

# Developing a neural network-based pipeline to improve Gene Regulatory Network (GRN) inference over the state-of-the-art algorithms

A thesis  
submitted towards the partial fulfilment of the  
BS-MS dual degree programme  
by

PAARTH DUDANI



27TH MARCH 2024

under the guidance of

DR. RUBEN PEREZ-CARRASCO

GROUP LEADER, LECTURER,  
THEORETICAL SYSTEMS BIOLOGY GROUP,  
DEPARTMENT OF LIFE SCIENCES,  
IMPERIAL COLLEGE LONDON

from May 2024 to March 2025

INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH PUNE

This thesis is dedicated to my sister, Pankhuri, and our eternal bond..



# Certificate

This is to certify that this dissertation entitled ‘Developing neural network algorithms to improve Gene Regulatory Network (GRN) inference over the state-of-the-art algorithms’ towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Paarth Dudani at Imperial College London under the supervision of Dr. Ruben Perez-Carrasco, Department of life sciences, during the academic year 2024-2025.

Supervisor:

Ruben Perez-Carrasco

Group leader, lecturer,

Theoretical Systems

Biology Group,

Department of Life

Sciences

Imperial College London

Paarth Dudani

20201113

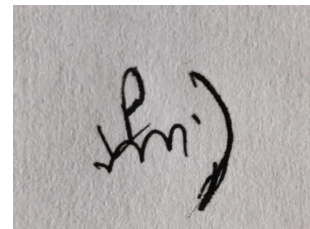
BS-MS

Student

IISER PUNE



Supervisor: Ruben Perez-Carrasco



# Declaration

I hereby declare that the matter embodied in this report titled “Developing a neural network-based pipeline to improve Gene Regulatory Network (GRN) inference over the state-of-the-art algorithms” is the result of investigations carried out by me at the Imperial College London, UK from the period 15-06-2024 to 27-03-2025 under the supervision of Dr. Ruben Perez-Carrasco and the same has not been submitted elsewhere for any other degree.

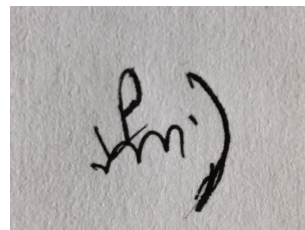
Supervisor:

Ruben Perez-Carrasco  
Group leader, lecturer,  
Theoretical Systems  
Biology Group,  
Department of Life  
Sciences  
Imperial College London

Paarth Dudani  
20201113  
BS-MS  
Student  
IISER PUNE



Supervisor: Ruben Perez-Carrasco



# Acknowledgements

Firstly, I would like to acknowledge my supervisor, Ruben Perez-Carrasco at Imperial College, for his continuous guidance, invaluable support, and insightful mentorship throughout this project. I am equally grateful to my postdoctoral supervisor, Daniel O'Hanlon, who has guided me at several obstacles, and whose insights often changed the course of the project. I would also like to thank another post doc researcher in the lab, Smitha for her valuable inputs and constructive feedback, greatly enriching my work. I extend my appreciation to all other lab members for the insightful discussions, support, and encouragement during my time in the lab. I sincerely thank my expert, Leelavati, from my institute, for kindly agreeing to participate in the evaluation of my thesis. Lastly, I want to acknowledge my sister, Pankhuri, whose unwavering and relentless support provided me with strength and courage during the challenging times spent at Imperial College and in London.

# Contributions

| Contributor name     | Contributor role             |
|----------------------|------------------------------|
| Ruben Perez-Carrasco | Supervision                  |
| Ruben Perez-Carrasco | Ideation                     |
| Daniel O'Hanlon      | Supervision                  |
| Daniel O'Hanlon      | Ideation                     |
| Paarth Dudani        | Literature Review            |
| Paarth Dudani        | Method Development           |
| Paarth Dudani        | Writing - review and editing |

# Abstract

Gene regulatory networks (GRNs) entail complex and nonlinear interactions which are captured by systems of ordinary differential equations (ODEs). Repressilator, the first artificial GRN to be engineered, is taken as the GRN of this study. For the repressilator, different parameter values for the ODEs lead to vastly different dynamics. This is tackled by inferring a posterior probability distribution as a part of Bayesian inference, incorporating prior beliefs and observed data. The state-of-the-art algorithms for this are slow, owing to the high dimensionality of the models. Moreover, they need to be re-run whenever new observations are available, keeping the average inference time per observed dataset high. A faster alternative with acceptable accuracy is explored via a neural network-based inference pipeline. Results from this pipeline demonstrate a reduction in the average inference time, or amortisation, below that obtained from the current approaches. This holds promise for a deeper understanding of natural GRNs, while enabling robust engineering of artificial ones.

# Contents

|          |                                                                                                                              |           |
|----------|------------------------------------------------------------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                                                                                          | <b>4</b>  |
| 1.1      | Understanding GRNs elucidates biology . . . . .                                                                              | 4         |
| 1.2      | The repressilator and parameter inference . . . . .                                                                          | 5         |
| 1.3      | Simulator models and inverse problems . . . . .                                                                              | 7         |
| 1.4      | Simulation-based inference: State-of-the-art and neural algorithms . . .                                                     | 8         |
| 1.5      | The Goal and the outline . . . . .                                                                                           | 9         |
| <b>2</b> | <b>Study hypothesis and objectives</b>                                                                                       | <b>12</b> |
| <b>3</b> | <b>Concepts and methods</b>                                                                                                  | <b>13</b> |
| 3.1      | Bayesian inference . . . . .                                                                                                 | 13        |
| 3.2      | Simulator models and likelihoods . . . . .                                                                                   | 14        |
| 3.2.1    | Additive noise: Direct injection . . . . .                                                                                   | 14        |
| 3.2.2    | Multiplicative noise: Injection in the <i>log</i> space . . . . .                                                            | 16        |
| 3.3      | Markov chain Monte Carlo (MCMC) . . . . .                                                                                    | 16        |
| 3.4      | Neural network architectures for SBI . . . . .                                                                               | 18        |
| 3.4.1    | Mixture density network (MDN) . . . . .                                                                                      | 18        |
| 3.4.2    | Masked autoregressive flows (MAF) . . . . .                                                                                  | 18        |
| 3.4.3    | Masked autoencoder for distribution estimation (MADE) . . . .                                                                | 18        |
| 3.5      | Hyperparameter optimisation . . . . .                                                                                        | 18        |
| 3.6      | Sequential neural estimation (SNE) . . . . .                                                                                 | 19        |
| 3.6.1    | Sequential neural posterior estimation (SNPE) . . . . .                                                                      | 20        |
| 3.6.2    | Other approaches: SNLE and SNRE (likelihood and ratio estimation) . . . . .                                                  | 20        |
| 3.6.3    | NPE: An alternative to SNPE . . . . .                                                                                        | 20        |
| 3.7      | Packages and implementation . . . . .                                                                                        | 21        |
| 3.7.1    | pymc for ODE Bayesian Inference . . . . .                                                                                    | 21        |
| 3.7.2    | sbi Package and Software Implementation . . . . .                                                                            | 22        |
| <b>4</b> | <b>Results and Interpretation</b>                                                                                            | <b>23</b> |
| 4.1      | Test model: Exponential decay . . . . .                                                                                      | 23        |
| 4.2      | Exponential decay: One-dimensional case . . . . .                                                                            | 24        |
| 4.2.1    | SNPE and NPE yield reasonable estimates with all three architectures . . . . .                                               | 26        |
| 4.2.2    | SNLE and NLE (enabling amortisation) yield reasonable estimates with all three architectures . . . . .                       | 28        |
| 4.2.3    | SNRE and NRE (enabling amortisation) yield reasonable estimates with both the architectures, but with lesser flexibility . . | 28        |
| 4.2.4    | Comparison of scores for SNPE, SNLE and SNRE . . . . .                                                                       | 30        |
| 4.3      | Test Model: Two-dimensional case . . . . .                                                                                   | 31        |

|          |                                                                           |           |
|----------|---------------------------------------------------------------------------|-----------|
| 4.4      | Repressilator Model . . . . .                                             | 38        |
| 4.5      | Repressilator Model: Three-dimensional version . . . . .                  | 39        |
| 4.6      | Repressilator Model: Six-dimensional model . . . . .                      | 43        |
| 4.7      | Repressilator Model: Nine-dimensional model . . . . .                     | 49        |
| 4.8      | Repressilator Model: Twelve-dimensional version . . . . .                 | 53        |
| 4.9      | Repressilator Model: 15-dimensional version . . . . .                     | 57        |
| 4.10     | Inference time comparison across models . . . . .                         | 60        |
| <b>5</b> | <b>Discussion</b>                                                         | <b>61</b> |
| 5.1      | Conclusions . . . . .                                                     | 61        |
| 5.2      | Limitations . . . . .                                                     | 61        |
| 5.3      | Future work . . . . .                                                     | 62        |
| 5.4      | Scope . . . . .                                                           | 63        |
| <b>6</b> | <b>Appendix</b>                                                           | <b>69</b> |
| 6.0.1    | Autocorrelation plots . . . . .                                           | 69        |
| 6.1      | Supplementary results: Repressilator model (Three-Dimensional) . . . . .  | 69        |
| 6.2      | Supplementary results: Repressilator model (Six-dimensional) . . . . .    | 71        |
| 6.3      | Supplementary results: Repressilator model (Twelve-Dimensional) . . . . . | 72        |
| 6.4      | Supplementary results: Repressilator model (15-Dimensional) . . . . .     | 73        |

# Chapter 1

## Introduction

### 1.1 Understanding GRNs elucidates biology

Complexity in biological cells arises, not just from the functions performed by the elements (proteins, mRNAs, etc.), but from their complex, non-linear interactions (Kittano, 2002). While the philosophy of reductionism has enabled their thorough characterization, biological functions such as replication or signal transduction are rarely attributable to individual elements alone (Barabasi and Oltvai, 2004). This has cultivated a systems perspective for the functioning of cells. Herein, they are proposed to be comprised of functional ‘modules’ linked in a network of interactions; a key idea in systems biology (Alon, 2019). A major class of these interaction networks are **Gene regulatory networks (GRNs)**. These are comprised of a group of genes which encode proteins that function as transcription factors. Within a GRN, these transcription factors can either enhance the expression of another gene (activators) or inhibit it (repressors). The specific configuration of interactions between the components of a GRN, constitute its **topology** (Figure 1.1). Naturally, whether a protein functions as an activator or a repressor depends on the context of the GRN and can change across different GRNs. From such interactions, GRNs regulate and coordinates numerous cellular functions in response to intracellular and extracellular cues, enabling cells dynamically adapt to their environments. Examples include driving cell differentiation into distinct cell types (Goode et al., 2016) (Cahan et al., 2014), leveraging stochasticities as an adaptive mechanism (Mangan and Alon, 2003), underlying patterning during embryonic development (López-Maury et al., 2008) etc.

Due to non-linear interactions among components, the emergent dynamics of a GRN can be unintuitive and challenging to describe qualitatively (Guo et al., 2016). More interestingly, even with the same topology, a GRN is capable of exhibiting different emergent dynamics or ‘behaviours’/‘functions’ (see next section). In natural systems, this behaviour is executed as a function by a GRN mediating cell survival. In artificial systems, this behaviour is a desired function to be executed by a GRN by engineering it into cells (Elowitz and Leibler, 2000).

With the growing use of mathematical tools in biology (Hunter, 2010), this realisation has led to two broad modeling approaches (Karlach and Shamir, 2008):

1. Logical models, which binarise gene activation states (Thomas, 1973; Kauffman et al., 2003), and



2. Continuous models that represent activation or repression using real-valued variables (Klipp et al., 2005)

Within continuous models, using systems of **Ordinary differential equations (ODEs)** is a well-established methodology for describing non-linear dynamics of GRNs (Chen et al., 2004). Briefly, a system of ODEs describes the rate of change of the variables of interest as a function of a set of parameters of the system. For this study, the variables are protein and mRNA concentrations and the parameters are biologically meaningful quantities such as degradation rates etc. A solution to this ODE describes how the concentration of a protein or mRNA evolves with time (time-series concentration trajectories).

## 1.2 The repressilator and parameter inference

Since the late 20th century, the complexity of biological cells has been attributed to the interactions between functional modules. A proposed approach to elucidate this complexity is to “design and build new ones” (Hartwell et al., 1999), echoing Feynman’s, “What I cannot create, I do not understand” (Caltech Archives). The design and implementation of the **repressilator** (Elowitz and Leibler, 2000) exemplified this philosophy, ushering in a new era in synthetic biology (Benner and Sismour, 2005). Since the early 2000s, the repressilator model has served as a widely studied framework for GRNs (Pušnik et al., 2019). Briefly, the repressilator consists of three genes, each encoding a repressor protein that inhibits the expression of the next gene, forming a repressive feedback loop. The following figure schematises this:

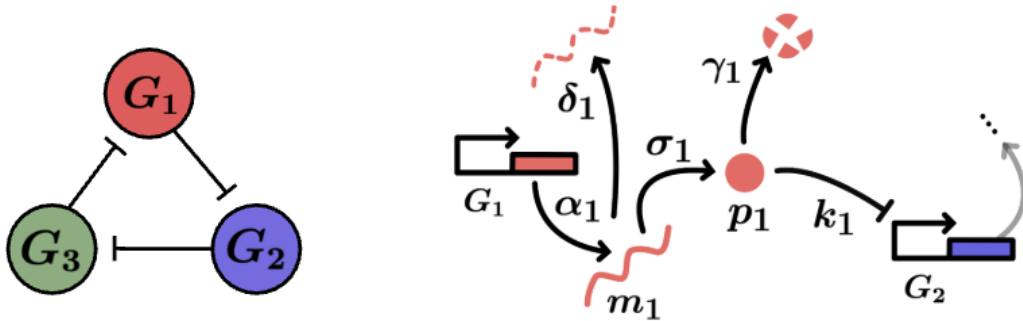


Figure 1.1: Schematic representation of the topology of the repressilator (left) and underlying biological processes (right).

The underlying ODE system captures the concentration evolution of the three mRNA species  $m_i$  and their corresponding proteins  $p_i$  through the following system:

$$\frac{dm_i}{dt} = \frac{\alpha_i}{1 + (k_i \cdot p_j)^{n_i}} + \chi_i - \delta_i \cdot m_i, \quad (1.1)$$

$$\frac{dp_i}{dt} = \sigma_i \cdot m_i - \gamma_i \cdot p_i. \quad (1.2)$$

In writing down this model, assumptions such as applicability of Hill kinetics to cooperativity and constancy of the parameters are made (no stochastic processes). All the parameters have biological significance and are collectively denoted by  $\boldsymbol{\theta} = \{\alpha_1, \alpha_2, \dots, \gamma_2, \gamma_3\}$   $\forall i \in \{1, 2, 3\}$  (bold for vector representation). These are:

1. Leaky transcription rates ( $\chi_i$ ),
2. Repressor binding affinities ( $k_i$ ),
3. Hill coefficients ( $n_i$ ),
4. mRNA translation rates ( $\sigma_i$ ),
5. mRNA degradation rates ( $\delta_i$ ),
6. Protein degradation rates ( $\sigma_i$ ) and
7. Maximum increment in expression, over leaky expression (under no repression) ( $\alpha_i$ )

A solution to this system of coupled ODE equations results in the time-series evolution of concentrations of each of the six species. Given a set of parameter values ( $\theta$ ), it is rather a trivial matter to numerically integrate and solve this ODE system. However, even slightly different sets of parameter values result in the solutions existing in vastly different dynamical regimes. The following shows an example of how merely changing the protein translation rates ( $\{\sigma_1, \sigma_2, \sigma_3\}$ ) by a small amount, keep all other parameters constant, can lead to vastly different dynamics:

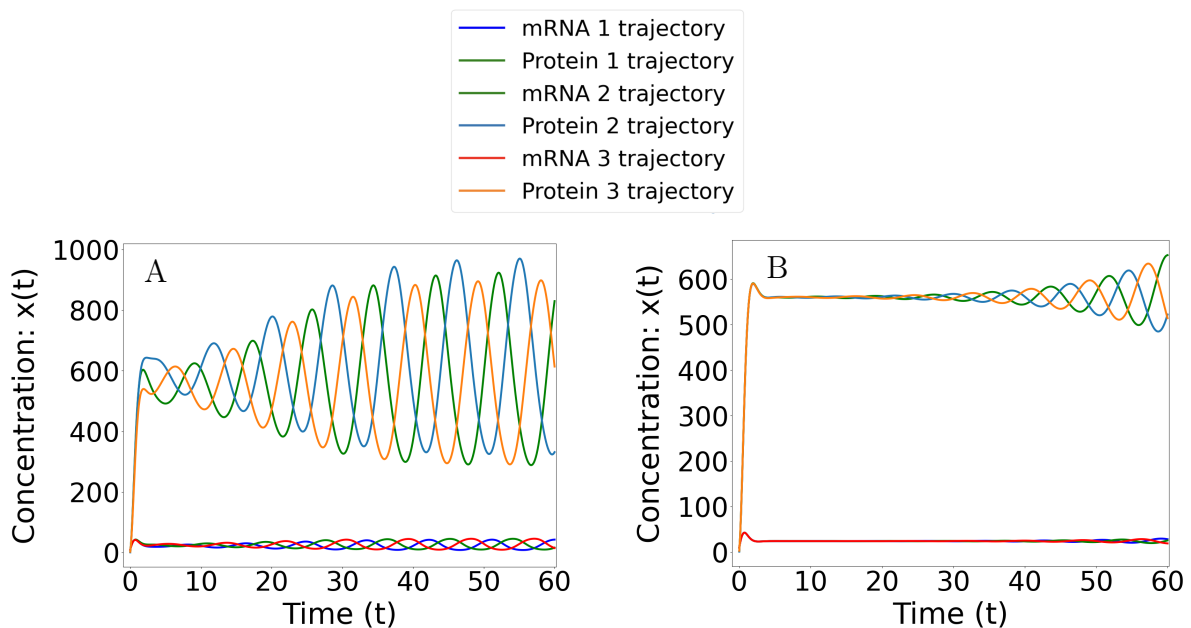


Figure 1.2: ODE solutions for all species in corresponding to marginally different set of translation rates, all else is kept constant. **A**  $\{\sigma_1, \sigma_2, \sigma_3\} = \{19.0, 18.5, 17.0\}$ . **B**  $\{\sigma_1, \sigma_2, \sigma_3\} = \{18.0, 18.0, 18.0\}$

Thus, knowing biologically meaningful domains of parameter values for this system of ODEs is essential as enables prediction of the dynamics or behaviour of the repressilator (Elowitz and Leibler, 2000).

Mathematical analyses (e.g., stability analysis or optimisation) can offer basic ranges for the parameters of the repressilator (Pušnik et al., 2019). Although such analyses predict the qualitative behaviour of the repressilator, the domain of such behaviour can

be large, yielding broad ranges. Furthermore, these are purely theoretical and are not informed by experimental data, precluding a deeper understanding of the system. In contrast, a parameter inference approach, informed by experimental data while quantifying uncertainty around the biochemical parameters is proposed to be an improvement. Such an approach could enable a deeper understanding of natural GRNs, while enabling robust engineering of artificial ones (refer to results and discussion). In this context, the Bayesian paradigm provides a principled framework for parameter inference (Kruschke, 2014). The prior over the parameters and the likelihood obtained using observed data, together, yield the joint posterior probability distribution (**joint posterior**, in short) over the full parameter space. In essence, the joint posterior encodes all the information about the system of interest, given our assumptions and data (Sec. 3.1). While point estimates of the parameters can be readily extracted (Mean, variance etc.), the joint posterior can inform a broad array of inferences. Examples include highest posterior density (HPD) estimation, estimating correlations between parameters (Kruschke, 2014) etc. This is of significance to the fields of synthetic and systems biology (sec. 5.4).

