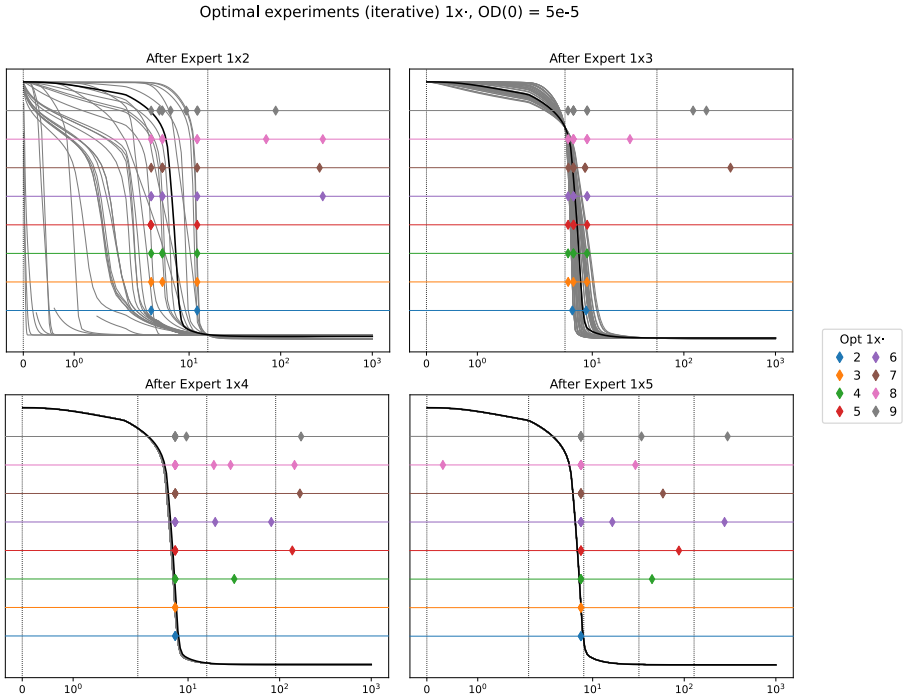


Figure 6.6.: Doses of antibiotic from the optimal designs used in combination with the expert designs (one initial OD – equal to $5 \cdot 10^{-5}$ – and between $k = 2$ and $k = 9$ wells per OD). Each horizontal line represents one value of k. On each line, the diamonds represent the chosen antibiotic doses. The x-axis is in symlog scale, and the y axis has no meaning with respect to antibiotic doses. In the background, a gray-scale version of Figure 6.3. **Remark:** The complete set of doses that will be used to learn can be read by taking into account both the diamonds and the dotted vertical lines.

These designs are strikingly more huddled up than any other previous ones: apart from a few doses in the low plateau of the dynamic response, most doses are at the dynamic response slope, virtually equal to one another. Notably, there is no zero dose. The algorithm has identified that the expert's experiments are best completed by putting doses where the sensitivity is the greatest.

The prediction power matrices (figure 6.7) of these designs reveal a very different picture than the preceding discussions:

- When using the very wide prior coming from the two-wells expert

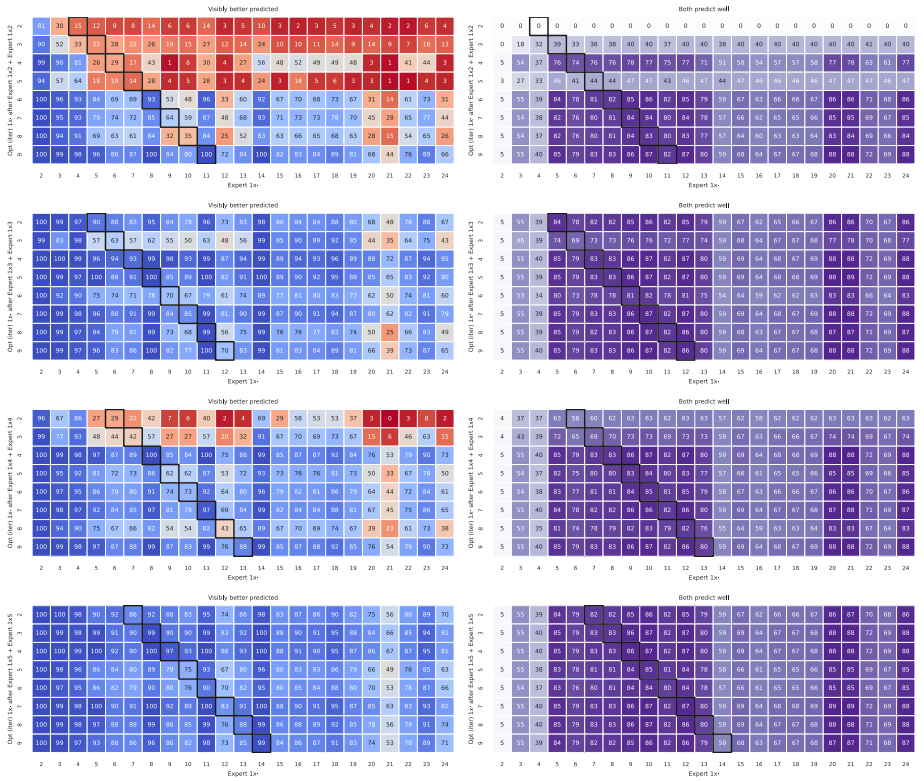


Figure 6.7.: Comparison matrices of the predictive power of to-be-combined optimal designs obtained using priors from the expert (y-axis) and expert designs (x-axis). Cells where the expert alone and the combination of the prior expert and the optimal design use the same number of wells are highlighted bty a black box. **Top to bottom:** Increasing number of wells used to generate the prior. **Left:** The value of each cell is the probability (as a percentage) that the prediction coming from the optimal design is visibly better. Cells are colored according to their value, red being a probability < 0.5 and blue > 0.5. **Right:** Probability that both prediction are correct.

(top row), it is remarkable than optimal experimental design can still present an advantage: by designing six or more experiments to be performed on top of the two already done, we can in most cases gain more information than if we had used the same total number of wells to perform only the expert design. However, optimal design with five wells or less done on such a wide prior can be defeated by expert designs with one well less than the total of our combined method. This can likely be explained by an ill placement of the non-zero antibiotic dose, "wasted" by Expert-1x2.

- However, as soon as the prior becomes slightly more precise, the advantage of adding even only two doses designed algorithmically is glaring. With priors coming from experts with three wells or more, it is nearly always better to use the remaining wells to design optimal experiments than it would have been to perform an expert with more wells.
- The combination of Expert-1x3 and its associated two-wells optimal design is also noteworthy: with a total of five wells it beats any expert design, when the optimal design done with the true parameters needed six wells (i.e.. one more) to offer such strong guarantees.

7. Conclusion and perspectives

This chapter summarizes the work presented in this manuscript, discusses the contributions outlined herein, and presents future perspectives of this line of research.

7.1. Summary

In this manuscript, we first proposed methodological developments aiming at designing experiments in a setting where non-identifiabilities make Bayesian experimental design computationally intractable and the Fisher Information Matrix non-invertible. To this end, we developed a family of algorithms to identify locally the direction of non-identifiabilities and decide whether these non-identifiabilities are inherent to the model (in which case they do not need to be resolved) or can be blamed on a specific experimental plan (in which case this experimental plan is not suited to properly predict the system's behavior). In this family of algorithms, we discussed how adversarial design is reminiscent of min-max strategies in game engines. This development was guided by considerations on the relationship between the parameter space, the behavior space, and the "iso-behavior" surfaces of the parameter space. However, these methods could not be translated to practical implementations, as numerical instabilities in the matrix diagonalization lead to too many protocols being wrongfully categorized as leading to non-identifiabilities.