## 1.3 Simulator models and inverse problems

Domains of science are routinely concerned with inverse problems (Aster et al., 2018). This encapsulates either devising a model or inferring the parameter values of a hypothesised model, to explain the observed data for the phenomenon of interest. With this in mind, an attempt is made to treat parameter inference for the repressilator model as an inverse problem.

In order for a phenomenon to be understood in practice, it must first be understood in principle. This philosophy guides the establishment of a proof of concept as the approach of this study. This is done as follows (Fig. 1.5):

1. From rudimentary parameter ranges from the literature (Pušnik et al., 2019), a viable set of ‘true’ parameter values is chosen.
2. These are used to solve the given system of ODEs, followed by injection of additive Gaussian noise, generating the ‘observed’ dataset. Injection of the noise makes the ‘observed’ dataset mimic experimental data, and maps the repressilator model to a simulator model (Lintusaari et al., 2017) (sec. 3).
3. Putting this synthetically generated ‘observed’ dataset into a ‘conceptual black-box’, an inference pipeline is then devised to infer a joint posterior probability distribution over the parameters.
4. The high dimensionality of the problem is tackled by starting with a toy model and proceeding onto simplified versions of the repressilator model, performing the inference from ground up (see 4).

The following figure shows an illustration of the ‘observed’ dataset obtained by injection of the Gaussian noise to the ODE solution corresponding to the parameter configuration:

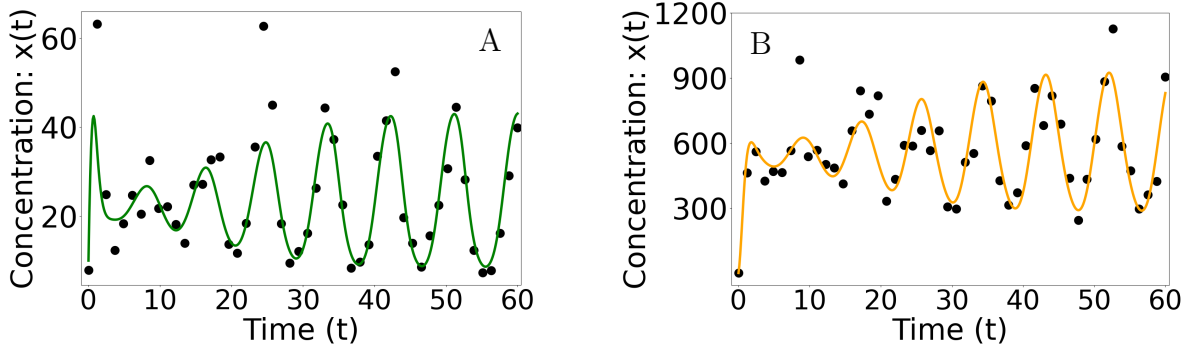


Figure 1.3: An example of ‘observed’ data used in the inference pipeline. **A** shows the ‘observed’ dataset for the mRNA 1. **B** shows the same for protein 1.

## 1.4 Simulation-based inference: State-of-the-art and neural algorithms

Parameter inference for simulator models is a rather well-established problem (Lintusaari et al., 2017). Briefly, the likelihood for simulator models is typically intractable (Sec. 3.2), hence naive Bayesian inference is not applicable. This is the problem of Likelihood-Free Inference or, more aptly, **simulation based inference (SBI)** (Cranmer et al., 2020). For such problems, **approximate Bayesian computation (ABC)** algorithms approximate the likelihood function and estimate the joint posterior over the parameters (Sisson et al., 2018; Blum, 2010; Toni et al., 2009; Beaumont et al., 2002). These constitute the state-of-the-art algorithms for SBI problems, and were used in the previous work (students (2024)).

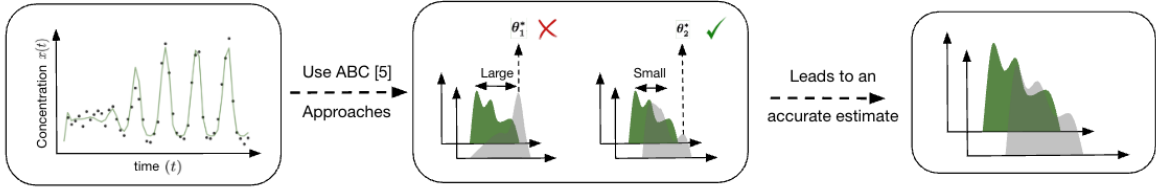
In this context, however, there are three key ideas concerning this study.

1. ABC algorithms, even asymptotically, do not approximate the ‘true’ joint posterior distribution (ground truth) over the parameters (Papamakarios and Murray, 2016).
2. Secondly, these algorithms become very slow for high dimensional problems (Li et al., 2017), such as the repressilator model.
3. For the repressilator model with additive Gaussian noise, its likelihood can indeed be numerically approximated (sec. 3.2). Thus, more conventional methods such as **Markov chain Monte Carlo (MCMC)** are applicable (Brooks et al., 2011).

However, as shown in the results section, inference using MCMC also grows exponentially slower (similar to ABC), in higher dimensions due to the curse of dimensionality (Li et al., 2017). Since the precise values of the parameters is not as important as the uncertainty quantification, one considers faster alternatives to infer the joint posterior, albeit with lesser accuracy. In this context, neural networks come to mind.

Towards the end of the 20th century, the advent of enormous data, compute power and representation learning (Bengio et al., 2013), brought new life to machine learning in the form of deep learning (Goodfellow et al., 2016; LeCun et al., 2015). This has occurred concomitantly with the establishment of multi-layered feedforward networks as universal approximators (Hornik et al., 1989; Nasrabadi, 2007). As part of the frenzy of throwing neural networks at a diverse set of problems (LeCun et al., 2015; Krizhevsky et al., 2012; Hinton et al., 2012; Helmstaedter et al., 2013; Leung et al., 2014), they have also been extended to SBI problems (Cranmer et al., 2020). Specifically, **sequential neural estimation (SNE)** involves algorithms approximating different inference objects related to the Bayes’ theorem (posterior, likelihood etc.) using various neural network architectures (sec. 3.4) (Papamakarios and Murray, 2016; Greenberg et al., 2019; Papamakarios et al., 2019; Hermans et al., 2020). Using these inference objects, they approximate the joint posterior distribution over the parameters. Although these algorithms were explored in the previous work, it suffered from the lack of a ground truth joint posterior distribution (**ground truth**, in short) over the parameters. This makes their results, although remarkable, somewhat untrustworthy (students (2024)). The following figure schematises the comparison between the conventional and proposed approaches:

**State-of-the-art approach:** Accurate, but slow posterior estimation



**Approach of this study:** Aim: fast and as accurate as possible (refer to the top)

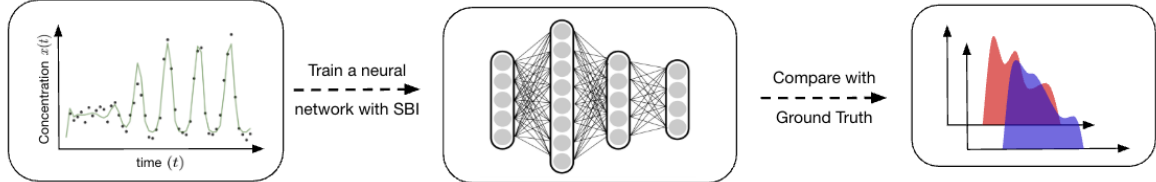


Figure 1.4: Comparison of conventional and proposed approaches

## 1.5 The Goal and the outline

Building on work from Perez-Carrasco’s lab, this study aims to develop a **fast and accurate inference pipeline** for inferring the joint posterior over the parameters of the repressilator model, using neural network-based algorithms. As a proof of concept, ‘observed’ data are synthetically generated under known priors, with the subsequent estimation of the joint posterior with neural approaches. This joint posterior estimate is compared with the ground truth obtained from MCMC (sec. 3.3) and diagnostic checks are performed, alongside inference time comparisons. This is schematised in figure 1.5.

Following is the outline for the remainder this study:

1. **Study hypothesis and objectives (Chapter 2):** Lays out the primary and auxiliary hypothesis for this study along with the broad objectives,
2. **Concepts and methods (Chapter 3):** Lays out key concepts, methods, and configurations used for the experiments and simulations.
3. **Results (Chapter 4):** Presents findings for two systems: a simple exponential decay model and the simplified versions of the repressilator.
4. **Discussion (Chapter 5):** Interprets the results, addresses limitations, and outlines the future work and scope.
5. **Appendix (Chapter 6):** Lays out supplementary results.

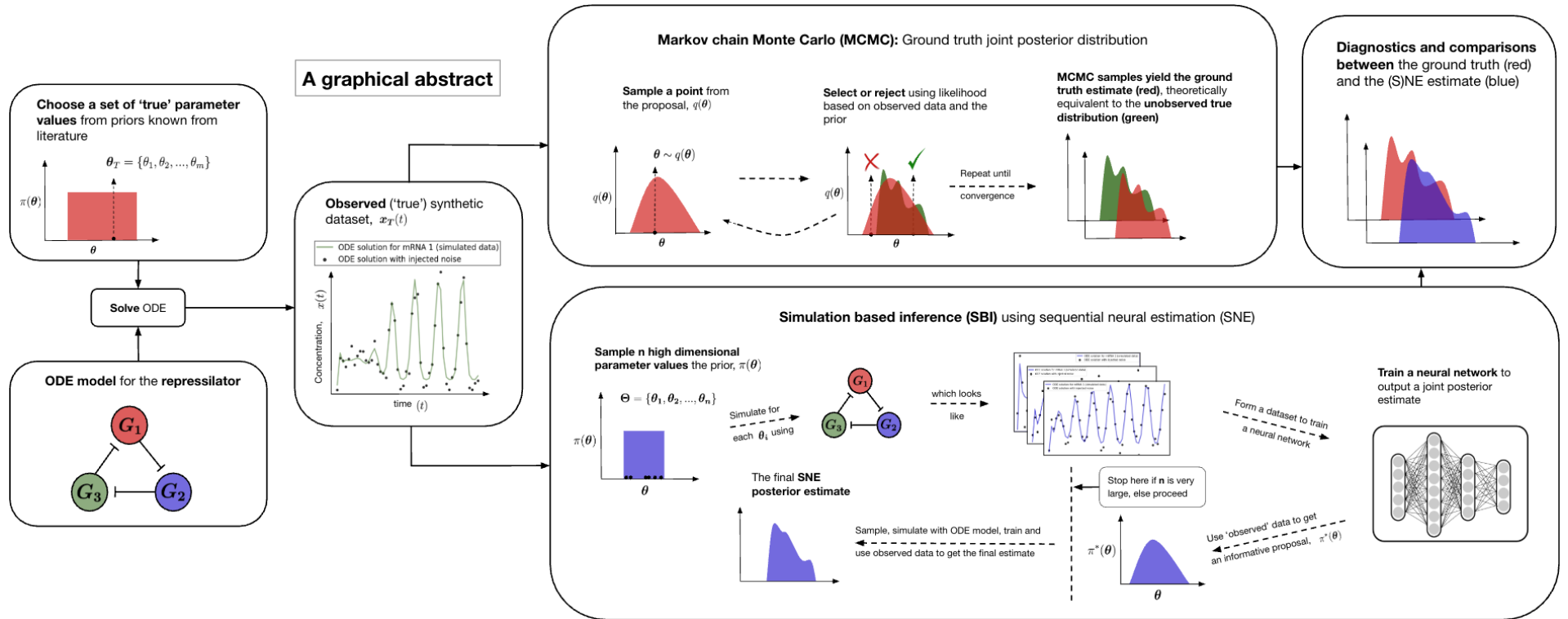


Figure 1.5: A graphical abstract showing schematising the inference pipeline. The synthetically generated ‘observed’ dataset shown above, mimics experimental data and is fed into an MCMC algorithm yielding the ground truth joint posterior distribution over the parameters. While being theoretically accurate, it takes exponentially longer for convergence for models with many parameters. SNE, then, provides a fast way to obtain joint posterior distributions. Through comparisons with the ground truth and several diagnostic checks, robustness of estimation and accuracy of predictions are ensured.

# Chapter 2

## Study hypothesis and objectives

The primary hypothesis of this study is that neural network-based algorithms (**neural algorithms**, in short), can be employed for parameter inference for the repressilator model, in a manner that is accurate and **faster** than the state-of-the-art algorithms (MCMC). This hypothesis is based on the assumption that neural network-based algorithms mitigate the curse of dimensionality better than MCMC algorithms and scale well with dimensions. Phrased differently, the ratio of inference times for joint posteriors obtained from the neural network-based algorithms and MCMC should be less than unity (in short, **ratio of the inference times**). However, there are two alternative neural network based approaches towards this end, sequential neural posterior estimation (SNPE) and neural posterior estimation NPE (Sec. 3.6 and 3.6.3). Specifically for testing inference using SNPE, an auxiliary hypothesis is also proposed as follows: While the ratio of the inference times may be greater than unity for simpler models, this ratio should approach unity in higher dimensions, eventually reaching below unity for models with higher complexity. This would be the regime where the primary hypothesis holds true. For NPE, the primary hypothesis will hold true if the trained neural network yields accurate posteriors for enough number of datasets (Sec 3.6.3 and 4). Thus, this informs the objectives of this study as follows:

1. Evaluate the joint posterior estimates from the neural network-based algorithms by comparisons with the ground truth using diagnostic checks and benchmarking (sec. 3.1),
2. Evaluate the potential of NPE yielding joint posterior estimates in a faster way (sec. 3.6.3) by reducing the average inference time across datasets (amortisation),
3. Evaluate the ratio of inference times under various scenarios for both SNPE and NPE and test the primary and auxiliary hypotheses.



# Chapter 3

## Concepts and methods

This section starts with brief explanations of the relevant concepts, some of which were mentioned in the introduction section. Subsequently, the implementation details of the packages used are briefly covered.

### 3.1 Bayesian inference

As mentioned in the previous section, Bayesian inference is a principled framework for parameter inference (Kruschke, 2014). There are two key objects in the Bayesian paradigm:

1. The prior probability distribution (**prior**, in short) over all the parameters ( $\boldsymbol{\theta}$ , bold to represent a vector), which encodes our initial beliefs about the parameter values.
2. The **likelihood**, which is the probability of observing data given a specific configuration of parameters and encodes the information contained in observed or simulated data (sec. 3.2).

The prior,  $p(\boldsymbol{\theta})$ , in conjunction with the likelihood,  $p(\mathbf{x} \mid \boldsymbol{\theta})$ , obtained using the ‘observed’ data,  $\mathbf{x}$ , is used to infer the joint posterior probability distribution (**joint posterior**, in short),  $p(\boldsymbol{\theta} \mid \mathbf{x})$ , over the full parameter space.

This is done in accordance with the Bayes’ theorem as follows:

$$p(\boldsymbol{\theta} \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta})}{p(\mathbf{x})}. \quad (3.1)$$

On a technical note,  $p(\mathbf{x})$  is referred to as the evidence, but is a constant with respect to the parameters. To check whether the model fit (the inferred joint posterior) is compatible with the data, posterior predictive checks (**PPC**) can be performed. This involves drawing samples of  $\boldsymbol{\theta}$  from the joint posterior, and for each  $\boldsymbol{\theta}$ , simulating new data  $\tilde{\mathbf{x}}$  from the data-generating process, and comparing the distribution of these simulated or predicted data to the ‘observed’ data,  $\mathbf{x}$  (Kruschke, 2014). In essence, PPCs serve as necessary conditions to be satisfied. If the model fit is adequate, the observed data should appear typical among the posterior predictive simulations and any deviations are easily noticeable. However, failure modes of PPCs exist (Lueckmann et al.,

2021), which is why using some kind of metric is critical to evaluate the joint posterior and validate the inference. This necessitates the need for benchmarking the obtained estimates (Sec. 4.2).

## 3.2 Simulator models and likelihoods

Simulator models have been used ubiquitously across domains to elucidate phenomena (Huang et al., 2023). Essentially, simulator models ( $M$ ) map the full space of parameters ( $\Theta$ ) and random variables ( $V$ ) (stochasticities) to the space of data ( $X$ ). This can be represented as follows (Lintusaari et al., 2017):

$$\Theta, V \xrightarrow{M} X \implies M(\theta, V) = X \quad (3.2)$$

While one can readily generate ‘simulated data’ from these models for any configuration of ( $\theta$ ), the likelihood for these models is intractable typically due to the presence latent variables (Cranmer et al., 2020). This precludes the conventional Bayesian inference procedure, motivating an entire field of inference with ABC methods (Sisson et al., 2018). However, for the repressilator model, the situation is different. The likelihood for the repressilator can be modeled as a product of Gaussian distributions, if we consider the following:

1. The system of ODEs as the simulator ( $M$ ) yielding time-series concentrations as solutions for a given parameter set ( $\theta$ ,  $\theta \in \Theta$ ),
2. Random variables ( $V$ ) taking the form of additive Gaussian noise injected into these ODE solutions. This can be done either by injecting additive Gaussian noise (sec. 3.2.1) or multiplicative Gaussian noise data (additive Gaussian in  $\log$  space, sec. 3.2.2). This yields simulated data ( $\tilde{x}$ ), reflecting epistemic uncertainty (Der Kiureghian and Ditlevsen, 2009), that is, uncertainty associated with measurement errors.

Doing this essentially maps the models to generalised linear models (Dobson and Barnett, 2018), for which the likelihoods are accessible.

### 3.2.1 Additive noise: Direct injection

Here, Gaussian noise is added to the concentration value at each time point ( $n$  time points in total) in the ODE solution. The following equations describe this:

$$\tilde{x}(t; \theta) = x_{ODE}(t; \theta) + \varepsilon(t), \quad \varepsilon(t) \sim \mathcal{N}(0, \sigma^2), \quad (\text{i.i.d}) \quad (3.3)$$

$$\implies \tilde{x}_i = x_{iODE} + \varepsilon_i, \quad \forall i \in \{1, \dots, n\} \quad (3.4)$$

Where,  $\tilde{x}(t)$  represents ‘simulated’ data (see below) obtained from the ODE solution  $x_{ODE}(t)$  for  $n$  time points, by injection with additive Gaussian noise ( $\varepsilon$ ). The Gaussian noise is injected through sampling independently and identically (i.i.d) from  $\mathcal{N}(0, \sigma^2)$  denoting a Gaussian distribution with mean 0 and variance  $\sigma^2$ .

1. For ‘true’ values of parameters,  $\theta$ ,  $\tilde{\mathbf{x}}(t)$  represents ‘**observed**’ data.
2. For sampled values of parameters,  $\theta$ ,  $\tilde{\mathbf{x}}(t)$  represents ‘**simulated**’ data used as a part of the simulation-based inference procedure (sec. 3.6).

With this as the set up, the likelihood can be described as follows:

$$L(\theta \mid \tilde{\mathbf{x}}) = p(\tilde{\mathbf{x}} \mid \theta) = \prod_{i=1}^n \mathcal{N}(\tilde{x}_i \mid x_{iODE}(\theta), \sigma^2), \quad (3.5)$$

Where,  $\tilde{\mathbf{x}} = \tilde{x}_1, \dots, \tilde{x}_n$  are concentration values at  $n$  time points, in the simulated data. The following figure visualises this process for the exponential model, a test model for this study (described in sec. 4.1) :

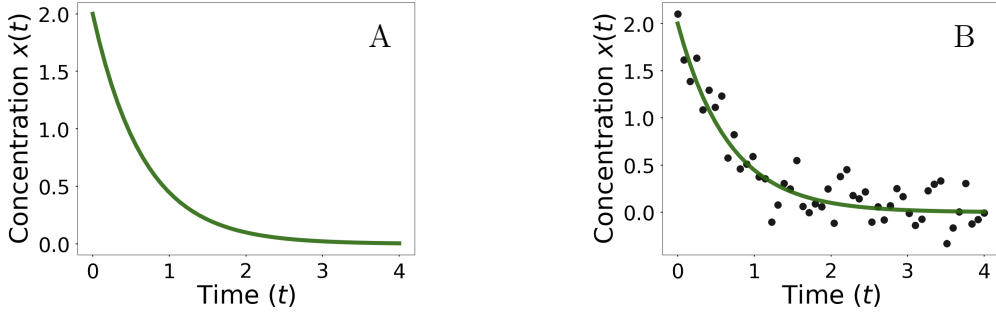


Figure 3.1: Exponential model: ODE solution and ‘observed’ data obtained using additive noise. **A** shows the ODE solution for decay rate,  $\beta = 1.5$ . **B** shows the result of injecting additive noise at 50 equally spaced time steps.

However, a potential pitfall is immediately obvious, that is, negative values of concentrations. While not a major concern for the exponential model (explained in sec. 4.1), this was a major limitation for the repressilator model in the previous work (students, 2024). Another limitation is uneven noise levels in the ‘observed’ dataset for mRNA and protein trajectories, for the same value of the standard deviation of the noise ( $\sigma$ ), as shown below:

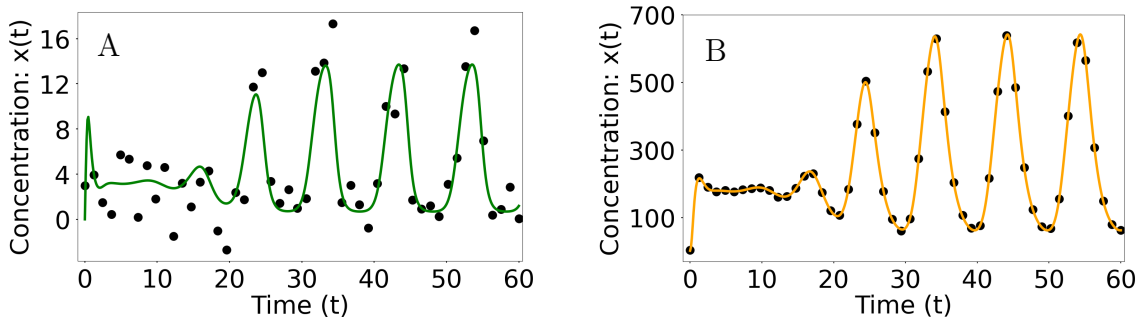


Figure 3.2: Repressilator model: ‘Observed’ data obtained using additive Gaussian noise. **A** shows the ODE solution, followed by injection of noise at 50 time steps, for mRNA 1. **B** shows the same but for protein 1.

Vastly different scales of mRNA and protein values in the ODE solutions explain this discrepancy. That is, adding the same level of noise at different scales of concentrations

is expected to lead to inconsistencies. However, such differences in scales are biologically significant and have to be incorporated into the inference procedure [ref]. This motivated the subsequent approach.

### 3.2.2 Multiplicative noise: Injection in the $\log$ space

Here, the data are first transformed and mapped to the logarithmic space. In the new space, Gaussian noise is added to the concentration (transformed) value at each time point in the ODE solution. The following equations describe this:

$$\mathbf{y}_{ODE}(t; \boldsymbol{\theta}) = \log(\mathbf{x}_{ODE}(t; \boldsymbol{\theta})), \quad (3.6)$$

$$\tilde{\mathbf{y}}(t; \boldsymbol{\theta}) = \mathbf{y}_{ODE}(t; \boldsymbol{\theta}) + \boldsymbol{\varepsilon}(t), \quad \boldsymbol{\varepsilon}(t) \sim \mathcal{N}(0, \sigma^2) \text{ (i.i.d)}, \quad (3.7)$$

$$\implies \tilde{y}_i = y_{iODE}(t; \boldsymbol{\theta}) + \varepsilon_i, \quad \forall i \in \{1, \dots, n\}, \quad (3.8)$$

$$\implies \tilde{x}_i = \exp(y_{iODE}(t; \boldsymbol{\theta})) \cdot \exp(\varepsilon_i), \quad \forall i \in \{1, \dots, n\}. \quad (3.9)$$

Where,  $\tilde{\mathbf{y}}(t)$  represents simulated data from the ODE solution  $\mathbf{x}_{ODE}(t)$  for  $n$  time points both in the  $\log$  space. In this space, the data are injected with Gaussian noise ( $\boldsymbol{\varepsilon}$ ). As before, the Gaussian noise is injected through sampling independently and identically (i.i.d) from  $\mathcal{N}(0, \sigma^2)$ . As before, the likelihood is the product of Gaussians as a function of the ODE solution injected with multiplicative noise. This yields the ‘simulated’ and ‘observed’ datasets and addresses the aforementioned limitations:

1. Multiplying the concentrations (all values in the ODE solutions are ensured to be positive) with exponential of the Gaussian noise guarantees positive values. This is the ‘multiplicative’ part of the noise.
2. The effect of multiplication is scale invariant, that is, the concentrations are modified in accordance with their magnitude. This evens out the effect of injected noise for both mRNA and protein trajectories.

This is elucidated by figure 1.3. This assumption provides a more accessible likelihood which encapsulates noise (mimicking experimental noise), making it also amenable to inference using Markov chain Monte Carlo (MCMC). This circumvents the need for using ABC and simplifies the implementation of the neural network based-SBI for inferring the joint posterior.

## 3.3 Markov chain Monte Carlo (MCMC)

MCMC algorithms are specifically deployed to infer a posterior distribution when the normalisation constant ( $p(\boldsymbol{x})$ ) in the Bayes’ theorem is unavailable (Brooks et al., 2011), making it relevant to this study (previous section). In the MCMC literature, such a distribution is called the **target distribution**. For this study, the ‘true’ joint posterior over the parameters is the target distribution and MCMC yields the **ground truth** estimate. The aim of any MCMC procedure is to generate independent samples from

the target distribution (Kruschke, 2014). Starting with a proposal distribution, samples generated sequentially. There are two well-established ways to do this (Neal et al., 2011; Hoffman et al., 2014; Chib and Greenberg, 1995):

1. Gradient-free methods: This uses a random walk sampling procedure. Naturally, many of these samples are far from the target distribution and therefore have to be rejected somehow. Metropolis-Hastings criterion (sec. 6) provides a principled way to do this. In the asymptotic limit, such a procedure of sampling and rejection is guaranteed to yield samples from the target distribution (Wasserman, 2013).
2. Gradient-based methods: These utilise gradients of the log-probabilities of the target distribution with respect to the model parameters to sequentially sample new points. This circumvents random walk associated limitations of the Metropolis-Hastings method. Examples include the Hamiltonian Monte Carlo (HMC) and its extension, the no-u-turn-sampler (NUTS).

Although, the gradient based methods are principled approaches for Bayesian inference, for this study, were found to be prohibitively slow and unwieldy. Hence, Metropolis-Hastings was the method of choice for ground truth inference and was implemented through the `pymc` package (sec. 4). The Gelman-Rubin (GR) statistic (Gelman and Rubin, 1996) and autocorrelation plots (Link and Eaton, 2012) were used to evaluate the ground truth estimate (sec. 6). In higher dimensions, the sampling in MCMC suffers from the ‘curse of dimensionality’ (Köppen, 2000), that is, as the number of model parameters grows, the parameter space volume expands exponentially. This means that the space available to explore, either by random walks or gradient based sampling increases exponentially. This makes it increasingly difficult to locate probability densities. This is the principle reason why the joint posterior inference becomes exponentially slower for these algorithms. The following figure illustrates this in practice for various parameterisations of the repressilator model:

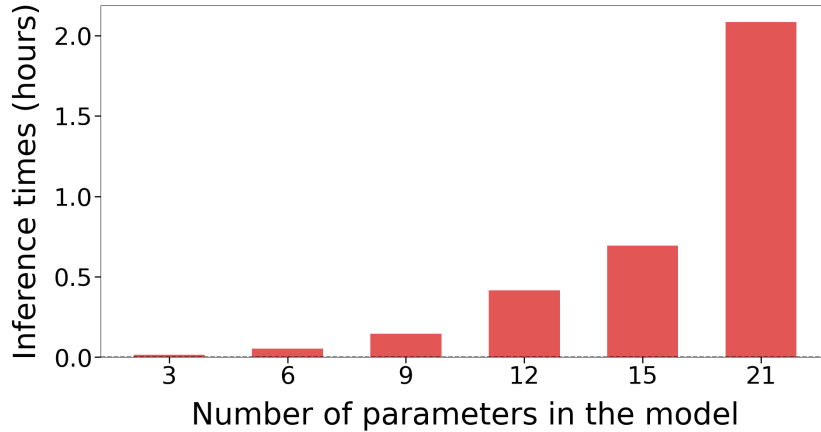


Figure 3.3: Inference times in hours using MCMC across various parameterisations (varying dimensionality) (sec. 4.4) of the repressilator model.

In this context, it is hypothesised that neural network-based algorithms can mitigate the curse of dimensionality and scale better than MCMC in higher dimensions, forming the basis of the primary and auxiliary hypothesis (sec. 2) of this study.

## 3.4 Neural network architectures for SBI

Multi-layered feedforward networks are universal approximators (Hornik et al., 1989). This explains their study and applications to a plethora domains over the past 30 years (Goodfellow et al., 2016; Gurney, 2018). In particular, several neural network architectures have been developed, building upon the original feedforward networks (LeCun et al., 1998; He et al., 2016; Medsker et al., 2001; Hochreiter and Schmidhuber, 1997; Cho et al., 2014). Several architectures are utilised as a part of the SBI pipeline for the exponential and the repressilator model, through the use of `sbi` package. These are described below:

### 3.4.1 Mixture density network (MDN)

In principle, MDNs can represent arbitrary distributions (Bishop, 1994), making them powerful approximators. These are neural networks which use a multi-layered feed-forward network (Hornik et al., 1989) to output parameters of a mixture or a linear combination of independent Gaussian distributions. This enables it to model multi-modal distributions, as a function of the weights and biases of the feedforward network which can be tuned via training using known gradient descent algorithms (Kingma and Ba, 2014).

### 3.4.2 Masked autoregressive flows (MAF)

MAFs are based on normalising flows which transform a simple, base probability distribution such as a Gaussian distribution, into a complex one via a series of invertible mappings (Trippe and Turner, 2018). Specifically, MAFs, combine the well-established auto-regressive models (Uria et al., 2016) with normalising flows and have been shown to estimate the target distribution for a variety of problems (Papamakarios et al., 2017). This makes MAF an interesting architecture to explore for this study.

### 3.4.3 Masked autoencoder for distribution estimation (MADE)

Finally, density estimation using autoregressive model estimation suffers issues in inference due to many possible orderings of conditional distribution in the product to model the joint distribution. A particular ordering has to be chosen prior to the training which may not be the most optimal option. MADE mitigates this by using an autoencoder (Pinheiro Cinelli et al., 2021) and masking specific connections such that each output variable’s prediction depends only on earlier input variables (in a specified ordering). This enforces a proper autoregressive order of the conditional distributions which can be learned through training (Germain et al., 2015), thus making MADE a viable candidate architecture for this study.

## 3.5 Hyperparameter optimisation

‘The question of “How good is this model on that dataset?” is ill-posed’ (Bergstra et al., 2013). Hyperparameters are variables of a neural network, which are not tuned during training with data (Probst et al., 2019). Hence, these are fixed before the training and can significantly affect the performance. Grid search is one of the simpler approaches

to tune hyperparameters (Yang and Shami, 2020), in which each hyperparameter configuration is used independently for training and the best performing configuration is chosen. The metric for best performance depend on the problem at hand. In this study, % error in the mean and standard deviation are considered (sec. 4.2).

For this study, the relevant hyperparameters are learning rate, ‘hidden features’ which reflects the size of architecture in terms of number of neurons, and the ‘number of components’ of a MDN or the number of transforms in a MAF/MADE. To implement grid search, triplets of hyperparameters are chosen within a specific range found through trial and error. Since, density estimation falls in the category of unsupervised learning (Nasrabadi, 2007) problems, choosing a performance metric is difficult. Since, the main objective is to compare the Ground Truth and the Neural network estimate of the posterior, differences of the means and standard deviations between them is considered as scores. Using this, the hyperparameter configuration with the lowest combined difference of mean and standard deviation for each parameter is chosen as the best configuration.

### 3.6 Sequential neural estimation (SNE)

Simulation-based inference (SBI) can be done non-sequentially (NPE) or sequentially (SNPE) (Cranmer et al., 2020). In particular, sequential neural estimation (SNE) infers a joint posterior by training sampling various parameter values ( $\tilde{\theta}$ ) from the prior and simulating corresponding data ( $\tilde{\mathbf{x}}$ ) using the simulator, over multiple rounds. The joint posterior is either directly estimated (SNPE) or inferred later on by estimating intermediate objects such as the likelihood (SNLE). Rudimentary estimates of the joint posterior obtained in every round, serves as an informed proposal for the next round, starting from the prior. This enables focused sampling of the parameter space and efficient estimation of the joint posterior (Papamakarios and Murray, 2016).

The general procedure is as follows:

1. Propose parameters from the proposal (prior in first round),  $\theta \sim \tilde{p}(\theta)$
2. Simulate data (implicitly from the likelihood)  $\mathbf{x} \sim p(\mathbf{x}|\theta)$
3. Minimise the loss as a function of the network weights conditioned on simulated data and infer the joint posterior  $P(\theta | \tilde{\mathbf{x}})$ , by conditioning the neural network on the ‘observed’ dataset.
4. The inferred joint posterior serves as the proposal for the next round of sampling and inference. This is repeated N times.

For the exponential model, three versions of SNE algorithms are considered (described below), but for the repressilator, only SNPE is used for inference.

Note: Hereafter, **training** refers to the neural network training. **Inference** refers to the overall process of obtaining the joint posterior estimate, either through MCMC or through SNE/NE approaches, which includes the training times.



### 3.6.1 Sequential neural posterior estimation (SNPE)

Following the general procedure, the joint posterior is directly estimated. That is, the parameter samples ( $\theta \sim \tilde{p}(\theta)$ ) and the simulated data ( $\mathbf{x} \sim p(\mathbf{x}|\theta)$ ) are combined into a dataset to train a neural network. The output of the neural network is the joint posterior itself which is obtained after conditioning the learned. It is of note that, sampling from the trained neural network to obtain the posterior is very fast ( $\sim$  few seconds).

### 3.6.2 Other approaches: SNLE and SNRE (likelihood and ratio estimation)

Instead of learning the posterior directly, one can infer other objects which are related to the posterior as follows:

1. **SNLE (Sequential neural likelihood estimation):** In this, instead of the posterior, likelihood is estimated ( $p(\mathbf{x} | \theta)$ ) as a distribution using various neural architectures. Multiplied with the prior, it yields the object that can be used by MCMC to yield the joint posterior estimate. Since every round uses the MCMC step, it can be expensive to run this algorithm. Indeed, it is empirically found to be computationally expensive (sec. 4). It offers MDN, MAD and MADE as architectures available for inference, since they all are probability density estimators and the likelihood can be defined as a probability density function (sec. 3.1).
2. **SNRE (Sequential neural ratio estimation):** Learns the ratio  $p(\mathbf{x} | \theta)/p(\mathbf{x})$  via a classifier approach. This can then yield posterior samples via slice sampling.

All three were investigated for the exponential model, but SNPE was found to be faster and more flexible (see results).

### 3.6.3 NPE: An alternative to SNPE

A key challenge with SNPE and MCMC is that they have to be re-run for every 'observed' dataset. This is because they utilise the 'observed' dataset during the training (SNPE) or inference (MCMC). This essentially means that the average training times across more datasets remains the **same**. That is, if the training time is  $t$ , then for  $n$  datasets, the average training time is  $n * t/n = t$  as shown below:

NPE, the non-sequential alternative to SNPE, is a one-shot estimation to yield a neural network which generalises over multiple datasets. This is done by simulating a large number of parameter values ( $\theta$ ) from the prior and simulating data corresponding to each sample  $\tilde{\mathbf{x}}$ . These sampled parameter values and simulated data,  $\{\theta, \tilde{\mathbf{x}}\}$ , are then used to train the neural network without ever using the 'observed' dataset during the training. In essence, this yields a neural network which 'has seen enough of the data space'. This means that the trained neural network yields the joint posteriors corresponding multiple 'observed' datasets, immediately, without retraining on every one of them. This is clearly in contrast to the SNE and the MCMC approaches mentioned above. Therefore, a trained NPE estimate, splits the training time across all the datasets which are supplied as inputs to obtain corresponding joint posteriors. This is referred to as **amortisation of training times** or amortisation in short. That is, if



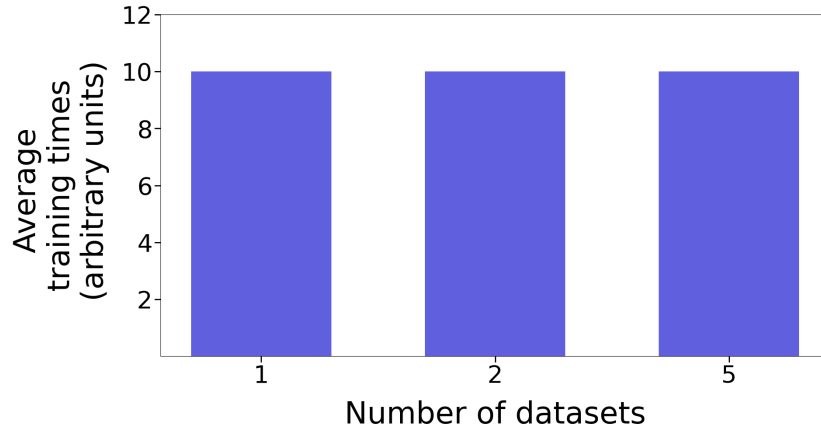


Figure 3.4: Training times for SNPE/inference times for MCMC in arbitrary units averaged over multiple datasets.

the training time is  $T$ , then for  $n$  datasets, the average training time is  $T/n$  as shown below:

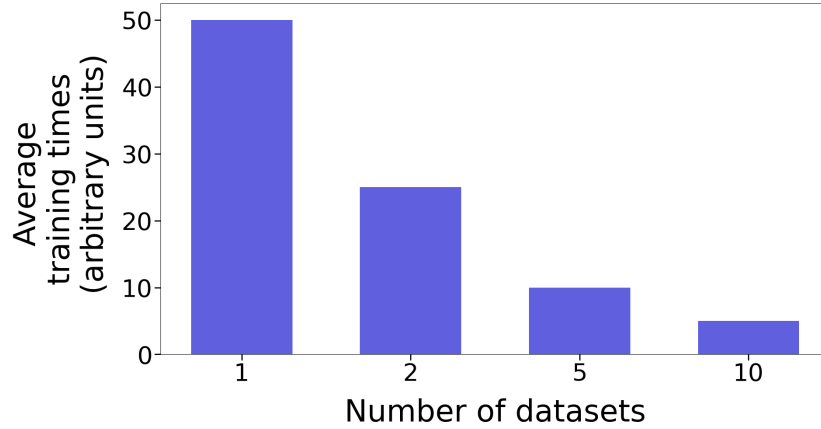


Figure 3.5: Training times for NPE in arbitrary units averaged over multiple datasets.

Notably, even if the training time for NPE is higher than that for SNPE or inference time of MCMC (10 units in fig. 3.4 and 50 units in fig. 3.5), with enough datasets, the average training time for NPE could be reduced below that for SNPE (the case of 10 datasets in fig. 3.5). This makes NPE desirable for decreasing the training times and hence inference times for the joint posteriors over parameters of interest.