Hence, we fell back to a more classical design method, inspired by the Bayesian Cramér-Rao bound. In this method, experiments are designed for parameter identification. However, we also proposed a distinct quality assessment pipeline to evaluate the experimental plans proposed by

our design method in light of the prediction power we gain by learning parameters on datasets obtained from said experimental plans. Our design pipeline uses a prior distribution of parameters to compute an information score and maximizes this information score using a stochastic numerical optimization routine CMA-ES. Our assessment pipeline is more complex: for each experimental plan to be evaluated, we simulate corresponding datasets and fit on them. Because we only select fits that have converged to a solution at least as good as the true parameters, this step requires that the optimization routine is run thousands of times, incurring massive computational costs. Once these parameter fits have been obtained, they are used to predict the dynamic of the system on a predefined set of two hundred twenty validation experiments. These predictions are then compared to the true behavior of the system for the validation experiments (which is known in our in silico study), and we obtain a set of two hundred twenty prediction quality distribution for the experimental protocol we wanted to assess (see Figure 4.5 and Figure 4.6).

These score have little interpretability on their own. However, they can be used to compare two experimental plans to one another. To do so, different visualizations were introduced. First, plots comparing the distributions of the prediction score for different experimental plans can be used. For a handful of experimental plans, classical whiskers-box (such as Figure 4.8) can be used, and when we want to compare inside a larger family of designs, ridge plots (such as Figure 5.8) are a good choice. To compare two designs, we defined cave plots (such as Figure 4.10), which illustrate the fundamental probabilistic aspect of our comparisons. Finally, to compare the numerous designs generated in our case-study, we present matrix comparisons (such as Figure 5.9), a condensed version of cave plots.

To support both pipelines, and allow us to generate the rich amount of results presented in the aforementioned visualizations, we also developed a software ecosystem. We developed Fwd:AD, a Rust library for automatic differentiation in forward-mode with one founding principle: optimize memory usage, and minimize memory copy at the interface of the library. We also developed bindings to sundial's ODE solver cvodes, using Fwd:AD to compute the sensitivities of the right-hand side and leverage sundials sensitivities capabilities to integrate both the initial value problem solution and its derivative with respect to the parame-

ters. We brought together these two pieces of software by developing a Rust macro implementing a domain specific language allowing to transform a user-friendly JSON-like description of an ODE model into high-performance machine code callable from any language. Finally, we developed a python library, interfacing with the machine code generated above to provide an higher-level user-friendly experience. It allows to easily simulate the model on various inputs, parameters, with or without the sensitivities, but also offers plotting capabilities as well as bindings to the theano machine-learning library for the model. Thanks to these bindings, we concluded that Bayesian inference was beyond our current abilities. To orchestrate the massive amount of jobs required by our pipeline, we developed Captain, a high-performance job manager. Naturally, one may wonder wether we practiced what we preached, and to what standard our own code can easily be shared and reused. We must concede that it varies from library to library. Fwd:AD for example has benefitted from extensive documentation, packaging, and public communication work. Adding it to your code base simply requires adding `fwd_ad = 0.2.0` to `Cargo.toml` a file describing a Rust project and present by default. On the other hand, odegen and captain still have not benefited from such work.

This methodological contributions enabled us to present our case-study: an application of optimal experimental design to the characterization of enzyme-mediated escape to antimicrobial treatments. In this study, we showed first that optimal experimental designs, when computed using the real parameters of the in silico system outperform designs made by an expert. To explain this, we showed that these designs can be interpreted a posteriori as exploring four zones of the space of possible experimental plans, and that these four zones can be linked to different responses of the system to antibiotic. We then switched to a more realistic scenario, or rather two sub-scenarios. In both scenarios, we used a limited number of expert experiments with varying information content as the first step of an iterative design process. Distributions of parameter fits obtained from these expert experiments were then used in an optimal design procedure. The difference laid in how these optimal design were done. In the first sub-scenario, the experimental plans were computed in the aim of being used on their own. The expert experiments had only been used to obtain a prior, but only the final design would be evaluated in our assessment pipeline. In the second sub

scenario however, we would design experiments meant to complete the ones of the expert, and it is the combination of those that would be evaluated. The first scenario highlights the capital importance of having learned enough from the expert to design meaningful experiments: for the least informative of the expert experiments, all optimal designs were of poor quality. However, it also highlights that there may only be little to be gained from investing too much in the first round of experiments. Indeed, once the prior is informative enough, subsequent refinements seem to provide little improvements in the quality of subsequent designs. In the combined scenario, the conclusions differ a little. Even for the least informative of our priors coming from expert experiments, using optimal design to complete the experimental plan can lead to better designs than the expert using the same number of wells are our prior expert plus our optimal design. When using more informative prior, this possibility becomes a certitude: when the prior comes from an expert that used three wells or more it is virtually always better to use the remaining well budget to design optimal experiments than it would have been to use them in a prior expert with more wells.

In conclusion, the manuscript presented methodological and software developments used to build large scale automated pipelines and leveraged those to perform an extensive study of optimal experimental design for the characterization of enzyme-mediated escape to antimicrobial treatments.

7.2. Discussion

Having recapitulated the content of this manuscript, I will now try to put it into perspective. The first point to be discussed regards the mathematical methods outline to design with non-identifiabilities for prediction power. Considerations on the FIM for prediction already exist in the literature, and relations between eigenvalues and non-identifiabilities are doubtlessly part of the mathematical folklore. However, there is to the best of my knowledge no previous statement of my proposed method. One may discuss the relevance of that method given that we did not manage to use it in practice, because of the numerical instabilities that plague matrix diagonalization – especially in orthonormal basis. However, numerical methods for matrix operations are a vast

area of research, and algorithms do exist to provide arbitrary precision in eigenvalue evaluation (to the price of course of more computation time). Hence, it is possible that with more work, this direction could lead to new developments, and the proposed mathematical method remain, in my opinion, relevant.

Regarding our design pipeline, getting inspiration from the Bayesian Cramér Rao Bound is no news. However, there was to the best of my knowledge no previous discussion of its use in cases where the analytical probability density function of the prior is unknown. In that regard, our discussion of the non usability of kernel density estimation, and the justification of the necessity of approximating further the Bayesian Cramér Rao Bound are novel. Regarding our assessment pipeline, I believe it is the first exhaustive description of such pipeline. As well, the discussion on the merit of using stochastic fitting method and ensuring that their output is indeed a better fit than the true parameters to clearly identify the set of "high likelihood estimates" is also a new contribution. It is true that this method may seem like an ersatz of Bayesian inference, and it is indeed conceptually less pure. However, we believe that the use presented here is sound and opens the way to a study that would not have been possible otherwise.

The scale of our pipeline also warrants discussion. As outlined in section 4.4, the results presented in this manuscript represent well over sixteen years of CPU time, on current generation hardware. In our opinion, computations at such massive scale were crucial to enable our study. It is only by producing enough fits of good quality that we can trust the conclusions drawn from them. That being said, in the context of the current global climate crisis, it is worth considering the scale of the energetic consumption at stake. A ballpark estimation of the power consumed by the cluster nodes at peak usage is 400W. Divided by the 96 cores and multiplied by 16 years, we obtain an energy consumption of around 584 kWh, which is on the same order of magnitude as 18 days of electrical consumption of the average US family. Put this way it seems that our energy consumption remained very reasonable, however this is only a lower bound as I only accounted for the computation used to generate once the final result while other versions of the pipelines ran multiple time during their development, and I did not take int account the networking and storage infrastructure of the cluster, nor its manufacturing, nor the idle time that the cluster had to go though to enable

me to launch thousands of jobs simultaneously.

Regarding our software contributions, the most pressing doubt that one may have is wether they fulfilled an actual need. Put bluntly: did I reinvent the wheel by re-implementing software that could be found elsewhere. To answer this question, it is needed to stress how much time was spent trying to get existing software to work. However, a programming library is only as usable as it is easy to install and its documentation easy to understand. It is only after failing many times to use or simply understand existing software, to great frustration, that I decided on implementing Fwd:AD for example. By doing so, I ensured that the library could actually provide the level of expected performances, and I knew that I fully understood the code and would not face issues several months down the line. There is no worse situation than having committed to a peculiar pre-existing software implementation only to discover after gruesome implementation work that it is in fact lacking a key feature for your use-case, and being too unfamiliar with its codebase to add it yourself. This situation presented itself when, after a year using it, I had to decide on replacing my then job manager snakemake which had become the bottleneck of my pipeline and code my own, captain.