## 3.7 Packages and implementation

### 3.7.1 pymc for ODE Bayesian Inference

The `pymc` probabilistic programming library, was used to implement Bayesian inference for ODE models (PyMC Development Team, 2024). `pymc` supports various samplers such as metropolis-hastings. gradient-based samplers (NUTS), while using `odeint` for integration of the ODEs. Starting with the simpler exponential model, to the simplified repressilator models, a prior is assigned to the parameters  $\theta$ , for example, a uniform distribution. Scipy `odeint` then solves the ODE for a given sampled parameter set. `DEMetropolisZ` is used as the sampler because of its efficiency, all the other samplers

were empirically found to be inefficient. `pymc`'s posterior samples and predictive simulations serve as the ground truth estimate for evaluating and benchmarking the neural network estimates.

### 3.7.2 `sbi` Package and Software Implementation

The SBI methods were implemented using the open-source `sbi` package (sbi Development Team, 2024) (version 0.23.3). In this framework, the prior is initialised and a simulator function (our ODE solver with noise) is defined. It internally handles the training loop for SNPE, SNLE, or SNRE, including sequential rounds. Various inputs can be specified, such as changing the architectures, specifying the number of rounds, specifying the hyperparameters of the different architectures. All experiments are performed on the Imperial College High Performance Computing (HPC) cluster.

# Chapter 4

## Results and Interpretation

As mentioned earlier, one of the limitations of the previous work was the lack of a ground truth for the repressilator model (students, 2024). This was the starting point for this study. Since, the repressilator model is high dimensional and unwieldy, including the simplified versions (sec. 4.4), this study starts with a simple exponential model. This was done for developing a handle on the neural network-based SBI algorithms (**neural algorithms**), conduct sanity checks, and establish a pipeline involving the joint posterior estimation using SBI and MCMC, utilising ground truth comparisons.

Revising the pipeline design, (1.5) we:

1. Start with synthetically generated 'observed' data for the model of interest,
2. Infer the ground truth posterior distribution (**ground truth**) from MCMC and establish it using diagnostics such as the Gelman Rubin (GR) statistic and Autocorrelation plots (not shown),
3. Infer the joint posterior for the same parameters using neural algorithms (**sequential neural estimation (SNE) estimate**), utilising various algorithms, neural network architectures and hyperparameter optimisation,
4. Compare the SNE estimate with the ground truth, benchmark with scoring and perform posterior predictive checks

### 4.1 Test model: Exponential decay

The exponential decay model is a simple model that models the protein decay dynamics (Murray, 2007). The rate of decay being proportional to the concentration of the protein, yields an exponentially decaying concentration trajectory, when these equations are solved. This is shown below (bold symbols indicate vectors):

$$\frac{dx}{dt} = -\beta x \quad (4.1)$$

$$\implies \mathbf{x}(t) = x_0 \cdot \exp(-\beta \mathbf{t}) \quad (4.2)$$

Where,  $\mathbf{t}$  is a vector of time points. As explained in sec, 3, one can inject additive Gaussian noise to this solution to mimic experimental data and cast this model to a generalised linear model. This was done as follows:

$$\mathbf{x}(t) = x_0 \cdot \exp(-\beta t) + \boldsymbol{\varepsilon} \quad \varepsilon \sim N(0, \sigma^2) \quad (4.3)$$

$$\implies x_i \sim N(x_0 \cdot \exp(-\beta t_i), \sigma^2) \quad (4.4)$$

Where,  $\mathbf{x} = x_1, \dots, x_n$  are simulated data at  $n$  time points,  $\boldsymbol{\varepsilon}$  is the vector of Gaussian noises,  $\beta$  is decay rate and  $\sigma$  is the standard deviation of the injected Gaussian noise ( $\boldsymbol{\varepsilon}$ ). The Gaussian noise is injected through sampling independently and identically (i.i.d) from  $\mathcal{N}(0, \sigma^2)$  denoting a Gaussian distribution with mean 0 and variance  $\sigma^2$ . This is illustrated by fig. 3.1.

The parameters of this model could either be  $\theta = \beta$  (sec, 4.2) or  $\boldsymbol{\theta} = (\beta, \sigma)$  (sec, 4.3). Since a likelihood function is accessible for this model (sec. 3.2), the `pymc` library can be used to obtain the ground truth. The `pymc` library has a feature of summarising various statistics, including the GR statistic. Hence, unless mentioned otherwise, the GR statistic for every ground truth estimate is close to one (sec. 3.3).

## 4.2 Exponential decay: One-dimensional case

As a starting point, the standard deviation of the injected noise and the initial conditions  $(\sigma, x_0)$  were assumed to be constant. This rendered the model one-dimensional with the decay rate ( $\beta$ ) being the sole parameter. Thus, here  $\theta = \beta$ . As mentioned in sec. 3.1, benchmarking is critical to evaluate the SNE estimates obtained from the SBI procedure. This is done as follows:

1. The **% mean score** (mean score, in short) is calculated as:

$$\% \text{ mean score} = \frac{|\mu_{SNE} - \mu_{GT}|}{\mu_{GT}} \cdot 100, \quad (4.5)$$

where  $\mu_{SNE}, \mu_{GT}$  are means of the SNE estimate and ground truth estimate respectively.

2. The **% standard deviation score** (standard deviation score, in short) is calculated as:

$$\% \text{ std score} = \frac{|\sigma_{SNE} - \sigma_{GT}|}{\sigma_{GT}} \cdot 100, \quad (4.6)$$

where  $\sigma_{SNE}, \sigma_{GT}$  are standard deviation of the SNE estimate and ground truth estimate respectively.

Here, both mean and standard deviation scores are used for benchmarking. This is because the quality of inference depends on how well the SNE estimate approximates the ground truth in terms of both its central value and the spread around it. Therefore, **higher scores imply poorer joint posterior inference** by SNE.

Although, better alternatives for benchmarking do exist (sec. 5.2), this kind of benchmarking is useful for comparing algorithms, architectures, and drawing conclusions about the difficulties of inference (see below). Before the algorithms are run, the following are kept invariant for all the SNE and NE algorithms:

1. The value of the initial condition is fixed,  $x_0 = 2.0$  in the eq. 4.2,
2. Since the true value of  $\beta = 1.5$ , a prior distribution is set to be  $p(\theta) = p(\beta) = [0, 4]$ ,
3. The standard deviation of the noise  $\sigma$  is kept consistent between the ‘observed’ dataset and the simulator which generates ‘simulated’ data (sec. 3.2.1).
4. The number of time points taken for the ‘observed’ and simulated data are  $n = 50$ ,
5. Any joint posterior estimate ‘passes’ the PPC if about 5% of the points lie outside the 95% region indicated by dotted lines in every PPC figure. The intensity of the red/blue colours indicate the density of trajectories in that region. The median trajectory (solid line) indicates the median concentration value at at time point across all predicted trajectories.

On a technical note, hereafter, the sequential neural estimation is referred to as SNE (**SNPE**, **SNLE**, **SNRE**), whereas the non-sequential neural estimation (enabling amortisation of average training times) is referred to as NE (**NPE**, **NLE**, **NRE**). Starting with all three algorithms with all their architectures, following are inference objectives for the one dimensional model:

1. Comparison of the joint posterior inference using SNPE, SNLE and SNRE on dataset 1, through ground truth comparisons, PPCs, scores and inference times,
2. Comparison of the joint posterior inference through non-sequential neural estimation (sec. 3.6.3) using NPE, NLE and NRE on dataset 1 and dataset 2, through scores and inference times.

The inference procedure is started with implementing SNPE, followed by SNLE and SNRE (Sec 3.6) (only for the one-dimensional case). The ‘observed’ datasets are obtained in accordance with the eq. 4.3 and are shown in the figure 4.1, alongside PPC for the ground truth corresponding to the first dataset:

1. Dataset 1:  $\beta = 1.5$  and  $\sigma = 0.1$ ,
2. Dataset 2:  $\beta = 1.5$  and  $\sigma = 0.2$ , reflecting more noise

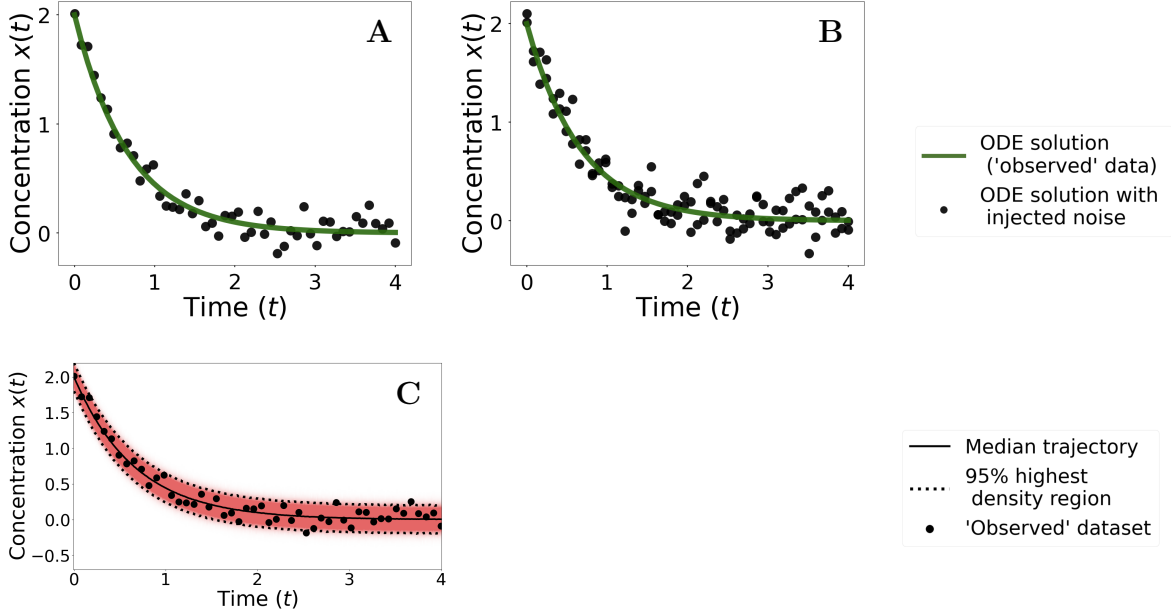


Figure 4.1: ‘Observed’ datasets with different true parameter values and PPC for one ground truth. **A**  $\beta = 1.5$ ,  $\sigma = 0.1$ , **B**  $\beta = 1.5$ ,  $\sigma = 0.2$ , reflecting more noise. **C** Ground truth for dataset 1

#### 4.2.1 SNPE and NPE yield reasonable estimates with all three architectures

The inference procedure is started with SNPE for dataset one using all architectures. Joint posterior estimates from each architecture visually appear to approximate the ground truth well. The same is true for NPE (enabling amortisation) estimates. To reiterate, the NPE estimates are obtained by conditioning (sec. 3.6.3) on datasets one and two, **post training**. This means that if it took time  $T$  for the training, the inference time per dataset becomes  $T/2$  for each of the two datasets. Lastly, the SNPE estimates from all architectures pass the PPC check. PPCs for NPE estimates are not shown, but they pass the checks as well.

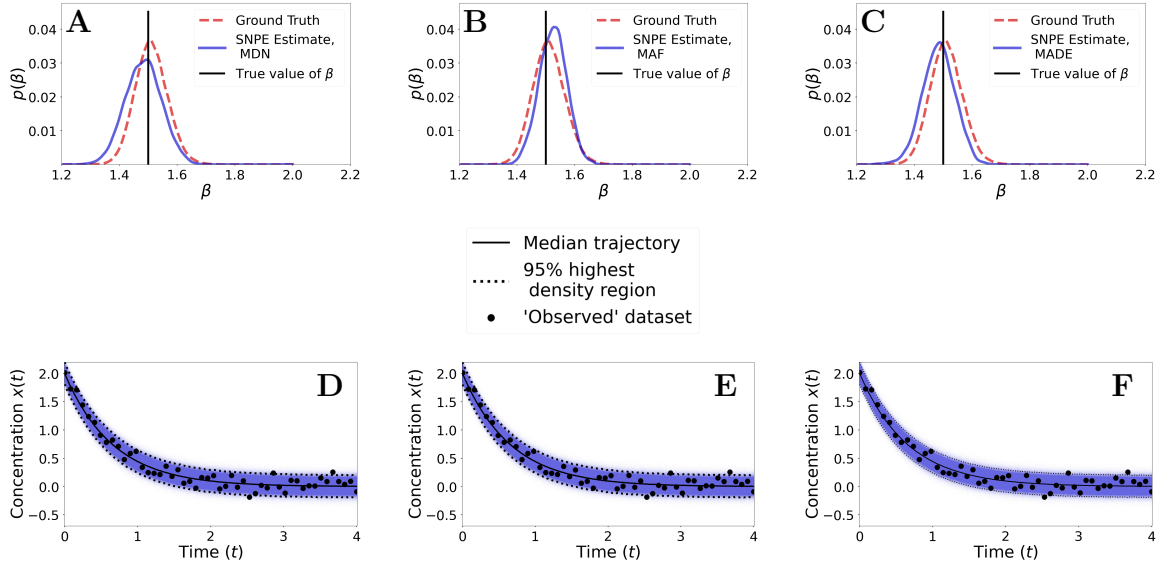


Figure 4.2: Visual comparison between SNPE estimates and the ground truth for dataset one. **A-C** SNPE estimates for parameter  $\beta$  using MDN, MAF and MADE respectively. **D-F** Positive PPC for SNPE estimates using MDN, MAF and MADE respectively.

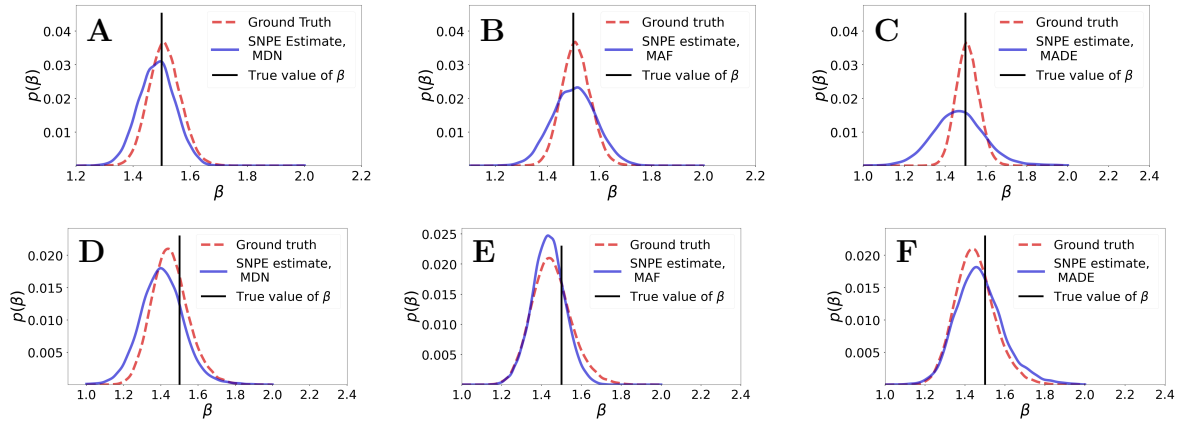


Figure 4.3: Visual comparison between NPE estimates and the ground truth for dataset one and two. **A-C** NPE estimates on **dataset one** for parameter  $\beta$  using MDN, MAF and MADE respectively. **D-F** NPE estimates on **dataset two** for parameter  $\beta$  using MDN, MAF and MADE respectively.

### 4.2.2 SNLE and NLE (enabling amortisation) yield reasonable estimates with all three architectures

The same inference procedure is conducted using SNLE for dataset one using all architectures. Results for only MDN is shown (other architectures in 4.2). Again, joint posterior estimates from each architecture visually appear to approximate the ground truth well. The same is true for NLE (enabling amortisation) estimates. Lastly, the SNLE MDN estimates passes the PPC check.

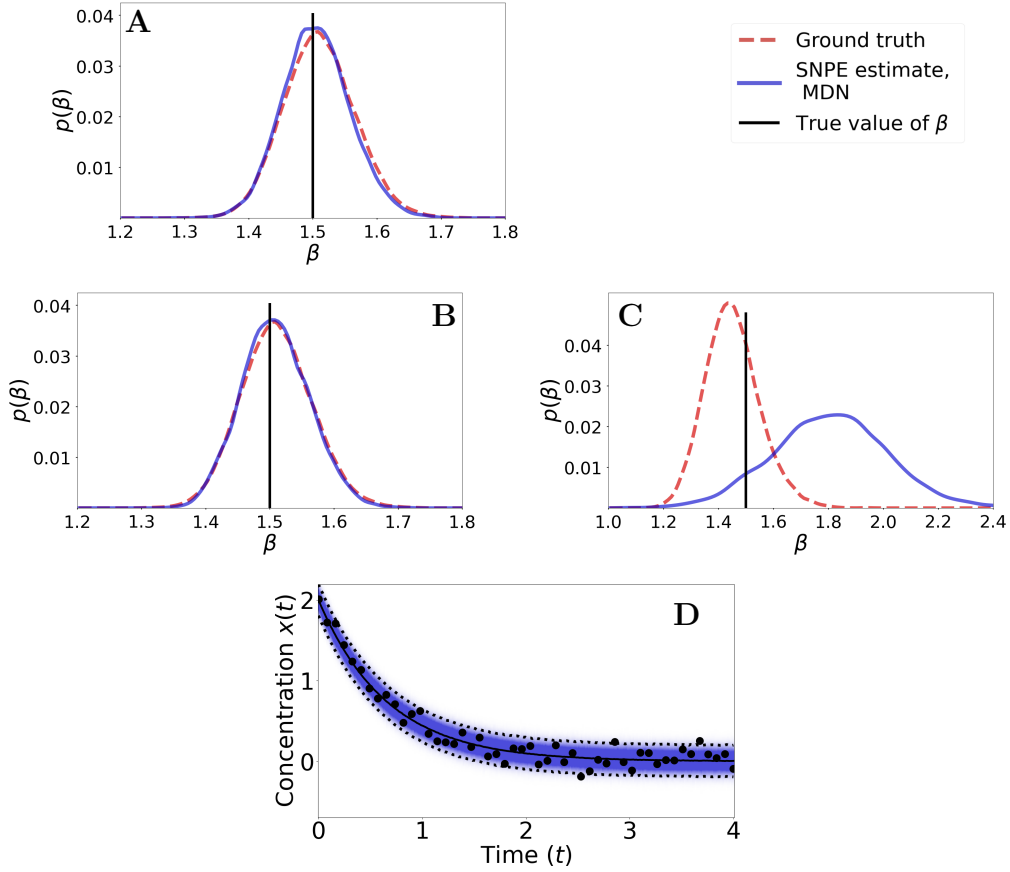


Figure 4.4: Visual comparison between SNLE and NLE estimates and the ground truth for dataset one and two. **A** SNLE estimate on **dataset one** for parameter  $\beta$  using MDN. **B**, **C** NLE estimates using MDN on datasets one and two. **D** PPC for SNLE estimate using MDN on **dataset one**.

### 4.2.3 SNRE and NRE (enabling amortisation) yield reasonable estimates with both the architectures, but with lesser flexibility

Finally, the same inference procedure is conducted using SNRE for dataset one using all architectures. Results for only Resnet is shown (other architectures in 4.2). Again, joint posterior estimates from each architecture visually appear to approximate the ground truth well. The same is true for NRE (enabling amortisation) estimates. Lastly, the SNRE Resnet estimates passes the PPC check.