7.3. Perspectives

As the three years (and three months) spent on this research project come to an end, it is only expected that I reflect on what its future extensions could be. Direct prolongations could be found by re-trying to solve the numerical stability issues preventing irrelevant component analysis. Many more optimal scenarios could also be run: comparison could be made to other experts, random designs or model-based experts, which would equally space doses in the slope region of the dynamic response. The evolution of information with the number of wells could be investigated in more depth as well. Experimental designs where the initial OD is freely chosen and antibiotic can be injected during the experiment could also be explored, though this would lead to a massive combinatorial exploration of the design space. More work could also be done regarding the interpretation of the designs proposed by the algorithm. For example, in cases where it is suspected that some

doses of antibiotic do not contribute much to the overall informativeness and are merely artifacts of the optimization routine, they could be deleted and the resulting experimental plan could be benchmarked. An other avenue would be to take this research to the bench. I actually attempted this during the first semester of my last year, but encountered too many issues to reasonably envision obtaining results in the time frame required by the PhD.

On a longer timescale, a general push to evangelize optimal experimental design to biologists as a common part of their toolbox seems needed. The importance of mathematical modeling and model fitting is now largely established and for optimal design to gain a similar status it seems needed to make tools even more accessible. Many specialized toolboxes exist, but optimal experimental design has for now remained absent of the default library and offering of virtually all used languages.

On an even longer term, one could envision to finish developing a fully automated platform for antibiotic resistance characterization. A possible perspective identified by V. Andreani's PhD was to use the model to perform model-based optimization of antibiotic treatment, ensuring that the bacterial population is completely destroyed with a minimal amount of antibiotic. Imagining that such model control has been demonstrated, and that application in clinical settings appear reasonable, an automatic characterization platform would be required to help transform clinical isolated of strains into personalized treatment schedules, and such platform would include optimal experimental design.

Appendix

A. Proof of section 4.2.1.1

Proof. We provide the proof for a one dimensional parameter space only. We want to show that

$$\mathcal{I}_{prior} = \mathrm{E}\left[\left(\frac{\partial \log \pi(\theta)}{\partial \theta}\right)^2\right]$$

depends on the bandwidth parameter of the kernel density estimation. Let's define:

- the $(\theta_i)_{0 \leq i \leq n-1}$ are the n parameter samples on which we are performing the KDE, with $\theta_0 \leq \theta_1 \leq \ldots$ etc.,
- f is the kernel of the KDE. Its domain is $\mathbb{R}$. We assume that:
  - f is a probability density function: it integrates to 1 and is positive.
  - f is an even function, meaning that $f(-x) = f(x)$,
  - f is increasing on $\mathbb{R}^-$ and decreasing on $\mathbb{R}^+$,
  - f is infinitely differentiable,
- h is the KDE bandwidth, a positive real number

The KDE estimate of the prior's density is:

$$\pi(\theta) = \frac{1}{nh} \sum_{i=0}^{n-1} f\left(\frac{\theta - \theta_i}{h}\right)$$

hence

$$\begin{aligned}\frac{\partial \log \pi(\theta)}{\partial \theta} &= \frac{\sum_{i=0}^{n-1} \frac{\partial f\left(\frac{\theta-\theta_i}{h}\right)}{\partial \theta}}{\sum_{i=0}^{n-1} f\left(\frac{\theta-\theta_i}{h}\right)} \\ &= \frac{\frac{1}{h}\sum_{i=0}^{n-1} f'\left(\frac{\theta-\theta_i}{h}\right)}{\sum_{i=0}^{n-1} f\left(\frac{\theta-\theta_i}{h}\right)}\end{aligned}$$

leading to

$$
\begin{aligned}
\mathcal{I}_{prior} &= \mathrm{E}\left[\left(\frac{\partial \log \pi(\theta)}{\partial \theta}\right)^2\right] \\
&= \frac{1}{h^2}\mathrm{E}\left[\frac{\left(\sum_{i=0}^{n-1} f'\left(\frac{\theta-\theta_i}{h}\right)\right)^2}{\left(\sum_{i=0}^{n-1} f\left(\frac{\theta-\theta_i}{h}\right)\right)^2}\right] \\
&= \frac{1}{h^2}\int_{-\infty}^{\infty} \frac{\left(\sum_{i=0}^{n-1} f'\left(\frac{\theta-\theta_i}{h}\right)\right)^2}{\left(\sum_{i=0}^{n-1} f\left(\frac{\theta-\theta_i}{h}\right)\right)^2}\pi(\theta)d\theta \\
&= \frac{1}{nh^3}\underbrace{\int_{-\infty}^{\infty} \frac{\left(\sum_{i=0}^{n-1} f'\left(\frac{\theta-\theta_i}{h}\right)\right)^2}{\left(\sum_{i=0}^{n-1} f\left(\frac{\theta-\theta_i}{h}\right)\right)}d\theta}_{\text{defined to be } A}
\end{aligned}
$$

We will now prove that there exist two constants (with respect to h) α and β such that: $\alpha h \leq A \leq \beta h$, and hence, $\frac{\alpha}{nh^2} \leq \mathcal{I}_{prior} \leq \frac{\beta}{nh^2}$, which will conclude our proof.

Upper bound

We start from

$$
A = \int_{-\infty}^{\infty} \frac{\left(\sum_{i=0}^{n-1} f'\left(\frac{\theta-\theta_i}{h}\right)\right)^2}{\left(\sum_{i=0}^{n-1} f\left(\frac{\theta-\theta_i}{h}\right)\right)}d\theta
$$

using Titu's Lemma[1] we find:

$$
\frac{\left(\sum_{i=0}^{n-1} f'\left(\frac{\theta-\theta_i}{h}\right)\right)^2}{\left(\sum_{i=0}^{n-1} f\left(\frac{\theta-\theta_i}{h}\right)\right)} \leq \sum_{i=0}^{n-1} \frac{\left(f'\left(\frac{\theta-\theta_i}{h}\right)\right)^2}{f\left(\frac{\theta-\theta_i}{h}\right)}
$$

hence:

$$
A \leq \sum_{i=0}^{n-1} \int_{-\infty}^{\infty} \frac{\left(f'\left(\frac{\theta-\theta_i}{h}\right)\right)^2}{f\left(\frac{\theta-\theta_i}{h}\right)}d\theta
$$

Now that we have been able to invert the integral and the sum, we can perform a change of variable $u = \frac{\theta-\theta_i}{h}$, giving our upper bound:

[1] Titu's lemma, also known as Sedrakyan's inequality is a corollary of the Cauchy-Schwarz inequality and states that for reals $a_0, \ldots, a_n$ and positive reals $b_0, \ldots, b_n$, $\frac{\left(\sum_{i=0}^{n} a_i\right)^2}{\sum_{i=0}^{n} b_i} \leq \sum_{i=1}^{n} \frac{a_i^2}{b_i}$. See [5].

$$A \leq \sum_{i=0}^{n-1} \int_{-\infty}^{\infty} \frac{f'(u)^2}{f(u)} h du$$
$$A \leq hn \int_{-\infty}^{\infty} \frac{f'(u)^2}{f(u)} du$$

Lower bound

We know that left of θ_0, all the $f\left(\frac{\theta-\theta_i}{h}\right)$ are increasing functions of θ and hence all $f'\left(\frac{\theta-\theta_i}{h}\right)$ are non-negative. Thus, for the numerator:

$$\sum_{i=0}^{n-1} f'\left(\frac{\theta-\theta_i}{h}\right) \geq f'\left(\frac{\theta-\theta_0}{h}\right)$$
$$\left(\sum_{i=0}^{n-1} f'\left(\frac{\theta-\theta_i}{h}\right)\right)^2 \geq \left(f'\left(\frac{\theta-\theta_0}{h}\right)\right)^2$$

In addition for the denominator:

$$\sum_{i=0}^{n-1} f\left(\frac{\theta-\theta_i}{h}\right) \leq \sum_{i=0}^{n-1} f(0) = nf(0)$$

so, for $\theta \leq \theta_0$:

$$\frac{\left(\sum_{i=0}^{n-1} f'\left(\frac{\theta-\theta_i}{h}\right)\right)^2}{\left(\sum_{i=0}^{n-1} f\left(\frac{\theta-\theta_i}{h}\right)\right)} \geq \frac{\left(f'\left(\frac{\theta-\theta_0}{h}\right)\right)^2}{nf(0)}$$

and because by change of variable,

$$\int_{-\infty}^{\theta_0} \left(f'\left(\frac{\theta-\theta_0}{h}\right)\right)^2 = \int_{-\infty}^{0} hf'(u)^2 du$$

we have our lower bound:

$$A \geq \frac{h}{nf(0)} \int_{-\infty}^{0} f'(u)^2 du$$

Putting together these two bounds, we proved that:

$$\frac{1}{n^2 f(0) h^2} \int_{-\infty}^{0} f'(u)^2 du \leq \mathcal{I}_{prior} \leq \frac{1}{h^2} \int_{-\infty}^{\infty} \frac{f'(u)^2}{f(u)} du$$

and hence that $\mathcal{I}_{prior}$ depends on h, and that it is asymptotically a $\Theta\left(\frac{1}{h^2}\right)$, meaning that when h is close to zero the information goes to ∞ and when h goes to ∞ the information goes to 0.

□

B. Algorithm of section 4.2.2.2

Algorithm 1 $O(n\log(n))$ comparison of pairs

Input: x a vector $[x_0, \ldots, x_{n-1}]$
Input: y a vector $[y_0, \ldots, y_{n-1}]$
x' = $[(x_0,'X'), \ldots, (x_{n-1},'X')]$
y' = $[(y_0,'Y'), \ldots, (y_{n-1},'Y')]$
v = MERGE(x',y')
v = SORTONFIRSTELEMENT(v)
count_x_passed = 0
count_y_passed = 0
both_ok = 0
x_greater = 0
y_greater = 0
for (s,origin) in v **do**
  if origin = 'X' **then**
    count_x_passed += 1
    if s $\leq \tau$ **then**
      both_ok += count_y_passed ▷ *All the elements of y passed are also* $\leq \tau$
    else
      x_greater += count_y_passed ▷ *All the elements of y passed are less than the current element of x*
    end if
  else ▷ *origin = 'Y'*
    count_y_passed += 1
    if s $\leq \tau$ **then**
      both_ok += count_x_passed
    else
      y_greater += count_x_passed
    end if
  end if
end for
Output: x_greater, both_ok, y_greater

C. Evolutionary relevance of ratiometric quorum sensing in *E. faecalis*

This appendix presents works that was realized during my PhD but on a merely loosely related topic. It is published in Banderas, 2020 [10].

C.1. Ratiometric quorum sensing governs the horizontal transfer of the *pCF10* plasmid in *Enterococcus faecalis*

As opposed to us, bacteria do not only carry DNA in their chromosomes. They also have small DNA rings called plasmids, that carry a handful of genes only, and are capable of autonomous replication. Apart from their size, plasmids have another difference with chromosomes, they can be horizontally transferred: a *donor* that carries several copies of a given plasmid can conjugate with a *recipient* to transfer it one of its copies. In this work, we were interested in a plasmid called *pCF10* found in the bacteria *Enterococcus faecalis*. *pCF10* provides *E. faecalis* with two additional capabilities: antibiotic resistance and mating. As many mating plasmid, *pCF10* comes with sensing machinery, allowing a given bacteria to detect other around them, identify whether they cary the plasmid or not, and initiate mating.

Because mating inherently consists in a tradeoff, it is beneficial to the cell to only turn it on when appropriate. Indeed, mating redirects resources from cell division and slows the latter down. In an environment where donors and recipients are competing for resources, it is hence crucial for donors to only mate when it would spread the plasmid

more efficiently than dividing. (note: It may seem counter intuitive that mates "compete" for resources but horizontal transfer of plasmids is ambivalent. It can be both seen as a form of "sexual" exchange that we would expect to be cooperative, as the donor's plasmid gives antibiotic resistance to the recipient, or as an infection, where donors contaminate recipients by transferring the plasmid to them, burdening them (whether activated or not) and hindering their growth.)

In the *pCF10* plasmid, sensing can detect the amount of recipients, which can be intuitively interpreted as donors sensing the amount of potential mates before trying to initiate conjugation (if that number is high enough). Intriguingly however, on top of the positive feedback of the recipients quantity on donors activation, *pCF10* also implements a negative feedback of the donors quantity on the donors activation. My collaborator Alvaro Banderas did various characterization experiments and has shown that mating activation is actually conditioned to the ratio of donors to recipients, independently of the total population size.

My work on this project was to model the evolution of a population consisting of donors and recipients, and compare in silico the actual ratiometric sensing strategy to other possible ones.

C.2. In silico comparison of the ratiometric quorum sensing strategy to others

Our model consisted of two coupled ordinary differential equation. Donor population – denoted d – is governed by three processes: first, an irreversible second-order mass-action–like process that depicts horizontal plasmid transfer (conjugation), transforming recipients r to donors with a rate constant λ_{conj}, weighted by a function $h(r, d)$, which takes values between 0 and 1 and whose form depends on the strategy analyzed; second, a logistic growth law (i.e., vertical transfer), with maximal rate λ_d limited by the carrying capacity K and hindered by the cost of mating activation up to a constant c, as estimated from our experiments and third, an imposed constant dilution rate (μ).

$$d' = \lambda_{conj} \cdot h(r,d) \cdot d \cdot r + \left(\lambda_d \cdot (1 - c \cdot h(r,d)) \cdot \left(1 - \frac{r+d}{K} \right) - \mu \right) \cdot d$$

Similarly, recipients grow logistically with a maximal rate λ_r (superior to λ_d) and are removed from the pool by dilution and upon conjugation.

$$r' = -\lambda_{conj}\dot{h}(r,d)\cdot d\cdot r + \left(\lambda_r\cdot\left(1-\frac{r+d}{K}\right)-\mu\right)\cdot r$$

The different possibilities for $h(r,d)$ (the "mating strategy") are:

1. Constitutive activation: $h(r,d)$ is a constant,
2. Recipient-sensing: $h(r,d)$ is a sigmoidal, Hill-type function of the recipient density,
3. Population-sensing: $h(r,d)$ is a Hill function of the total density, and
4. Ratio-sensing: $h(r,d)$ is a Hill function of the recipients to donor ratio.

To compare the different strategies, we considered them from the evolutionary point of view, and asked ourselves why would ratio-sensing have evolved over more simple strategies? To answer this question, I studied the steady state repartition of donors and recipients across varying carrying capacities, to represent more or less favorable growth conditions.

Of the four strategies – each tried with different values (for the constant strategy) or thresholds (for the sensing strategies), **ratio-sensing is the only one that preserves a nearly constant donors to recipients ratio across several orders of magnitude of carrying capacity** (see Figure C.1). This stabilization of the ratio can benefit the cells by ensuring that a sub-part of the population keeps the antibiotic-resistant plasmid while ensuring that the population as a whole is not outgrown by cells that would not be burdened by that plasmid. From a more donor-centered perspective, my simulations show that the ratio-sensing strategy is also the one that **maximizes the absolute number of donors across several orders of magnitude of carrying capacity**: maintaining recipients as a substrate for spread seems more efficient than simply taking over and relying on a comparatively slower vertical spread.