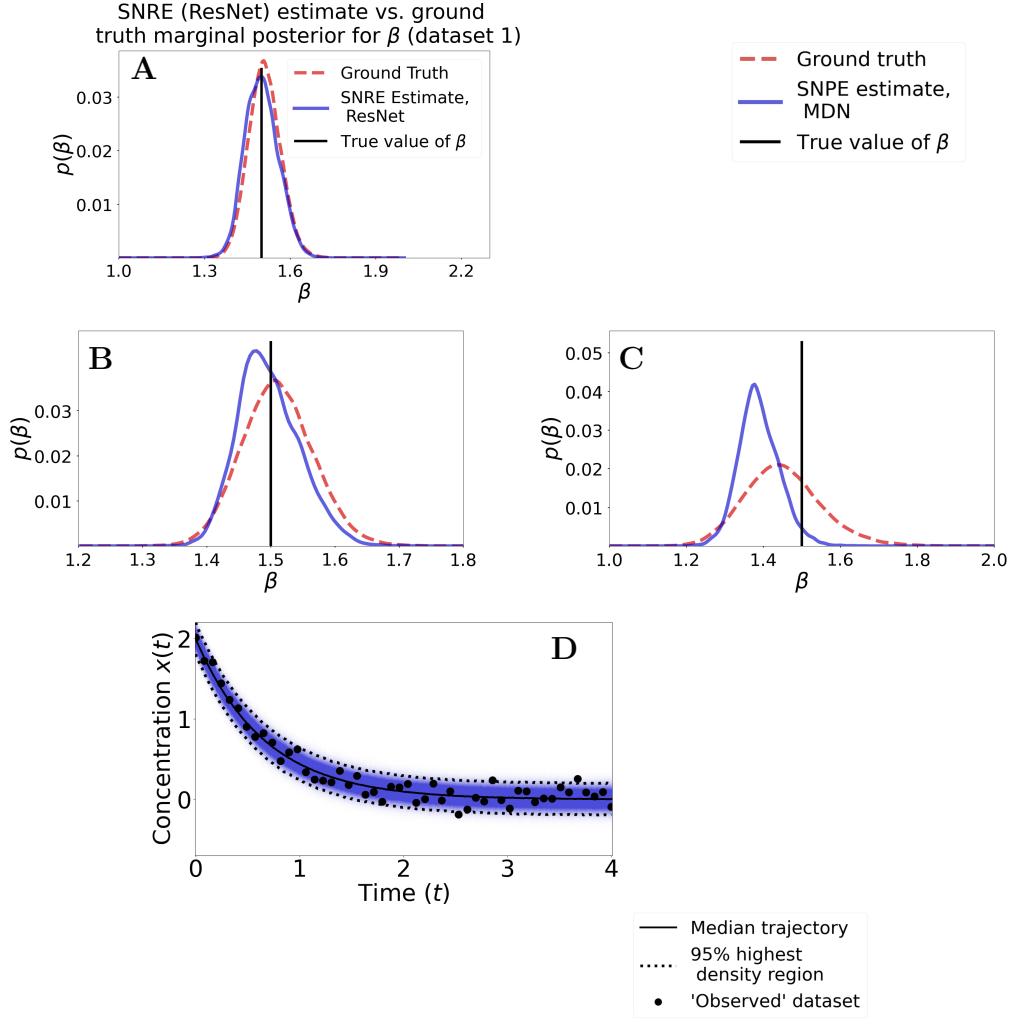


Figure 4.5: Visual comparison between SNRE and NRE estimates and the ground truth for dataset one and two. **A** SNRE estimate on **dataset one** for parameter  $\beta$  using Resnet. **B,C** NRE estimates using Resnet on datasets one and two. **D** PPC for SNRE estimate using Resnet on dataset one.

At this point it is clear that the SNPE, SNLE and SNRE all yield joint posterior estimates which pass the preliminary checks. This is then immediately followed by benchmarking, to quantify their performance.

#### 4.2.4 Comparison of scores for SNPE, SNLE and SNRE

The computed mean and standard deviation scores and the obtained inference times for all algorithms across architectures, are shown below. The mean and standard deviation scores are lowest for SNLE, while its inference time is the highest (not shown). This is because SNLE utilises MCMC to build the posterior from the learned likelihood (sec. 3.6.2). The performance of SNRE is comparable to that of SNPE. However, for all the algorithms, the mean score is categorically less than the standard deviation score. Although absolute differences are computed, typically, a high standard deviation score is due to SNE estimates being more **conservative**. This means that SNE estimate predicts a higher uncertainty for the given parameter(s) than the ground truth estimate. This is a recurrent feature of these algorithms and preferred to the opposite case, where the estimates are ‘overconfident’ (Hermans et al., 2021).

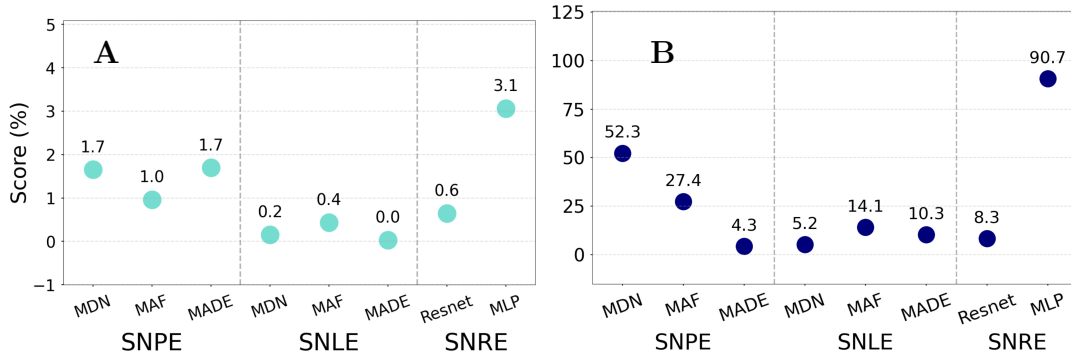


Figure 4.6: Comparison of mean and standard deviation scores for SNPE, SNLE and SNRE across all architectures. **A** Mean scores. **B** Standard deviation scores. **C** Inference times.

Lastly, the sampling time of SNPE (time taken to draw  $N$  samples) is at least an order of magnitude lower than that for SNLE and SNRE (not shown). This becomes an important factor for NPE estimates, from which samples are drawn independently for each dataset. Based on the results so far, the following can be concluded:

1. For a simple, one dimensional model, SNPE, SNRE and SNLE, yield conservative estimates which pass PPCs and yield low mean scores but high standard deviation scores,
2. SNPE is preferred to SNLE. This is because of its faster inference time, albeit with marginally lesser accuracy, which aligns with the objectives of this study (chapter 2),
3. amortisation is viable for all the NE algorithms (PPCs and scores not shown),
4. SNPE is preferred to SNRE because SNPE offers more flexibility in the form of state-of-the-art architectures (MDN, MAF and MADE). Furthermore, the sam-

pling time for SNPE is much lesser, while its estimates are almost as accurate as SNRE

Thus, SNPE is the algorithm of choice moving forward, with all three architectures. Amortisation appears viable. The objectives of this section are met.

## 4.3 Test Model: Two-dimensional case

Now, we consider the standard deviation of the injected noise to be a parameter as well ( $\sigma$ ). Thus, here,  $\theta = (\beta, \sigma)$ . Since uncertainty on the noise is also being inferred, the inference becomes non-trivial. Unprincipled implementation SNPE for the 2D model has resulted in numerous arbitrary estimates (results not shown). This prompted the exploration of hyper-parameter optimisation (sec.3.5) to tune the hyper-parameters such as hidden features, learning rate and the number of components/transformations (sec. 3.5). With this, two phases are defined for the whole inference pipeline. The **hyper-parameter optimisation** phase and the **main inference** phase. This is done as follows:

1. We start with the **hyper-parameter optimisation** phase. Ranges for each of the hyper-parameter are chosen from experience with the algorithms, for example, [6, 7, 8] for number of components/transformations,
2. Grid search (sec.3.5) is employed, by choosing each parameter configuration (triplet) sequentially,
3. For every configuration, the SBI algorithm is run on a given ‘observed’ dataset using a small number of simulations ( $\sim 1/10$  times the number of simulations used for **main inference**), yielding  $N$  SNPE estimates for  $N$  configurations,
4. A separate scoring scheme than plotting is followed to score each SNE estimate. Firstly, the difference of the mean (MD score) of the ground truth and each of the  $N$  estimates is taken. Ten % of the estimates with lowest MD scores were selected. Secondly, the difference of the variance (VD score) of the ground truth and each of ten % estimates is taken. The SNE estimate with the lowest VD score is chosen (**best estimate**).
5. Finally, the hyper-parameter configuration which gives rise to the best estimate is chosen and the hyper-parameter optimisation stage is terminated (best configuration).
6. With the best configuration, the **main inference** is initiated with suitable number of simulations, as was done in the previous section.

Having narrowed down to SNPE, following are the inference objectives for the two dimensional model:

1. Comparison of the joint posterior inference using SNPE, across all architectures, through ground truth comparisons, PPCs, scores and inference times,
2. Comparison of the joint posterior inference through NPE (enabling amortisation)

across all architectures, (sec.3.6.3) on multiple datasets, through scores and inference times.

Interestingly, the best configuration changes with the ‘observed’ dataset in use. This necessitates the **hyper-parameter optimisation** stage, prior to the main inference stage, for every dataset. The following figure shows the datasets used for inference and the PPC checks for dataset one and dataset four for comparison with the SNPE estimates:

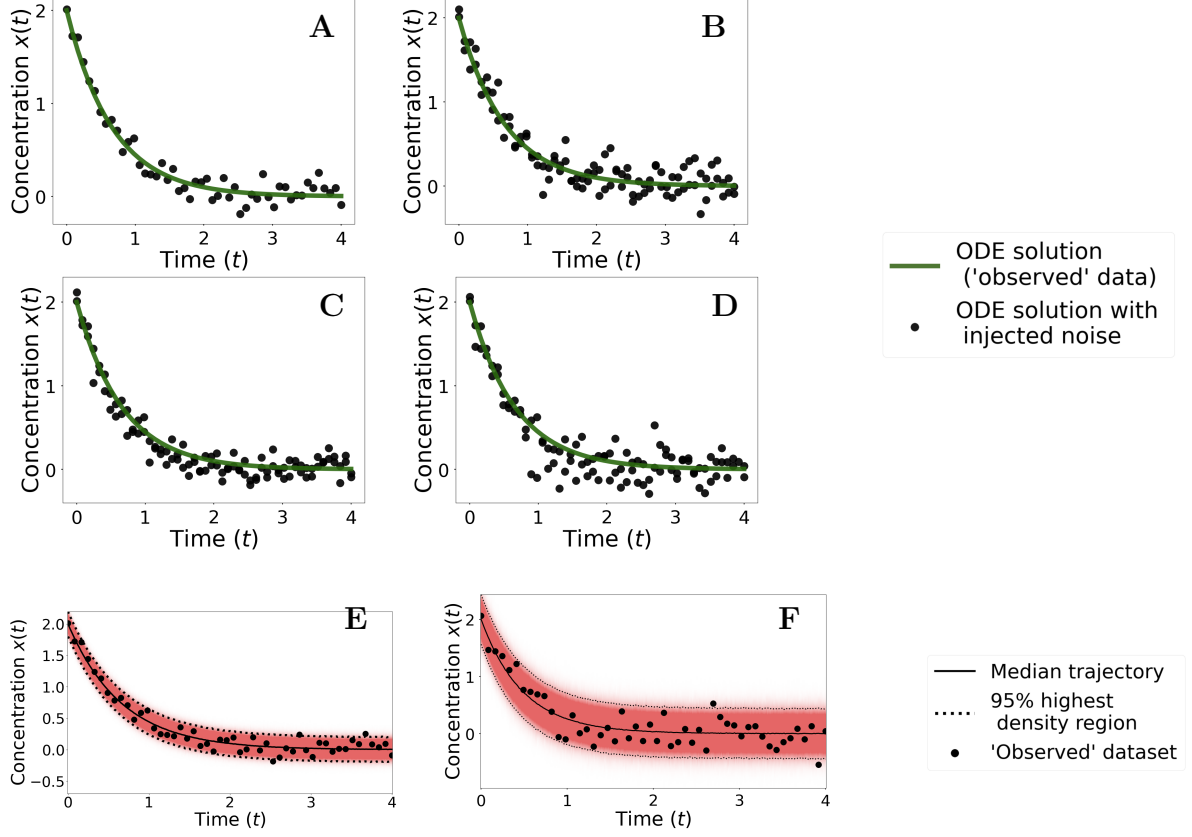


Figure 4.7: ‘Observed’ datasets with different true parameter values and PPC for two ground truths. **A**  $\beta = 1.5$ ,  $\sigma = 0.1$ , **B**  $\beta = 1.5$ ,  $\sigma = 0.2$  **C**  $\beta = 2.0$ ,  $\sigma = 0.1$  and **D**  $\beta = 2.0$ ,  $\sigma = 0.2$ . **E** Ground truth for **A**. **F** Ground truth for **D**.

Hereafter, the joint posterior obtained from SNPE is the **SNPE estimate** or **NPE estimate** (when amortisation is possible).

## SNPE followed with hyperparameter optimisation leads to problematic estimates for dataset one

Dataset one was considered as the ‘observed’ dataset and the hyper-parameter optimisation stage was followed by the main inference stage, using all three architectures. The resultant SNPE estimates are highly conservative, so much so that estimates from every architecture fails the PPC. This finding is crucial as it highlights a key limitation of these algorithms (sec. 5.2)

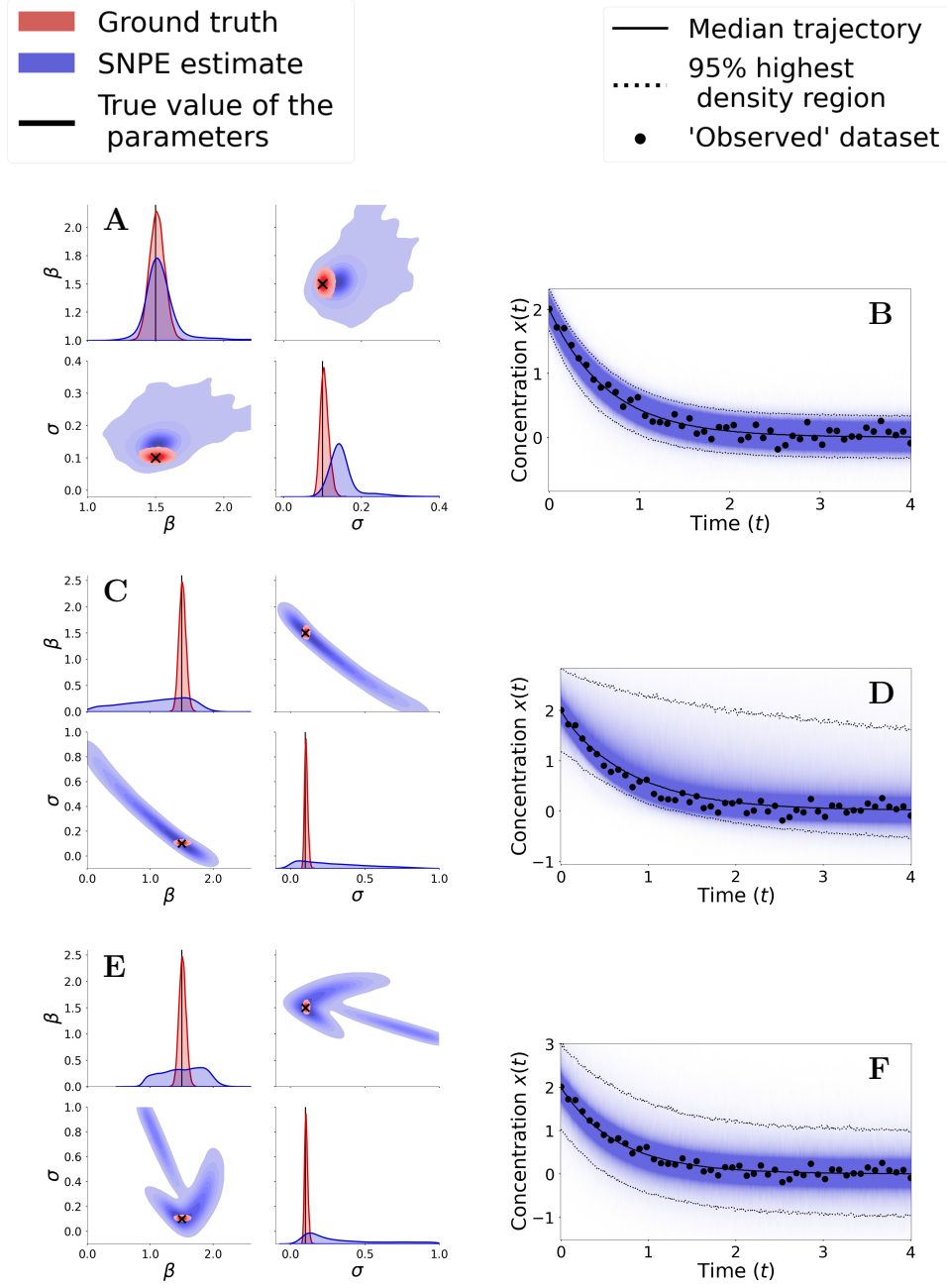


Figure 4.8: Visual comparison between SNPE estimates and the ground truth for dataset one. **A**, **C** and **E** Pair plots for the SNPE estimate for both the parameters ( $\beta$ ,  $\sigma$ ) (blue) using MDN, MAF and MADE respectively. **B**, **D** and **F** SNPE estimates fail PPCs for MDN, MAF and MADE respectively.

## SNPE followed with hyperparameter optimisation leads to relatively conservative estimates for dataset 4

The same inference procedure was now implemented for dataset four, using all three architectures. Results only for MDN are shown here. In contrast with the previous estimate, The SNPE estimate approximates the ground truth well (conservative) and passes the PPC. Similar estimates are obtained using other architectures (not shown).

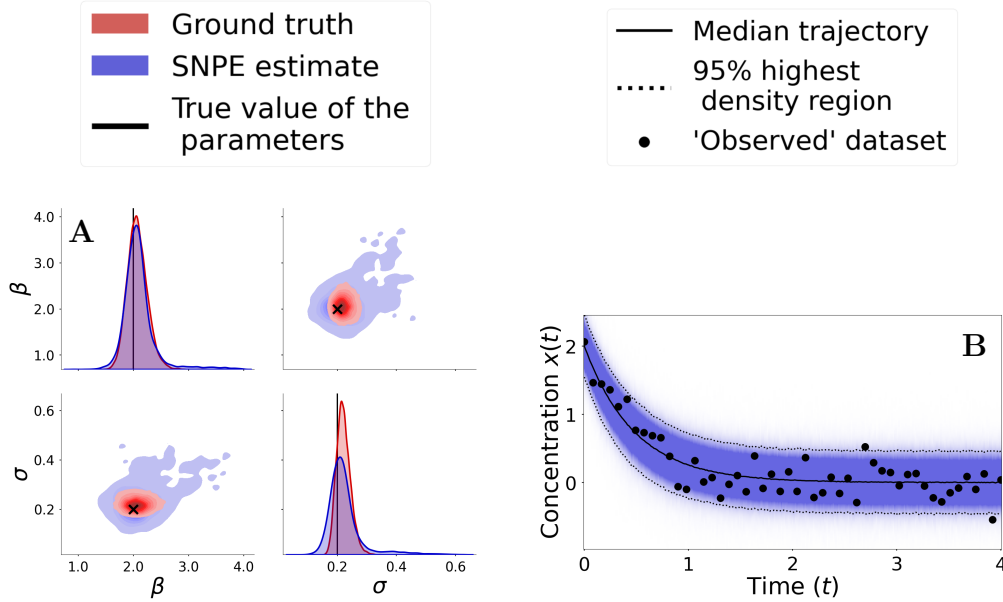


Figure 4.9: Visual comparison between SNPE estimates and the ground truth for dataset four. **A** shows pair plot for the SNPE estimate using MDN. **B** shows PPC checks on the SNPE estimate using MDN.