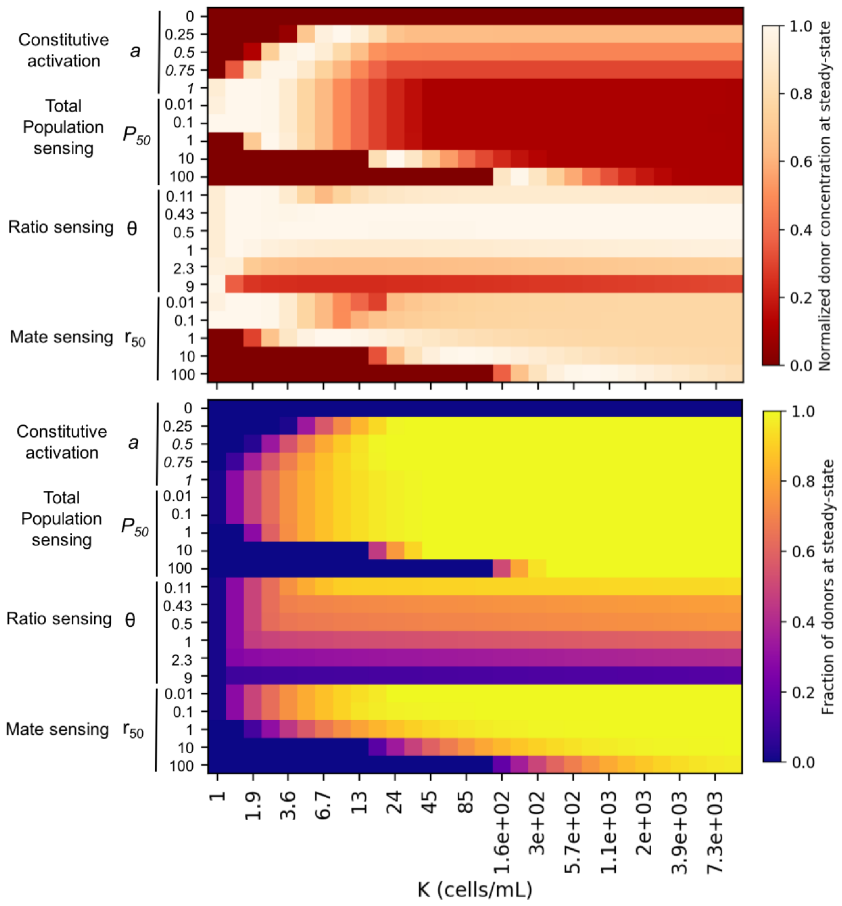


Figure C.1.: Strategy comparison showing simulations varying relevant parameter values for each strategy: activation level for the constitutive activation strategy and activation threshold for the rest. **Top:** Absolute donor concentration in the media at steady state (each value is normalized by the maximum value observed within each K). **Bottom:** Fraction of donors at steady state. In both cases; the ratio-sensing strategy is the one that both maximizes the absolute donor concentration and maintains a constant ratio across the largest range of values of carrying capacity K.

Bibliography

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: https://www.tensorflow.org/.

[2] Joel Andersson, Johan Åkesson, and Moritz Diehl. "CasADI: A Symbolic Package for Automatic Differentiation and Optimal Control". In: *Recent Advances in Algorithmic Differentiation*. Ed. by Shaun Forth et al. Vol. 87. Lecture Notes in Computational Science and Engineering. Berlin: Springer, 2012, pp. 297–307. ISBN: 978-3-540-68935-5. DOI: 10.1007/978-3-642-30023-3âĆĆ7.

[3] Virgile Andreani. "Modelling and Efficient Characterization of Enzyme-Mediated Response to Antibiotic Treatments". PhD thesis. Ecole polytechnique, Dec. 17, 2020. URL: https://tel.archives-ouvertes.fr/tel-03161857.

[4] Virgile Andreani et al. "A Model-Based Approach to Characterize Enzyme-Mediated Response to Antibiotic Treatments: Going beyond the SIR Classification". In: (July 17, 2021), p. 2021.07.16.452741. DOI: 10.1101/2021.07.16.452741. URL: https://www.biorxiv.org/content/10.1101/2021.07.16.452741v1.

[5] Titu Andreescu and Bogdan Enescu. "1.2 Cauchy–Schwarz Revisited". In: *Mathematical Olympiad Treasures*. Boston, MA: Birkhäuser Boston, 2012, pp. 7–9. ISBN: 978-0-8176-8253-8. DOI: 10.1007/978-0-8176-8253-8. URL: http://link.springer.com/10.1007/978-0-8176-8253-8.

[6] Eva Balsa-Canto, Maria Rodriguez-Fernandez, and Julio R. Banga. "Optimal Design of Dynamic Experiments for Improved Estimation of Kinetic Parameters of Thermal Degradation". In: *Journal of Food Engineering* 82.2 (Sept. 2007), pp. 178–188. ISSN: 02608774. DOI: 10.1016/j.jfoodeng.2007.02.006. URL: https://linkinghub.elsevier.com/retrieve/pii/S026087740700088X.

[7] Eva Balsa-Canto et al. “AMIGO2, a Toolbox for Dynamic Modeling, Optimization and Control in Systems Biology”. In: *Bioinformatics* 32.21 (Nov. 1, 2016), pp. 3357–3359. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw411. URL: https://doi.org/10.1093/bioinformatics/btw411.

[8] Eva Balsa-Canto et al. “Hybrid Optimization Method with General Switching Strategy for Parameter Estimation”. In: *BMC Systems Biology* 2.1 (Mar. 24, 2008), p. 26. ISSN: 1752-0509. DOI: 10.1186/1752-0509-2-26. URL: https://doi.org/10.1186/1752-0509-2-26.

[9] Samuel Bandara et al. “Optimal Experimental Design for Parameter Estimation of a Cell Signaling Model”. In: *PLOS Computational Biology* 5.11 (Nov. 6, 2009), e1000558. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1000558. URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000558.

[10] Alvaro Banderas et al. “Ratiometric Quorum Sensing Governs the Trade-off between Bacterial Vertical and Horizontal Antibiotic Resistance Propagation”. In: *PLOS Biology* 18.8 (Aug. 14, 2020), e3000814. ISSN: 1545-7885. DOI: 10.1371/journal.pbio.3000814. URL: https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.3000814.

[11] L. Bandiera et al. “Optimally Designed vs Intuition-Driven Inputs: The Study Case of Promoter Activity Modelling”. In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018 IEEE Conference on Decision and Control (CDC). Dec. 2018, pp. 1880–1885. DOI: 10.1109/CDC.2018.8618920.

[12] Lucia Bandiera et al. “On-Line Optimal Input Design Increases the Efficiency and Accuracy of the Modelling of an Inducible Synthetic Promoter”. In: *Processes* 6.9 (9 Sept. 2018), p. 148. DOI: 10.3390/pr6090148. URL: https://www.mdpi.com/2227-9717/6/9/148.

[13] Lucia Bandiera et al. “Optimally Designed Model Selection for Synthetic Biology”. In: *ACS Synthetic Biology* 9.11 (Nov. 20, 2020), pp. 3134–3144. DOI: 10.1021/acssynbio.0c00393. URL: https://doi.org/10.1021/acssynbio.0c00393.

[14] Julio R. Banga and Eva Balsa-Canto. “Parameter Estimation and Optimal Experimental Design”. In: *Essays in Biochemistry* 45 (Sept. 30, 2008). Ed. by Olaf Wolkenhauer, Peter Wellstead, and Kwang-Hyun Cho, pp. 195–210. ISSN: 0071-1365, 1744-1358. DOI: 10.1042/bse0450195. URL: https://portlandpress.com/essaysbiochem/article/doi/10.1042/bse0450195/78346/Parameter-estimation-and-optimal-experimental.

[15] Irene Bauer et al. “Numerical Methods for Initial Value Problems and Derivative Generation for DAE Models with Application to Optimum Experimental Design of Chemical Processes”. In: *Scientific Computing in Chemical Engineering II*. Vol. 2. Springer-Verlag, Heidelberg, 1999, pp. 282–289.

[16] Irene Bauer et al. “Numerical Methods for Optimum Experimental Design in DAE Systems”. In: *Journal of Computational and Applied Mathematics* 120.1 (Aug. 1, 2000), pp. 1–25. ISSN: 0377-0427. DOI: 10.1016/S0377-0427(00)00300-9. URL: https://www.sciencedirect.com/science/article/pii/S0377042700003009.

[17] Daniel Bernoulli and Sally Blower. “An Attempt at a New Analysis of the Mortality Caused by Smallpox and of the Advantages of Inoculation to Prevent It”. In: *Reviews in Medical Virology* 14.5 (2004), pp. 275–288. ISSN: 1099-1654. DOI: 10.1002/rmv.443. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rmv.443.

[18] Michael Betancourt. *A Conceptual Introduction to Hamiltonian Monte Carlo*. July 15, 2018. arXiv: 1701.02434 [stat]. URL: http://arxiv.org/abs/1701.02434.

[19] Christian Bischof et al. “ADIFOR–Generating Derivative Codes from Fortran Programs”. In: *Scientific Programming* 1.1 (1992), pp. 11–29. ISSN: 1058-9244. DOI: 10.1155/1992/717832. URL: https://www.hindawi.com/journals/sp/1992/717832/.

[20] Adrian Bürger. *Adbuerger/Casiopeia*. Oct. 4, 2021. URL: https://github.com/adbuerger/casiopeia.

[21] Alberto Giovanni Busetto et al. “Near-Optimal Experimental Design for Model Selection in Systems Biology”. In: *Bioinformatics* 29.20 (Oct. 15, 2013), pp. 2625–2632. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btt436. URL: https://doi.org/10.1093/bioinformatics/btt436.

[22] Arthur Carcano. *Cvode-Wrap*. Version 0.1.3. June 14, 2021. URL: https://crates.io/crates/cvode-wrap.

[23] Nessa Carson. *96-Well Plate*. May 29, 2020. URL: https://commons.wikimedia.org/wiki/File:96-Well_plate.svg.

[24] Oana-Teodora Chis, Julio R. Banga, and Eva Balsa-Canto. "Structural Identifiability of Systems Biology Models: A Critical Comparison of Methods". In: *PLOS ONE* 6.11 (Nov. 22, 2011), e27755. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0027755. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0027755.

[25] Paolo Di Tommaso et al. "Nextflow Enables Reproducible Computational Workflows". In: *Nature biotechnology* 35.4 (2017), pp. 316–319.

[26] Bill Dietrich. "Future Perfect – Thanks To Bill Gates' $12-Million Endowment, Scientist Leroy Hood Continues His Search For A New Genetic Destiny". In: *The Seattle Tiles* (Feb. 9, 1992). URL: https://archive.seattletimes.com/archive/?date=19920209&slug=1474735.

[27] Klaus Dietz and J. A. P. Heesterbeek. "Daniel Bernoulli's Epidemiological Model Revisited". In: *Mathematical Biosciences* 180.1 (Nov. 1, 2002), pp. 1–21. ISSN: 0025-5564. DOI: 10.1016/S0025-5564(02)00122-0. URL: https://www.sciencedirect.com/science/article/pii/S0025556402001220.

[28] Jose A. Egea et al. "Dynamic Optimization of Nonlinear Processes with an Enhanced Scatter Search Method". In: *Industrial & Engineering Chemistry Research* 48.9 (May 6, 2009), pp. 4388–4401. ISSN: 0888-5885. DOI: 10.1021/ie801717t. URL: https://doi.org/10.1021/ie801717t.

[29] Jose A. Egea et al. "MEIGO: An Open-Source Software Suite Based on Metaheuristics for Global Optimization in Systems Biology and Bioinformatics". In: *BMC Bioinformatics* 15.1 (May 10, 2014), p. 136. ISSN: 1471-2105. DOI: 10.1186/1471-2105-15-136. URL: https://doi.org/10.1186/1471-2105-15-136.

[30] R. P. Elander. "Industrial Production of Beta-Lactam Antibiotics". In: *Applied Microbiology and Biotechnology* 61.5-6 (June 2003),

pp. 385–392. ISSN: 0175-7598. DOI: 10.1007/s00253-003-1274-y. pmid: 12679848.

[31] G. Elfving. “Optimum Allocation in Linear Regression Theory”. In: *The Annals of Mathematical Statistics* 23.2 (June 1952), pp. 255–262. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177729442. URL: https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-23/issue-2/Optimum-Allocation-in-Linear-Regression-Theory/10.1214/aoms/1177729442.full.

[32] Ronald Aylmer Fisher. “Design of Experiments”. In: *British medical journal* 1.3923 (1936), pp. 554–554.

[33] Zachary R. Fox and Brian Munsky. “The Finite State Projection Based Fisher Information Matrix Approach to Estimate Information and Optimize Single-Cell Experiments”. In: *PLoS computational biology* 15.1 (Jan. 2019), e1006365. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1006365. pmid: 30645589.

[34] Zachary R. Fox, Gregor Neuert, and Brian Munsky. “Optimal Design of Single-Cell Experiments within Temporally Fluctuating Environments”. In: *Complexity* 2020 (June 13, 2020), e8536365. ISSN: 1076-2787. DOI: 10.1155/2020/8536365. URL: https://www.hindawi.com/journals/complexity/2020/8536365/.

[35] Gaia Franceschini and Sandro Macchietto. “Model-Based Design of Experiments for Parameter Precision: State of the Art”. In: *Chemical Engineering Science*. Model-Based Experimental Analysis 63.19 (Oct. 1, 2008), pp. 4846–4872. ISSN: 0009-2509. DOI: 10.1016/j.ces.2007.11.034. URL: https://www.sciencedirect.com/science/article/pii/S0009250907008871.

[36] Joseph Diaz Gergonne. “Analise. Application de la méthode des moindres quarrés à l’interpolation des suites”. In: *Annales de mathématiques pures et appliquées* 6 (1815–1816), pp. 242–252. URL: http://www.numdam.org/item/AMPA_1815-1816__6__242_0/.

[37] Joseph Diaz Gergonne. “The Application of the Method of Least Squares to the Interpolation of Sequences”. In: *Historia Mathematica* 1.4 (Nov. 1974), pp. 439–447. ISSN: 03150860. DOI: 10.1016/0315-0860(74)90034-2. URL: https://linkinghub.elsevier.com/retrieve/pii/0315086074900342.

[38] Richard D. Gill and Boris Y. Levit. “Applications of the van Trees Inequality: A Bayesian Cramér-Rao Bound”. In: *Bernoulli* 1.1-2 (Mar. 1995), pp. 59–79. ISSN: 1350-7265. URL: https://projecteuclid.org/euclid.bj/1186078362.

[39] Camille Gontier and Jean-Pascal Pfister. “Identifiability of a Binomial Synapse”. In: *Frontiers in Computational Neuroscience* 14 (2020), p. 86. ISSN: 1662-5188. DOI: 10.3389/fncom.2020.558477. URL: https://www.frontiersin.org/article/10.3389/fncom.2020.558477.

[40] Daniel Goujot, Xuan Mi Meyer, and Francis Courtois. “Identification of a Rice Drying Model with an Improved Sequential Optimal Design of Experiments”. In: *Journal of Process Control* vol. 22.1 (Jan. 2012), pp. 95–107. DOI: 10.1016/j.jprocont.2011.10.003. URL: https://hal.archives-ouvertes.fr/hal-00879520.

[41] Andreas Griewank. “Who Invented the Reverse Mode of Differentiation?” In: *Documenta Mathematica* (2012), p. 12.

[42] Peter Guttorp and Georg Lindgren. “Karl Pearson and the Scandinavian School of Statistics”. In: *International Statistical Review* 77.1 (2009), pp. 64–71. ISSN: 1751-5823. DOI: 10.1111/j.1751-5823.2009.00069.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1751-5823.2009.00069.x.

[43] Nikolaus Hansen. *The CMA Evolution Strategy: A Tutorial*. Apr. 4, 2016. arXiv: 1604.00772 [cs, stat]. URL: http://arxiv.org/abs/1604.00772.

[44] Nikolaus Hansen et al. *CMA-ES/Pycma: R3.1.0*. Zenodo, June 20, 2021. DOI: 10.5281/zenodo.5002422. URL: https://zenodo.org/record/5002422.