## Independent runs of SNPE yield robust estimates for Dataset 4

Every time the SNPE algorithm is run, the weights of the neural network architecture to be trained are initialised to be random values within a specified range. This, combined with the stochastic gradient descent employed for network training, introduces stochasticity in the outputs (sec. 3.4). As a result, the SNPE estimates from independent implementations can differ from each other. To evaluate this, ten independent SNPE estimates (only five plotted) are obtained using the same inference procedure. This is done for dataset two as the ‘observed’ dataset. For benchmarking % mean and standard deviation scores were computed as before (sec. 4.2). Only the results for MDN are shown. As is clear, both visually and numerically, the independent estimates differ ever so slightly from each other. This establishes robustness for SNPE for this model.

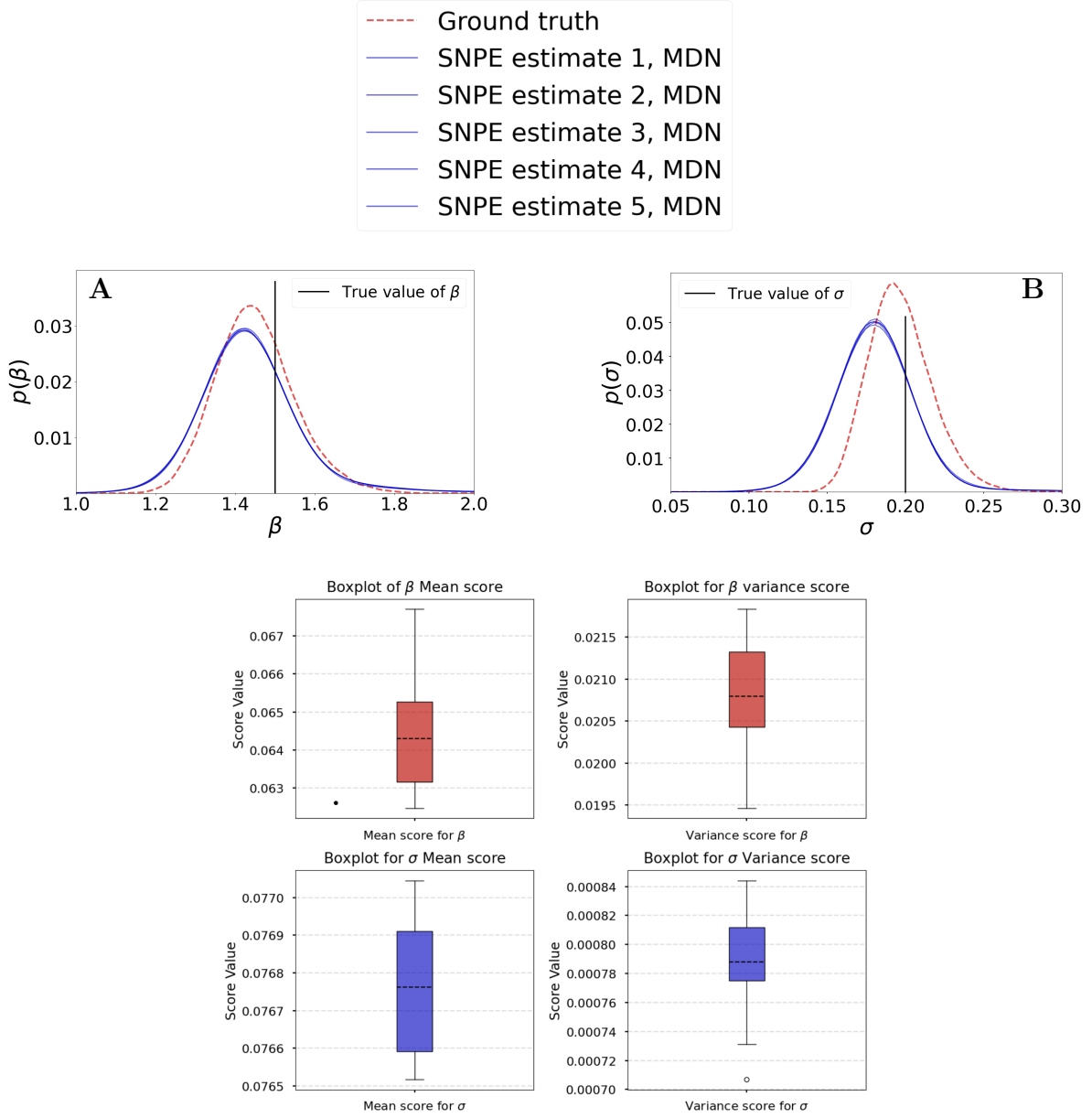


Figure 4.10: Visual comparison between 5 independent SNPE vs. ground truth estimates for dataset one. The box plot shows variation in the mean and standard deviation scores for both parameters across 10 independent estimates. The scores are not shown in percentages.

## NPE yields estimates enabling amortisation for the 2D model for all datasets

On a technical note, hyper-parameter optimisation is not repeated in the case of NPE. That is, the best hyperparameter configuration chosen for SNPE is also utilised for NPE. The following figure shows estimates enabling amortisation corresponding to all four datasets. Here, the NPE estimates are shown only for MDN. Similar results are obtained for the other two architectures (not shown). These estimates also pass PPCs (not shown). This establishes amortisation for this model.

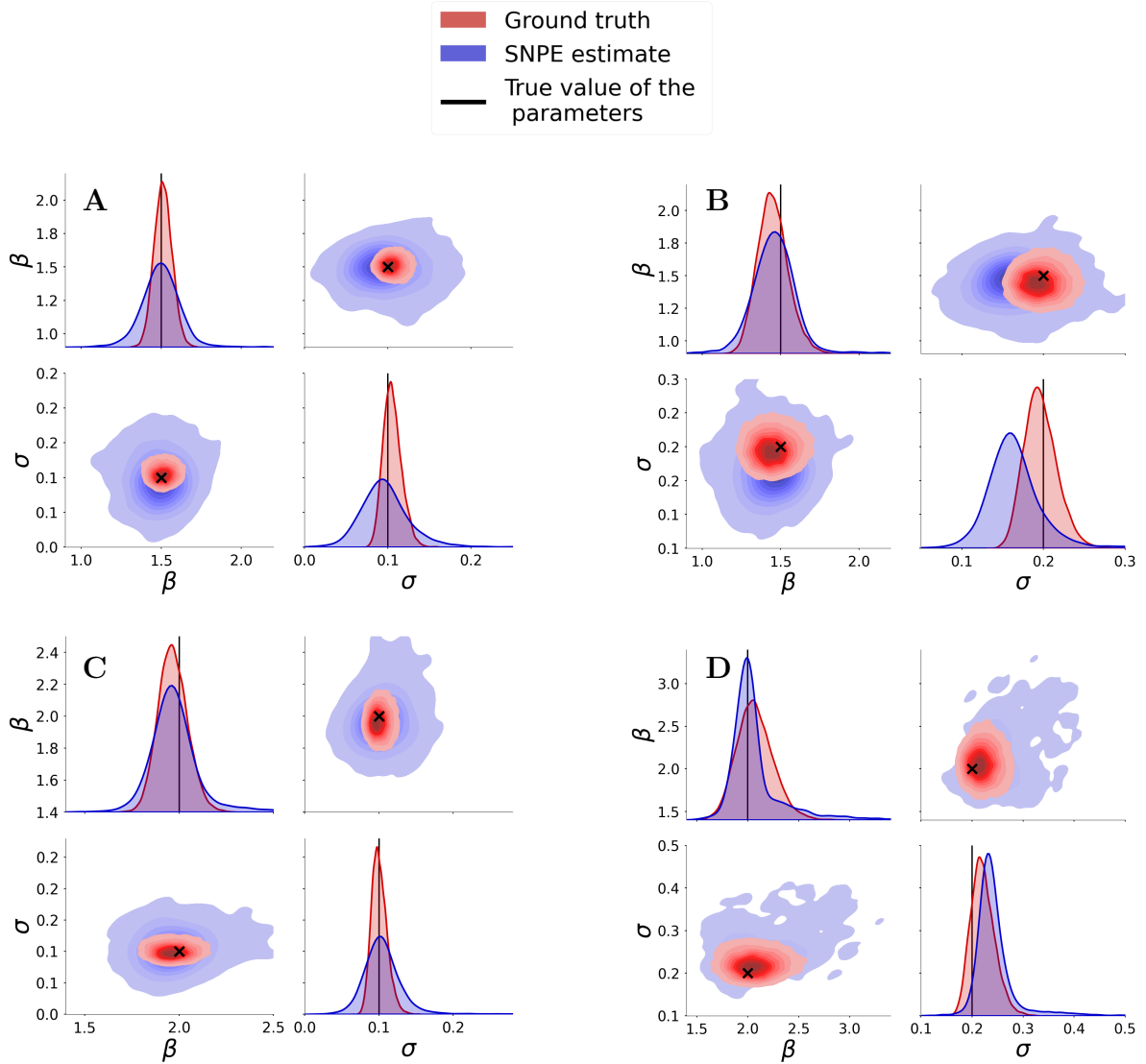


Figure 4.11: Visual comparison between NPE estimates using MDN and the ground truth for all datasets. **A**, **B**, **C** and **D** show pair plots of the NPE estimate compared with the ground truth for datasets one, two, three and four respectively.

Again, it is clear that the all architectures of SNPE posterior estimates which pass the preliminary checks (with occasional exceptions). This is again immediately followed by benchmarking, to quantify their performance.



## Scores and inference times for all architectures

Invariably, both the mean and standard deviation scores are categorically higher for  $\sigma$  than for  $\beta$ . This suggests that inferring the parameters of noise is more difficult than inferring other parameters. Finally, inference time is the highest for MAF, and comparable for MDN and MADE. However, these inference times are orders of magnitude higher than the ground truth inference time from `pymc`. This is because, `pymc` is highly optimised for simple problems such as the exponential models, for which a closed form likelihood can be written down.

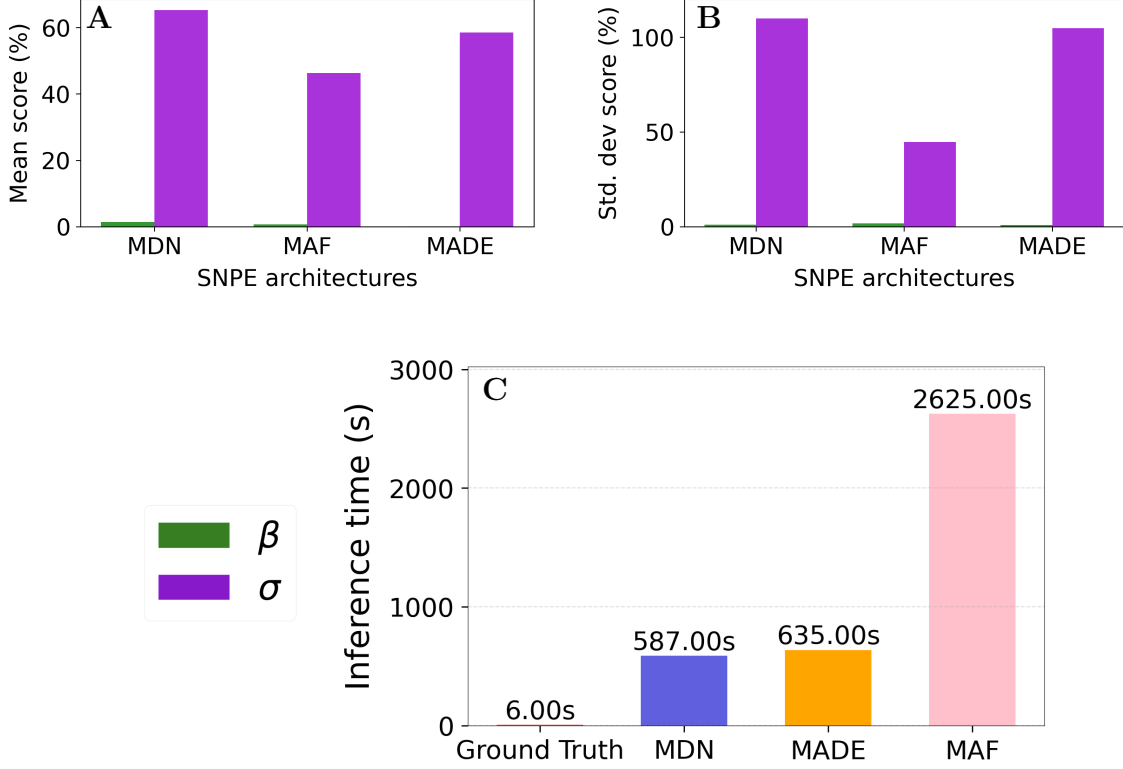


Figure 4.12: Visual of mean and standard deviation scores for SNPE across all architectures. **A** % mean scores. **B** % standard deviation scores. **C** Inference times.

Based on the results so far, the following can be concluded.

1. Hyperparameter optimisation, followed by SNPE yields reasonably accurate joint posteriors with MDN, MAF and MADE (with exceptions) however this can be dataset dependent,
2. SNPE yields much higher scores for the noise parameters than for the model parameters indicating relative difficulties with the inference of the noise,
3. SNPE yields robust estimates for the two dimensional model using all architectures,
4. Amortisation is viable for all the architectures (PPCs and scores not shown),
5. SNPE inference time is significantly higher for MAF, than for MADE and MDN.

Thus, MDN and MADE are relatively fast approximators of the joint posterior compared to MAF for the two dimensional exponential model. Amortisation appears viable.

The objectives of this section are met.

## 4.4 Repressilator Model

Equipped with positive sanity checks and a working pipeline, the primary model of the study, the repressilator, was tackled. Since inference becomes exponentially slower in higher dimensional models, it was tackled in a principled manner, by making simplifications to the eq 1.1 and 1.2. This yielded six parameterisations of the repressilator model (students, 2024), namely the three, six, nine, twelve, 15 and 21 (**full model**) parameter versions in increasing complexity and dimensionality. With these simplifications, it was assumed that, if successful for the simpler models, inference with minor changes would suffice for the more complex versions. The data shown in the rest of this chapter support this assumption to a certain extent. In simplifying, the parameter values associated with three mRNAs or three proteins were assumed to have the same value (in most cases). For example, leaky expression rates were assumed to be the same for each mRNA. Similarly for other parameters. Although this simplifies inference, it has ramifications for the biological significance of the results (sec. 5.2).

As explained in sec. 3.2.2, multiplicative noise was injected into the ODE solutions of the repressilator (additive Gaussian noise in the log space) to mimic experimental data. This was done to obtain the ‘observed’ dataset for each of the parameterised models. For every model only SNPE was used for inference. The benchmarking was done by computing the % mean and % standard deviation scores as before (sec. 4.2). Finally, the inference times for different architectures were compared as follows:

1. Since, the best configuration depends on the data, in every case, the inference time for SNPE using a particular architecture was computed by combining the times for the hyperparameter optimisation phase and the main inference phase (**SNPE inference time**),
2. A ratio of the SNPE inference time and the ground truth inference time was taken as a metric of the speed of SNPE inference. According to the auxiliary hypothesis (sec. 2), this ratio should approach unity for higher models, eventually going below 1.0 (faster inference than MCMC).

This was done to test the primary and auxiliary hypotheses in a principled manner. That is, to test if the ratio of inference times is below unity and if not, testing if it approaches unity for more complex models. This would also test whether the neural algorithms are able to mitigate the curse of dimensionality better than the MCMC algorithms as the complexity increases. With this, the following are the inference objectives for the repressilator models:

1. Comparison of SNPE joint posterior estimates using all architectures, through ground truth comparisons, PPCs, scores and inference times,
2. Comparison of the joint posterior inference through NPE, using all architectures,
3. Finding evidence to support or falsify the primary and auxiliary hypothesis.

## 4.5 Repressilator Model: Three-dimensional version

To start off, all the parameters except the three protein translation rates were assumed to be constants (1.1, 1.2). Thus, the parameters of this system are the three protein translation rates:  $\sigma_1, \sigma_2, \sigma_3$ . The following is the simplified system of ODEs for this model.

$$\frac{dm_i}{dt} = \frac{120.8}{1 + (0.002 \cdot p_j)^3} + 4.0 - 2.2 \cdot m_i \quad (4.7)$$

$$\frac{dp_i}{dt} = \sigma_i \cdot m_i - 0.8 \cdot p_i \quad (4.8)$$

Here, the parameters are assumed to be invariant across mRNA/protein species. The following figure shows ‘observed’ dataset for all three mRNA and proteins. This is done to show the ‘observed’ data in their totality. PPC for the ground truth on the dataset indicating predictive trajectories for mRNA 1 ( $m_1$ ) and protein 1 ( $p_1$ ) is also shown. This is shown to enable comparison with the PPC done on the SNPE estimate in figure (in blue).

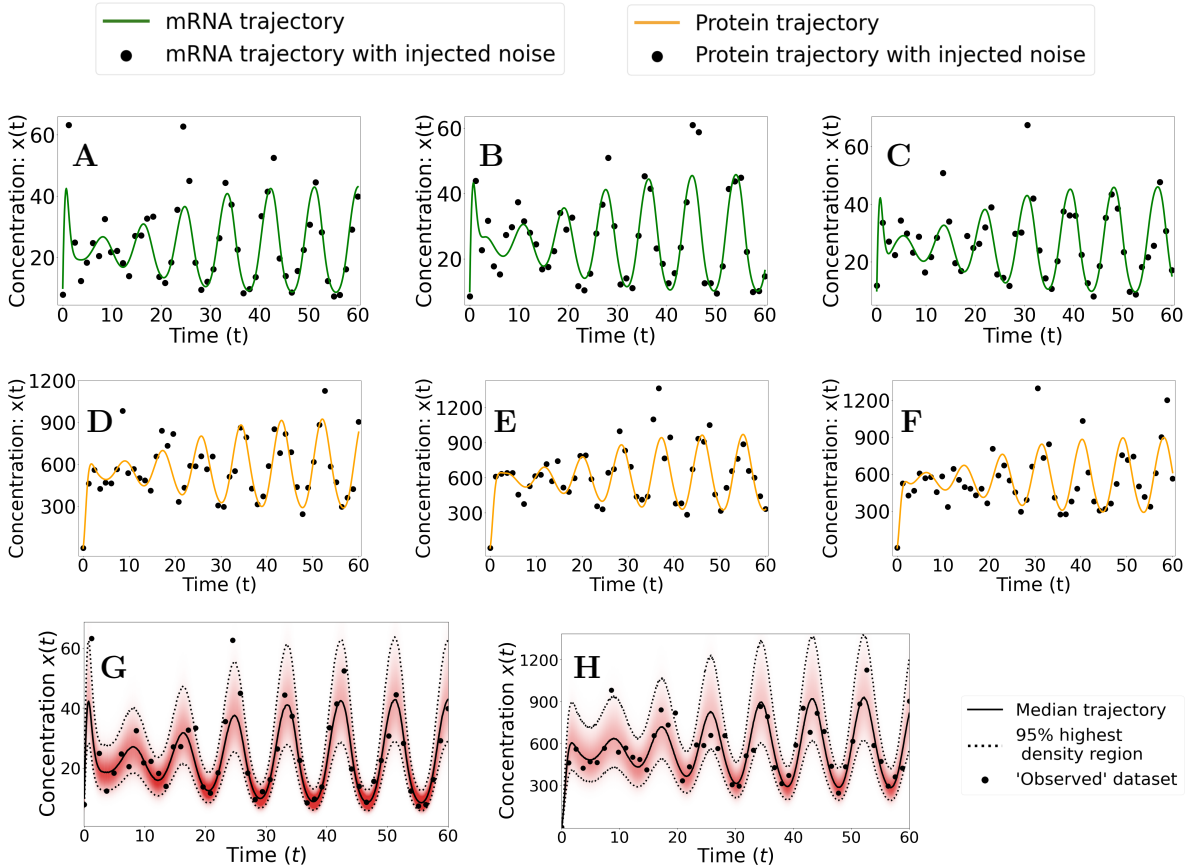


Figure 4.13: ‘Observed’ dataset alongside PPC from the ground truth for the three parameter model. **A**, **B** and **C** show ODE solutions injected with noise for mRNA 1, 2 and 3 respectively. **D**, **E** and **F** show ODE solutions injected with noise for protein 1, 2 and 3 respectively. (**G**) and (**H**) show predicted trajectories for mRNA and protein 1 from the PPC on the ground truth.