[45] Alan C Hindmarsh et al. “2.6 Mathematical Considerations > Forward Sensitivity Analysis”. In: *User Documentation for Cvodes v5. 7.0 (Sundials v5. 7.0)*. 2021. URL: https://computing.llnl.gov/sites/default/files/cvs_guide-5.7.0.pdf.

[46] Alan C Hindmarsh et al. “SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers”. In: *ACM Transactions on Mathematical Software (TOMS)* 31.3 (2005), pp. 363–396.

[47] Matthew D. Hoffman and Andrew Gelman. *The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo*. Nov. 17, 2011. arXiv: 1111.4246 [cs, stat]. URL: http://arxiv.org/abs/1111.4246.

[48] Philipp H. W. Hoffmann. "A Hitchhiker's Guide to Automatic Differentiation". In: *Numerical Algorithms* 72.3 (July 2016), pp. 775–811. ISSN: 1017-1398, 1572-9265. DOI: 10.1007/s11075-015-0067-6. URL: http://link.springer.com/10.1007/s11075-015-0067-6.

[49] Trey Ideker, Timothy Galitski, and Leroy Hood. "A New Approach to Decoding Life: Systems Biology". In: *Annual Review of Genomics and Human Genetics* 2.1 (2001), pp. 343–372. DOI: 10.1146/annurev.genom.2.1.343. pmid: 11701654. URL: https://doi.org/10.1146/annurev.genom.2.1.343.

[50] Morris A. Jette, Andy B. Yoo, and Mark Grondona. "SLURM: Simple Linux Utility for Resource Management". In: *In Lecture Notes in Computer Science: Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP) 2003*. Springer-Verlag, 2002, pp. 44–60.

[51] *JuliaDiff*. URL: https://juliadiff.org/.

[52] P. E. Jupp. "A van Trees Inequality for Estimators on Manifolds". In: *Journal of Multivariate Analysis* 101.8 (Sept. 1, 2010), pp. 1814–1825. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2010.03.007. URL: https://www.sciencedirect.com/science/article/pii/S0047259X10000667.

[53] Arthur L. Koch. "Some Calculations on the Turbidity of Mitochondria and Bacteria". In: *Biochimica et Biophysica Acta* 51.3 (Aug. 19, 1961), pp. 429–441. ISSN: 0006-3002. DOI: 10.1016/0006-3002(61)90599-6. URL: https://www.sciencedirect.com/science/article/pii/0006300261905996.

[54] Michał Komorowski et al. "Sensitivity, Robustness, and Identifiability in Stochastic Chemical Kinetics Models". In: *Proceedings of the National Academy of Sciences* 108.21 (May 24, 2011), pp. 8645–8650. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1015814108. pmid: 21551095. URL: https://www.pnas.org/content/108/21/8645.

[55] Maxim Kryukov et al. “Can Optimal Experimental Design Serve as a Tool to Characterize Highly Non-Linear Synthetic Circuits?” In: *2019 18th European Control Conference (ECC)*. 2019 18th European Control Conference (ECC). June 2019, pp. 1176–1181. DOI: 10.23919/ECC.2019.8796209.

[56] Feldman Bell Laboratories and S. I. Feldman. “Make — A Program for Maintaining Computer Programs”. In: 9 (1979), pp. 255–265.

[57] Abraham Lee. *Ad: Fast, Transparent First- and Second-Order Automatic Differentiation*. Version 1.3.2. URL: http://pythonhosted.org/ad.

[58] Juliane Liepe et al. “Maximizing the Information Content of Experiments in Systems Biology”. In: *PLOS Computational Biology* 9.1 (Jan. 31, 2013), e1002888. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1002888. URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002888.

[59] Seppo Linnainmaa. “Taylor Expansion of the Accumulated Rounding Error”. In: *BIT* 16.2 (June 1976), pp. 146–160. ISSN: 0006-3835, 1572-9125. DOI: 10.1007/BF01931367. URL: http://link.springer.com/10.1007/BF01931367.

[60] Seppo Linnainmaa. “The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors.” MA thesis. 1970.

[61] Yingbo Ma et al. *A Comparison of Automatic Differentiation and Continuous Sensitivity Analysis for Derivatives of Differential Equation Solutions*. July 20, 2021. arXiv: 1812.01892 [cs]. URL: http://arxiv.org/abs/1812.01892.

[62] R. Mehra. “Optimal Input Signals for Parameter Estimation in Dynamic Systems–Survey and New Results”. In: *IEEE Transactions on Automatic Control* 19.6 (Dec. 1974), pp. 753–768. ISSN: 1558-2523. DOI: 10.1109/TAC.1974.1100701.

[63] Hannah R. Meredith et al. “Applying Ecological Resistance and Resilience to Dissect Bacterial Antibiotic Responses”. In: *Science Advances* (Dec. 2018). URL: https://www.science.org/doi/abs/10.1126/sciadv.aau1873.

[64] Hongyu Miao et al. “On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics”. In: *SIAM Review* 53.1 (Jan. 1, 2011), pp. 3–39. ISSN: 0036-1445. DOI: 10.1137/090757009. URL: https://epubs.siam.org/doi/10.1137/090757009.

[65] Felix Mölder et al. “Sustainable Data Analysis with Snakemake”. In: *F1000Research* 10 (2021).

[66] Christoph Molnar. *Interpretable Machine Learning*. Lulu.com, 2020. 320 pp. ISBN: 978-0-244-76852-2. Google Books: jBm3DwAAQBAJ.

[67] R. Muñoz-Tamayo et al. “IDEAS: A Parameter Identification Toolbox with Symbolic Analysis of Uncertainty and Its Application to Biological Modelling”. In: *IFAC Proceedings Volumes* 42.10 (2009), pp. 1271–1276. ISSN: 14746670. DOI: 10.3182/20090706-3-FR-2004.00211. URL: https://linkinghub.elsevier.com/retrieve/pii/S1474667016388255.

[68] GM Ostrovski, Yu.M. Volin, and W.W. Borisov. “Über Die Berechnung von Ableitungen”. In: (1971).

[69] Dan Padilha et al. “Modern Numerical Programming with Julia for Astrodynamic Trajectory Design”. In: Feb. 3, 2021.

[70] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[71] Edouard Pauwels, Christian Lajaunie, and Jean-Philippe Vert. “A Bayesian Active Learning Strategy for Sequential Experimental Design in Systems Biology”. In: *BMC Systems Biology* 8.1 (Dec. 1, 2014), p. 102. ISSN: 1752-0509. DOI: 10.1186/s12918-014-0102-6. URL: https://doi.org/10.1186/s12918-014-0102-6.

[72] E. S. PEARSON. “Studies in the History of Probability and Statistics. XX: Some Early Correspondence between W. S. Gosset, R. A. Fisher and Karl Pearson, with Notes and Comments”. In: *Biometrika* 55.3 (Nov. 1, 1968), pp. 445–457. ISSN: 0006-3444. DOI: 10.1093/biomet/55.3.445. URL: https://doi.org/10.1093/biomet/55.3.445.

[73] Stuart J. Pocock and Richard Simon. “Sequential Treatment Assignment with Balancing for Prognostic Factors in the Controlled Clinical Trial”. In: *Biometrics* 31.1 (1975), pp. 103–115. ISSN: 0006-341X. DOI: 10.2307/2529712. JSTOR: 2529712.

[74] Christopher Rackauckas. *A Comparison Between Differential Equation Solver Suites In MATLAB, R, Julia, Python, C, Mathematica, Maple, and Fortran*. Stochastic Lifestyle, A Random Blog About Math and Life. Sept. 26, 2017. URL: http://www.stochasticlifestyle.com/comparison-differential-equation-solver-suites-matlab-r-julia-python-c-fortran/.

[75] A. Raue et al. “Data2Dynamics: A Modeling Environment Tailored to Parameter Estimation in Dynamical Systems”. In: *Bioinformatics* 31.21 (Nov. 1, 2015), pp. 3558–3560. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btv405. URL: https://doi.org/10.1093/bioinformatics/btv405.