To reiterate, in the PPCs, the intensity of the red colour indicates the density of trajectories in that region. The dotted lines indicate the region containing 95 % of the trajectories. The median trajectory indicates the median concentration value at a time point across all predicted trajectories.

### **SNPE with hyperparameter optimisation leads to reasonable estimates for the ‘observed’ dataset**

The same inference procedure as before (hyperparameter optimisation followed by main inference) is implemented using all three architectures. Results are shown only for MDN. Although this estimate appears to ‘miss’ the true parameter values along some dimensions, it passes the PPC. Similar results are obtained for MADE. SNPE estimate from MAF, however, fails the PPC.

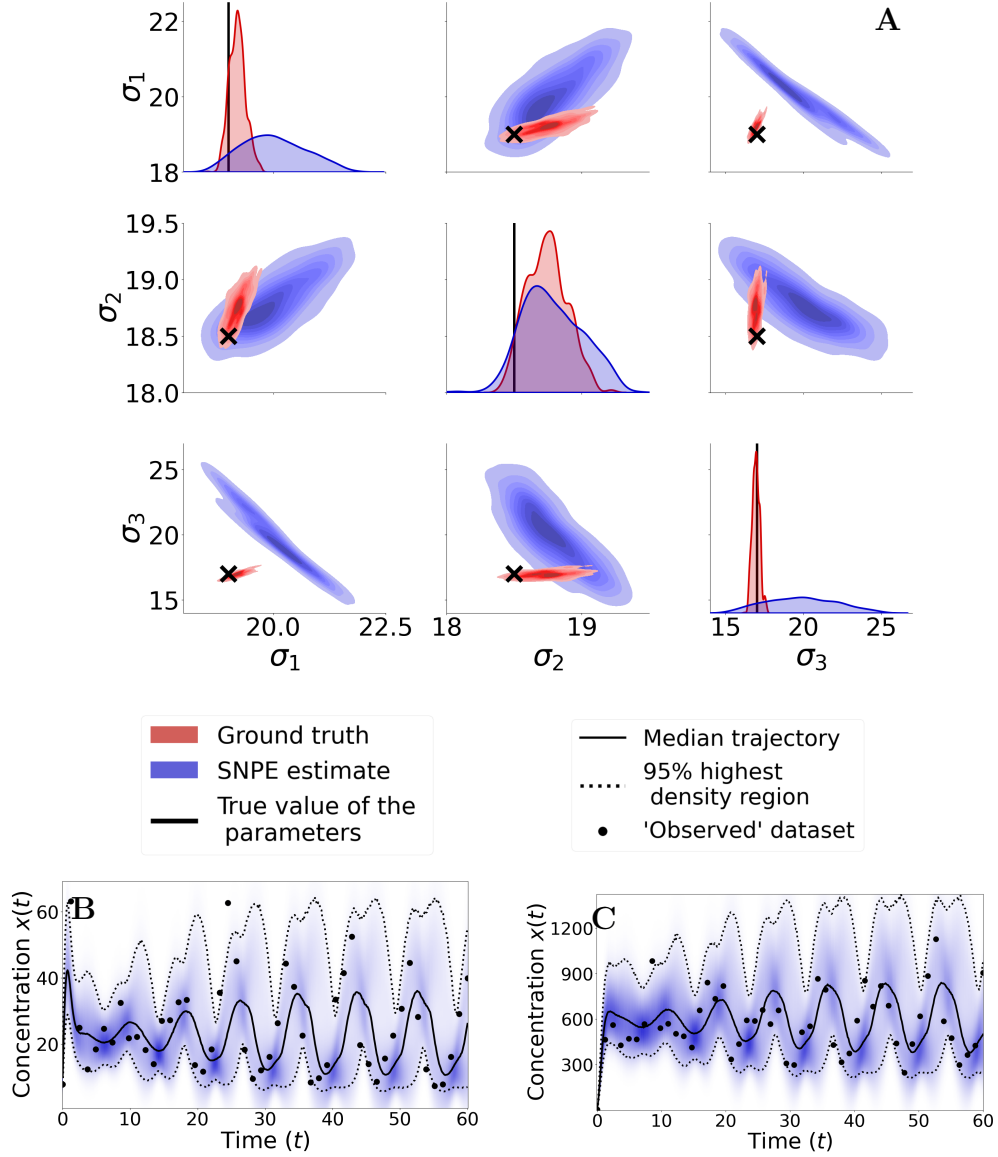


Figure 4.14: Visual comparison between the SNPE estimate using MDN and the ground truth for the ‘observed’ dataset. **A** shows pair plots of the SNPE estimate compared with the ground truth for the three protein translation rates ( $\sigma_1$ ,  $\sigma_2$ ,  $\sigma_3$ ). **B** and **C** show predicted trajectories for mRNA 1 and protein 1, respectively, from PPC on the SNPE estimate.

## Scores and inference times for all architectures

Benchmarking was done as before. For this model, ratios of inference times are also computed and compared across architectures. The standard deviation scores are categorically higher than mean scores which, along with the pair plots, indicates conservative estimates yielded by all architecture. MDN, MADE and MAF yield high and low scores for different parameters in an overall similar fashion. Lastly, the ratios of inference times are compared, showing highest ratio for MAF, and lowest for MDN.

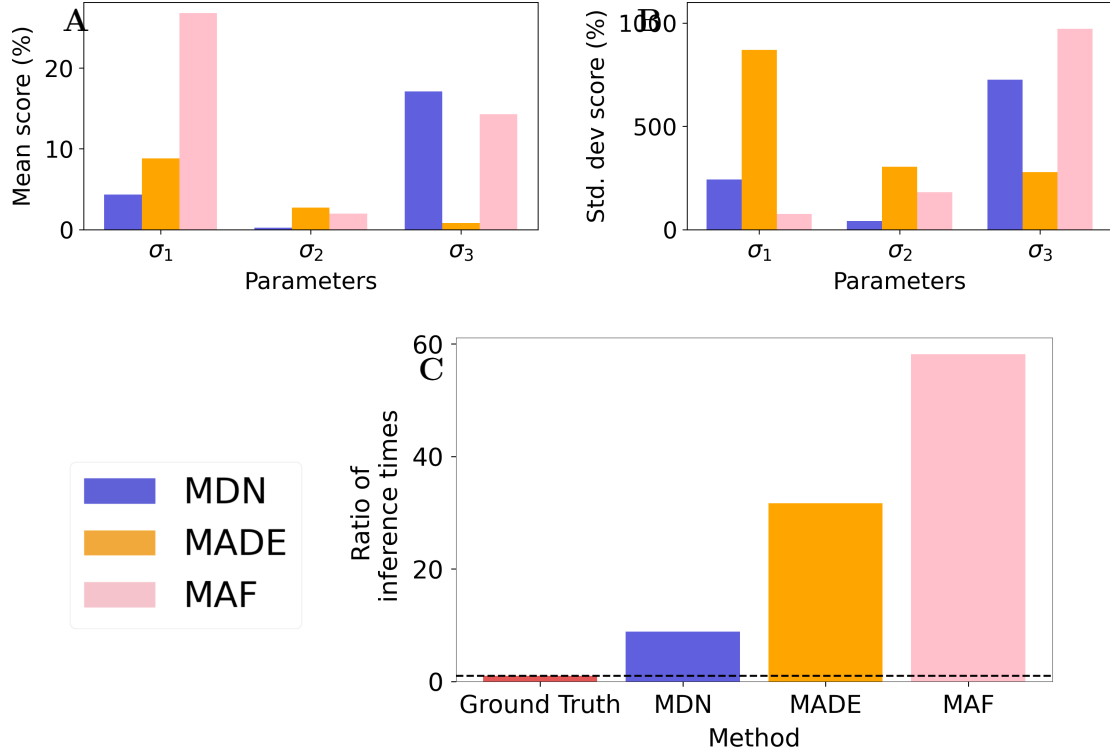


Figure 4.15: Visual comparison between the scores and inference times for SNPE using all architectures.

Based on the results so far, the following can be concluded.

1. MDN and MADE yield reasonable, conservative, fairly consistent (between architectures) estimates, MAF estimate fails PPC,
2. MDN is faster to train than MADE,
3. amortisation is infeasible for this model (results not shown),
4. Ratio of inference times is highest for MAF, making it undesirable as a candidate architecture for this study.

Thus, MDN and MADE are **faster and more accurate** approximators of the joint posterior than MAF. Amortisation appears infeasible. However, the ratio of inference times for both MDN and MADE is appreciably higher than 1.0. This falsifies the study hypothesis, making SNPE undesirable as an inference mechanism for the three parameter repressilator model. Thus, the objectives for this section are met.

## 4.6 Repressilator Model: Six-dimensional model

More complexity was introduced in the form of the six-dimensional version. Here, the parameters considered were the leaky expression rates,  $\chi_1, \chi_2, \chi_3$  and inverse of the binding affinities  $1/k_1, 1/k_2, 1/k_3$ . Since the binding affinity values are small ( $\sim 10^{-3}$ ), inferring a posterior becomes difficult due to closeness to zero. Thus, inverse of the binding affinities were taken as the parameters. Everything else was assumed to be constant. The following is the simplified system of ODEs:

$$\frac{dm_i}{dt} = \frac{120.8}{1 + (k_i \cdot p_j)^3} + \chi_i - 2.2 \cdot m_i \quad (4.9)$$

$$\frac{dp_i}{dt} = \sigma_i \cdot m_i - 0.8 \cdot p_i, \quad \{\sigma_1, \sigma_2, \sigma_3\} = \{19.0, 18.5, 17.0\} \quad (4.10)$$

The architectures are narrowed down to MDN and MADE for higher models, starting with this one. The following figure only shows mRNA 1 and protein 1 trajectories but from two ‘observed’ datasets. PPC for the ground truth on the **first dataset** indicating predictive trajectories for mRNA 1 ( $m_1$ ) and protein 1 ( $p_1$ ) is also shown.

### SNPE with hyperparameter optimisation leads to reasonable estimates for the dataset

With the same inference procedure as before the SNPE estimate using MDN is shown. Similar results are obtained for MADE as well (sec. 6). In both cases, the estimates pass the PPC.

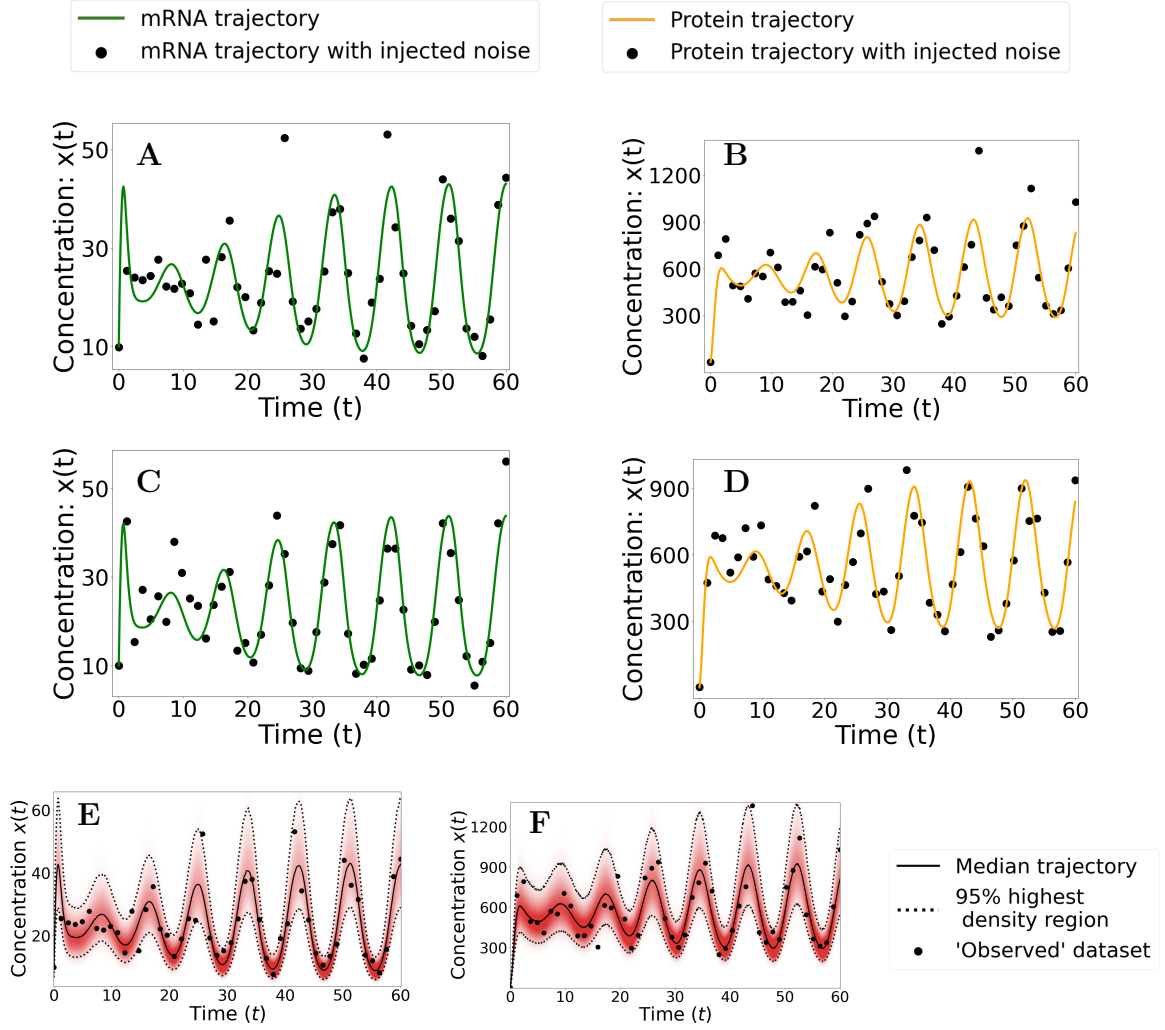


Figure 4.16: ‘Observed’ dataset alongside PPC from the ground truth for the six parameter model. **A** and **B** show ODE solutions injected with noise for mRNA 1 and protein 1, respectively. **E** and **F** show predicted trajectories for mRNA 1 and protein 1 from the PPC on ground truth



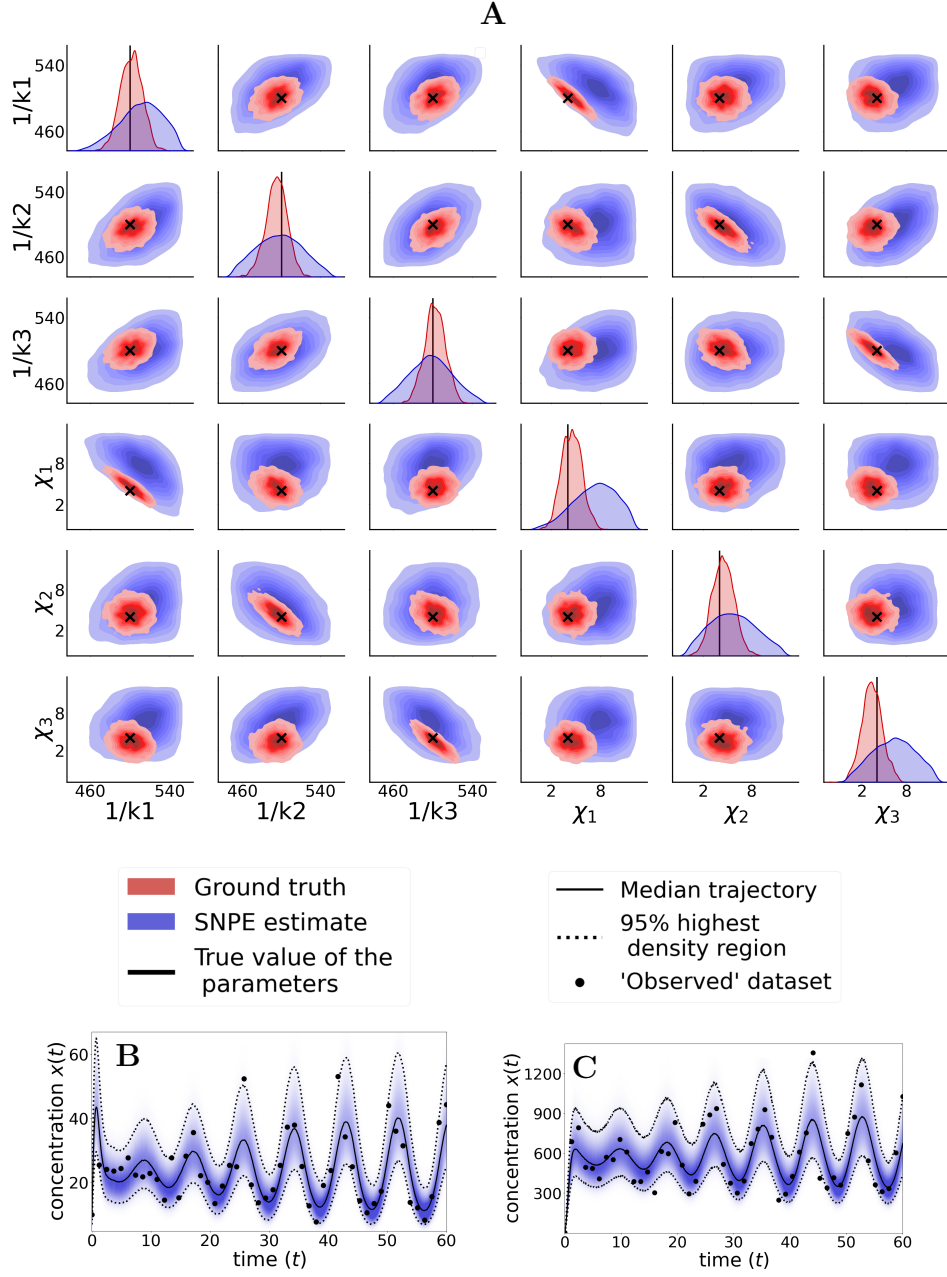


Figure 4.17: Visual comparison between the SNPE estimate using MDN and the ground truth for the ‘observed’ dataset. **A** shows pair plots of the SNPE estimate compared with the ground truth for the inverse binding affinities and leaky expression rates ( $\chi_1$ ,  $\chi_2$ ,  $\chi_3$ ,  $1/k_1$ ,  $1/k_2$ ,  $1/k_3$ ). **B** and **C** show predicted trajectories for mRNA 1 and protein 1, respectively, from PPC on the SNPE estimate.

## Scores and inference times for MDN and MADE

Benchmarking was done as before. Ratio of inference times were also computed and compared across the two architectures. Remarkably, their mean and standard deviation scores are largely similar. Like before, the standard deviation scores are categorically higher than the mean scores (along with pair plots), indicating conservative estimates. Most notably, ratios of inference times have reduced for both the architectures, compared to the three dimensional model (Fig. 4.15). This indicates that neural algorithms mitigate the curse of dimensionality better than MCMC and scale well with complexity. Although this falsifies the primary hypothesis, it supports the auxiliary hypothesis.