[76] A. Raue et al. “Structural and Practical Identifiability Analysis of Partially Observed Dynamical Models by Exploiting the Profile Likelihood”. In: *Bioinformatics* 25.15 (Aug. 1, 2009), pp. 1923–1929. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btp358. URL: https://doi.org/10.1093/bioinformatics/btp358.

[77] Andreas Raue et al. “Lessons Learned from Quantitative Dynamical Modeling in Systems Biology”. In: *PLOS ONE* 8.9 (Sept. 30, 2013), e74335. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0074335. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0074335.

[78] Federico Reali, Corrado Priami, and Luca Marchetti. “Optimization Algorithms for Computational Systems Biology”. In: *Frontiers in Applied Mathematics and Statistics* 3 (2017), p. 6. ISSN: 2297-4687. DOI: 10.3389/fams.2017.00006. URL: https://www.frontiersin.org/article/10.3389/fams.2017.00006.

[79] Rami Al-Rfou et al. “Theano: A Python Framework for Fast Computation of Mathematical Expressions”. In: *arXiv e-prints* abs/1605.02688 (May 2016). URL: http://arxiv.org/abs/1605.02688.

[80] Maria Rodriguez-Fernandez, Pedro Mendes, and Julio R. Banga. "A Hybrid Approach for Efficient and Robust Parameter Estimation in Biochemical Pathways". In: *Biosystems* 83.2-3 (Feb. 2006), pp. 248–265. ISSN: 03032647. DOI: 10.1016/j.biosystems.2005.06.016. URL: https://linkinghub.elsevier.com/retrieve/pii/S0303264705001334.

[81] Thomas J. Rothenberg. "Identification in Parametric Models". In: *Econometrica* 39.3 (1971), pp. 577–591. ISSN: 0012-9682. DOI: 10.2307/1913267. JSTOR: 1913267.

[82] Jakob Ruess, Andreas Milias-Argeitis, and John Lygeros. "Designing Experiments to Understand the Variability in Biochemical Reaction Networks". In: *Journal of The Royal Society Interface* 10.88 (Nov. 6, 2013), p. 20130588. ISSN: 1742-5689, 1742-5662. DOI: 10.1098/rsif.2013.0588. URL: https://royalsocietypublishing.org/doi/10.1098/rsif.2013.0588.

[83] Jakob Ruess et al. "Iterative Experiment Design Guides the Characterization of a Light-Inducible Gene Expression Circuit". In: *Proceedings of the National Academy of Sciences* 112.26 (June 30, 2015), pp. 8148–8153. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1423947112. pmid: 26085136. URL: https://www.pnas.org/content/112/26/8148.

[84] Caitríona M. Ryan, Christopher C. Drovandi, and Anthony N. Pettitt. "Optimal Bayesian Experimental Design for Models with Intractable Likelihoods Using Indirect Inference Applied to Biological Process Models". In: *Bayesian Analysis* 11.3 (Sept. 2016), pp. 857–883. ISSN: 1936-0975, 1931-6690. DOI: 10.1214/15-BA977. URL: https://projecteuclid.org/journals/bayesian-analysis/volume-11/issue-3/Optimal-Bayesian-Experimental-Design-for-Models-with-Intractable-Likelihoods-Using/10.1214/15-BA977.full.

[85] Maria Pia Saccomani and Karl Thomaseth. "The Union between Structural and Practical Identifiability Makes Strength in Reducing Oncological Model Complexity: A Case Study". In: *Complexity* 2018 (Feb. 11, 2018), e2380650. ISSN: 1076-2787. DOI: 10.1155/2018/2380650. URL: https://www.hindawi.com/journals/complexity/2018/2380650/.

[86] Richard P. Savage. “The Paradox of Nontransitive Dice”. In: *The American Mathematical Monthly* 101.5 (1994), pp. 429–436. ISSN: 0002-9890. DOI: 10.2307/2974903. JSTOR: 2974903.

[87] Kirstine Smith. “On the Standard Deviations of Adjusted and Interpolated Values of an Observed Polynomial Function and Its Constants and the Guidance They Give Towards a Proper Choice of the Distribution of Observations”. In: *Biometrika* 12.1/2 (1918), pp. 1–85. ISSN: 0006-3444. DOI: 10.2307/2331929. JSTOR: 2331929.

[88] Mission Spares. *Surveillance de la consommation des antibiotiques et des résistances bactériennes en établissement de santé. Résultats préliminaires 2019.* URL: https://www.santepubliquefrance.fr/import/surveillance-de-la-consommation-des-antibiotiques-et-des-resistances-bacteriennes-en-etablissement-de-sante.-mission-spares.-resultats-preliminaire.

[89] Stan Developpment Team. *Stan Math Library*. URL: //mc-stan.org/users/interfaces/math.

[90] Stephen M Stigler. “Gergonne’s 1815 Paper on the Design and Analysis of Polynomial Regression Experiments”. In: *Historia Mathematica* 1.4 (Nov. 1974), pp. 431–439. ISSN: 03150860. DOI: 10.1016/0315-0860(74)90033-0. URL: https://linkinghub.elsevier.com/retrieve/pii/0315086074900330.

[91] Anne Penfold Street. *Combinatorics of Experimental Design*. Oxford, New York: Clarendon Press, 1987. xiv, 400. ISBN: 978-0-19-853256-9.

[92] Stan Development Team. *Stan Modeling Language*. 2021. URL: https://mc-stan.org/.

[93] Tina Toni and Michael P. H. Stumpf. *Parameter Inference and Model Selection in Signaling Pathway Models*. May 27, 2009. arXiv: 0905.4468 [q-bio]. URL: http://arxiv.org/abs/0905.4468.

[94] *UCLA at Washington Box Score, November 10, 1990*. College Football at Sports-Reference.com. URL: https://www.sports-reference.com/cfb/boxscores/1990-11-10-washington.html.

[95] J. Vanlier et al. “A Bayesian Approach to Targeted Experiment Design”. In: *Bioinformatics* 28.8 (Apr. 15, 2012), pp. 1136–1142. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bts092. URL: https://doi.org/10.1093/bioinformatics/bts092.

[96] Karina J.E. Versyck and Jan F.M. Van Impe. “Robustness of the Iterative Optimal Experiment Design Scheme for Parameter Estimation: Identification of Microbial Heat Resistance Parameters”. In: *IFAC Proceedings Volumes* 34.5 (June 2001), pp. 31–36. ISSN: 14746670. DOI: 10.1016/S1474-6670(17)34191-5. URL: https://linkinghub.elsevier.com/retrieve/pii/S1474667017341915.

[97] Alejandro F Villaverde et al. “Benchmarking Optimization Methods for Parameter Estimation in Large Kinetic Models”. In: *Bioinformatics* 35.5 (Mar. 1, 2019), pp. 830–838. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty736. URL: https://doi.org/10.1093/bioinformatics/bty736.

[98] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17.3 (Mar. 2, 2020), pp. 261–272. ISSN: 1548-7091, 1548-7105. DOI: 10.1038/s41592-019-0686-2. URL: http://www.nature.com/articles/s41592-019-0686-2.

[99] A. Walther and A. Griewank. “Getting Started with ADOL-C”. In: *Combinatorial Scientific Computing*. Ed. by U. Naumann and O. Schenk. Chapman-Hall CRC Computational Science, 2012, pp. 181–202.

[100] R. E. Wengert. “A Simple Automatic Derivative Evaluation Program”. In: *Communications of the ACM* 7.8 (Aug. 1, 1964), pp. 463–464. ISSN: 0001-0782. DOI: 10.1145/355586.364791. URL: https://doi.org/10.1145/355586.364791.

[101] Christoph Zimmer. “Experimental Design for Stochastic Models of Nonlinear Signaling Pathways Using an Interval-Wise Linear Noise Approximation and State Estimation”. In: *PLOS ONE* 11.9 (Sept. 1, 2016), e0159902. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0159902. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0159902.