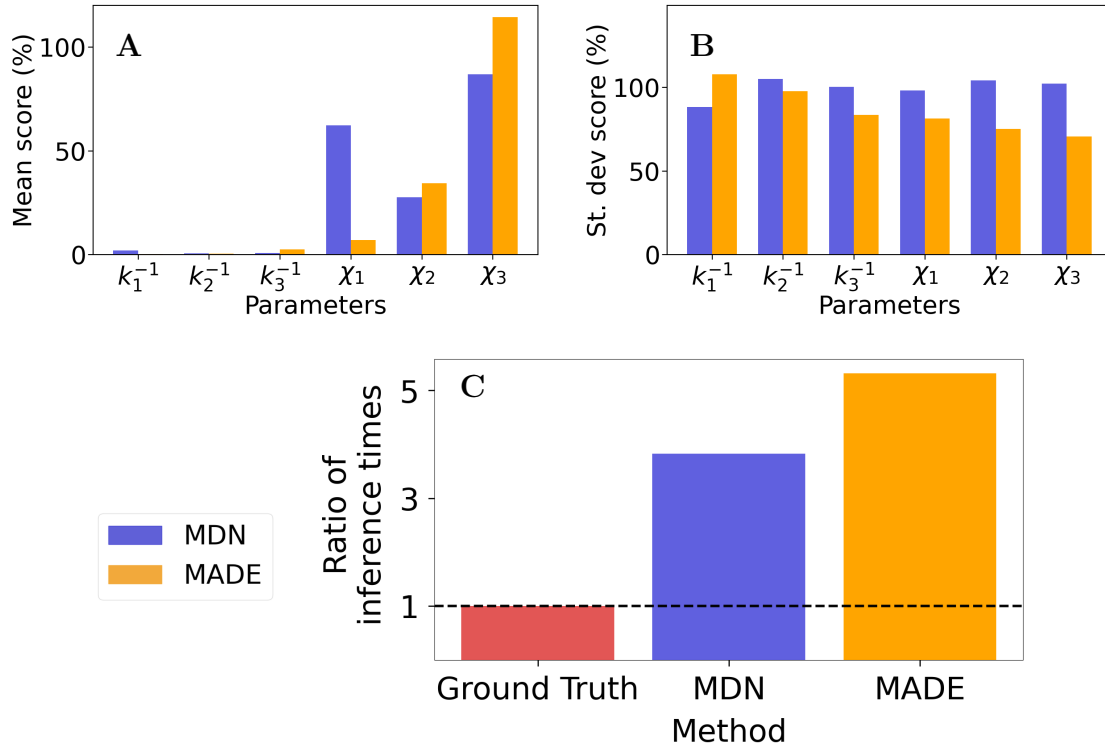


Figure 4.18: Visual comparison between the scores and inference times for SNPE using two architectures.

## NPE yields estimates enabling amortisation for two datasets

Here, NPE estimates are shown only for MDN on two datasets (results for MADE underway). Both pass the PPC.

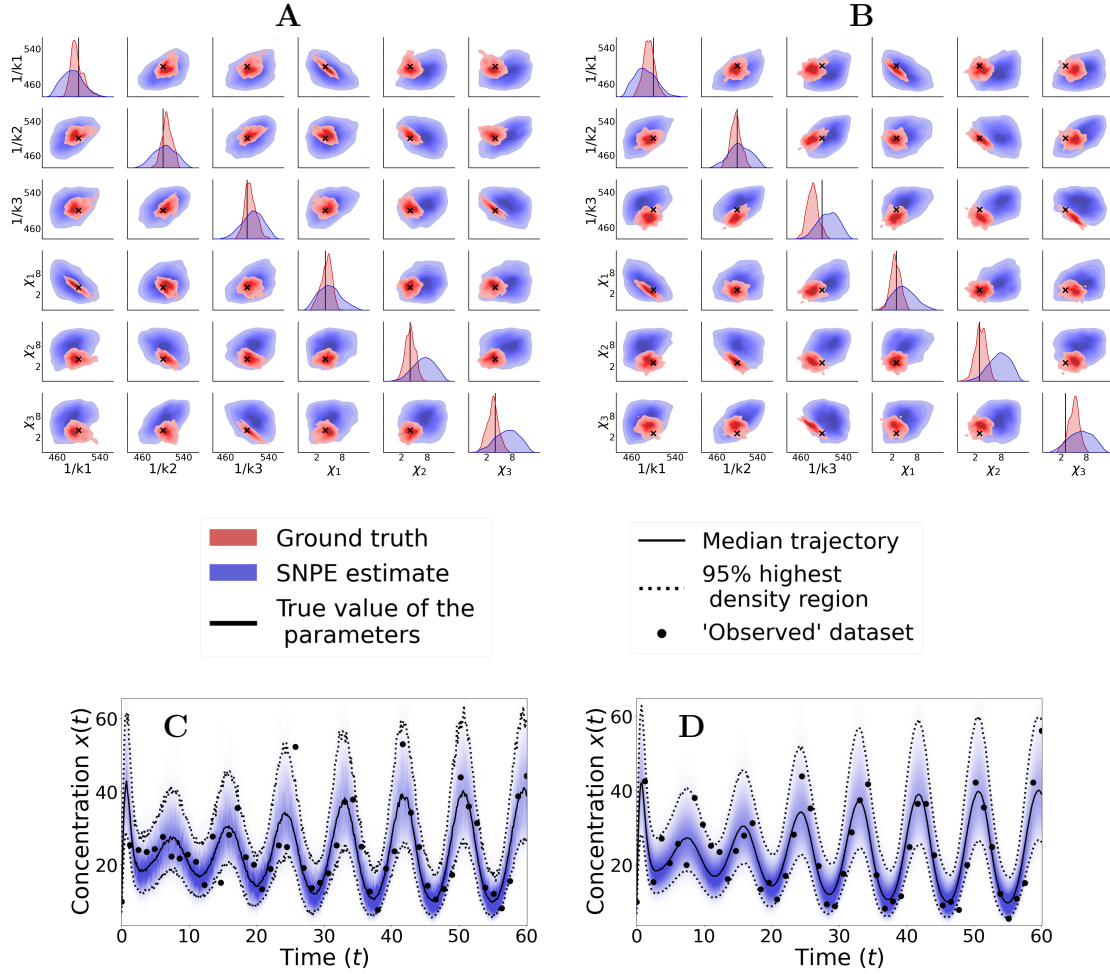


Figure 4.19: Visual comparison between the NPE (enabling amortisation) estimate using MDN and the ground truth for two 'observed' datasets. **A,B** Pair plots of the NPE estimate compared with the ground truth for the inverse binding affinities and leaky expression rates ( $\chi_1, \chi_2, \chi_3, 1/k_1, 1/k_2, 1/k_3$ ). **C, D** Predicted trajectories for mRNA 1 and protein 1, respectively, from PPC on the NPE estimate.

## Comparison of scores and inference times for NPE estimates

In this case, the benchmarking was done by computing the scores for six different datasets using MDN (MADE not shown) for two parameters. Ratio of inference times were also computed and compared for MDN. Again the standard deviation scores are fairly consistent across estimates. However, there are discrepancies in the mean scores for  $\chi_1$ . While NPE can take longer than SNPE, if successful (wth thorough benchmarking), the average inference time per dataset can dramatically reduced. This shown in the inference times plot. Average inference time per dataset, is computed by dividing the ratio of inference time by the number of datasets. Even for this model, where ratio of inference times is relatively high, amortisation holds the potential to dramatically reduce or 'amortise' the inference time (sec. 3.6.3).

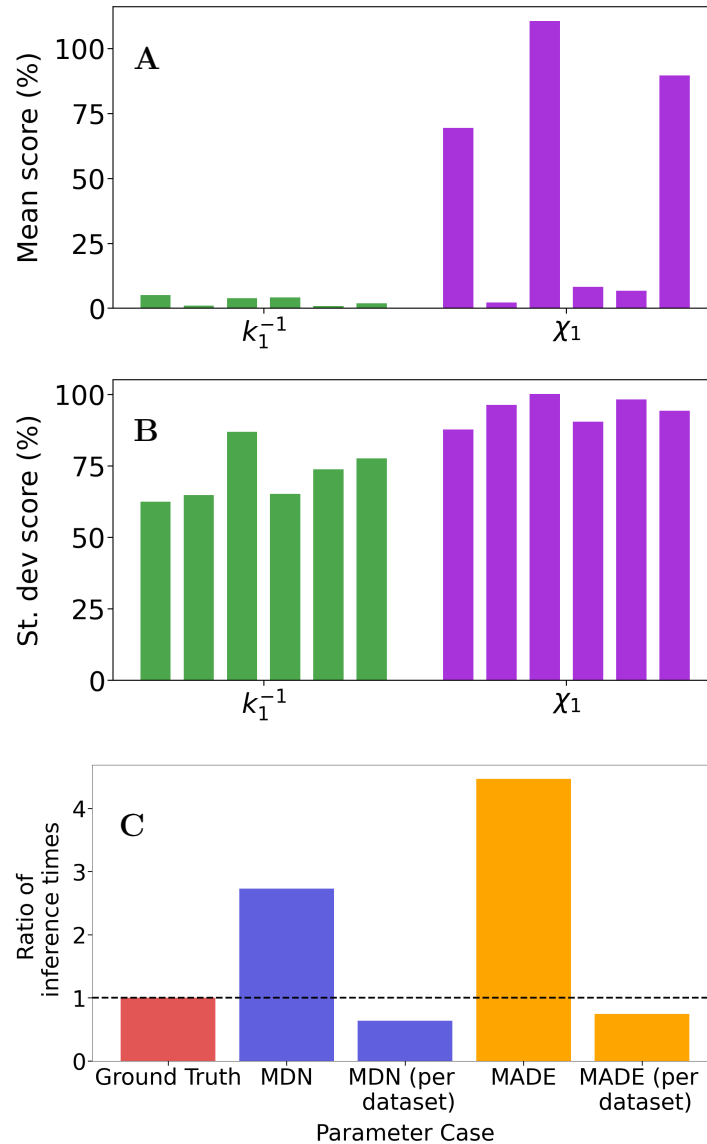


Figure 4.20: Visual comparison between the scores for SNPE using MDN and inference times using both MDN and MADE. **A**, **B** Comparison of mean and standard deviation score for two parameters across the six NPE estimates. **C** Inference times for NPE estimates both in total and averaged over the datasets.

Based on the results so far, the following can be concluded:

1. MDN and MADE yield reasonable, conservative, fairly consistent and robust (6) estimates,
2. MDN remains faster to train than MADE,
3. Amortisation has the potential to substantially reduce the average inference time per dataset, eventually, speeding up inference relative to ground truth inference by MCMC,
4. Ratios of inference times for both MADE and MDN are greater than unity, falsifying the primary hypothesis,
5. Ratios of inference times for both MADE and MDN have improved from the previous case, supporting the auxiliary hypothesis.

Thus, MDN and MADE are accurate approximators of the ground truth. Amortisation holds potential to support the auxiliary hypothesis as well as the primary hypothesis. Inference time for SNPE estimates falsify the primary hypothesis, while supporting the auxiliary hypothesis. Thus, the objectives for this section are met.

## 4.7 Repressilator Model: Nine-dimensional model

With positive results in hand, the complexity is increased further by adding maximum increment over leaky expression rates,  $\alpha_1$ ,  $\alpha_2$  and  $\alpha_3$ , to the list of parameters from before ( $\{\chi_1, \chi_2, \chi_3, 1/k_1, 1/k_2, 1/k_3\}$ ). Everything else was assumed to be constant. The following is the simplified system of ODEs:

$$\frac{dm_i}{dt} = \frac{\alpha_i}{1 + (k_i \cdot p_j)^3} + \chi_i - 2.2 \cdot m_i \quad (4.11)$$

$$\frac{dp_i}{dt} = \sigma_i \cdot m_i - 0.8 \cdot p_i, \quad \{\sigma_1, \sigma_2, \sigma_3\} = \{19.0, 18.5, 17.0\} \quad (4.12)$$

As before, the following figure only shows the ‘observed’ dataset in terms of mRNA 1 and protein 1 trajectories. PPC for the ground truth indicating predictive trajectories for mRNA 1 ( $m_1$ ) and protein 1 ( $p_1$ ) is also shown.

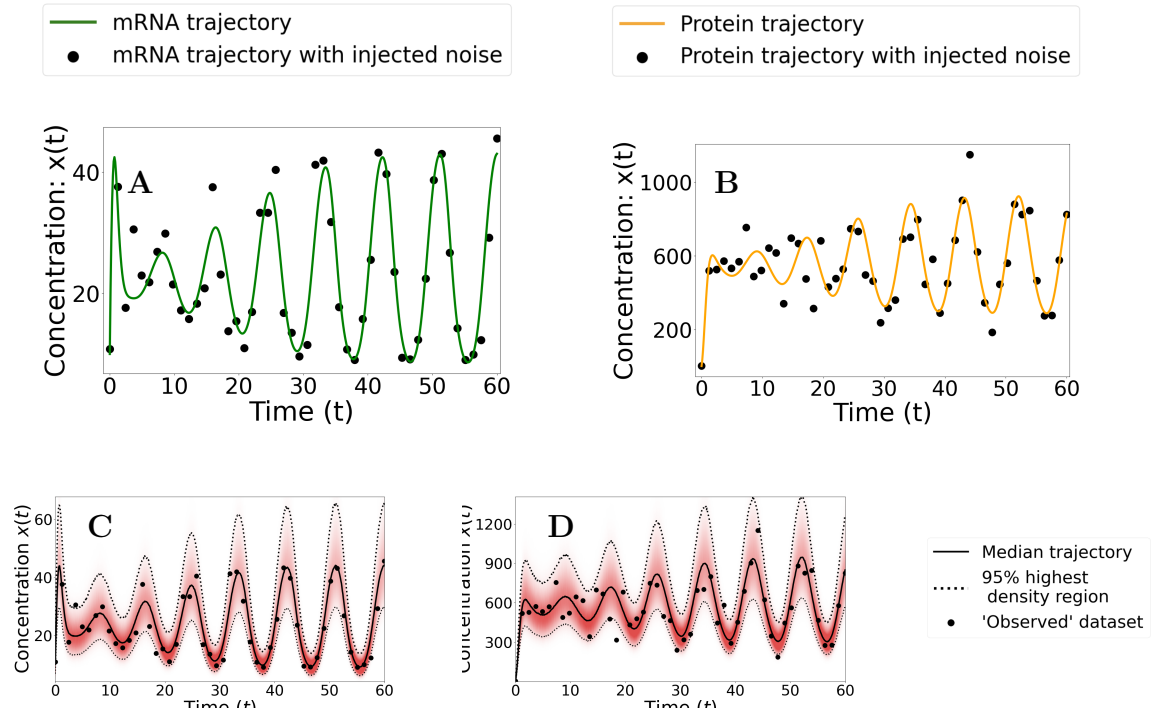


Figure 4.21: ‘Observed’ dataset alongside PPC from the ground truth for the nine parameter model. **A** and **B** show ODE solutions injected with noise for mRNA 1 and protein 1, respectively. **C** and **D** show predicted trajectories for mRNA 1 and protein 1 from the PPC on ground truth.

## SNPE with hyperparameter optimisation leads to reasonable estimates for the dataset

Inference procedure implemented as before. Results only shown for MDN (results for MADE are underway). The SNPE estimate is conservative and passes the PPC.

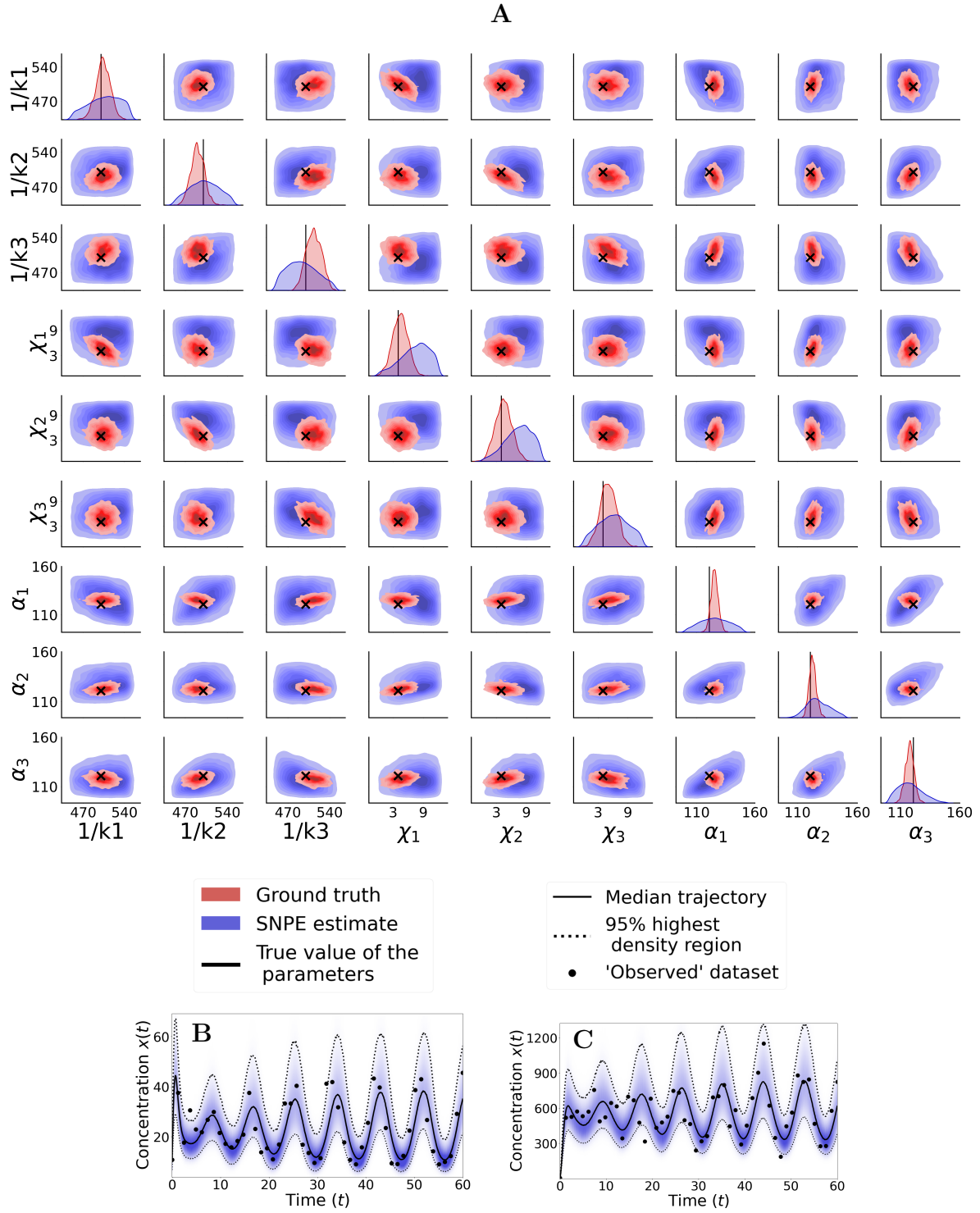


Figure 4.22: Visual comparison between the SNPE estimate using MDN and the ground truth for the ‘observed’ dataset. **A** shows pair plots of the SNPE estimate compared with the ground truth for the inverse binding affinities, leaky expression rates and maximum increments over leaky expression ( $\chi_1, \chi_2, \chi_3, 1/k_1, 1/k_2, 1/k_3, \alpha_1, \alpha_2, \alpha_3$ ). **B** and **C** show predicted trajectories for mRNA 1 and protein 1, respectively, from PPC on the SNPE estimate.

## Scores and inference times for MDN

Benchmarking and ratio of inference times shown for MDN (results for MADE underway). Interestingly, the mean scores are again disproportionately higher for the leaky expression rates ( $\chi_i$ ). This is reminiscent of the previous case (Fig. 4.20). This suggests that it might be relatively more difficult to infer joint posterior densities over some parameter compared to other parameters. As before, the standard deviation scores are categorically higher. Finally, the inference time ratio of MDN has further reduced supporting the auxiliary hypothesis.

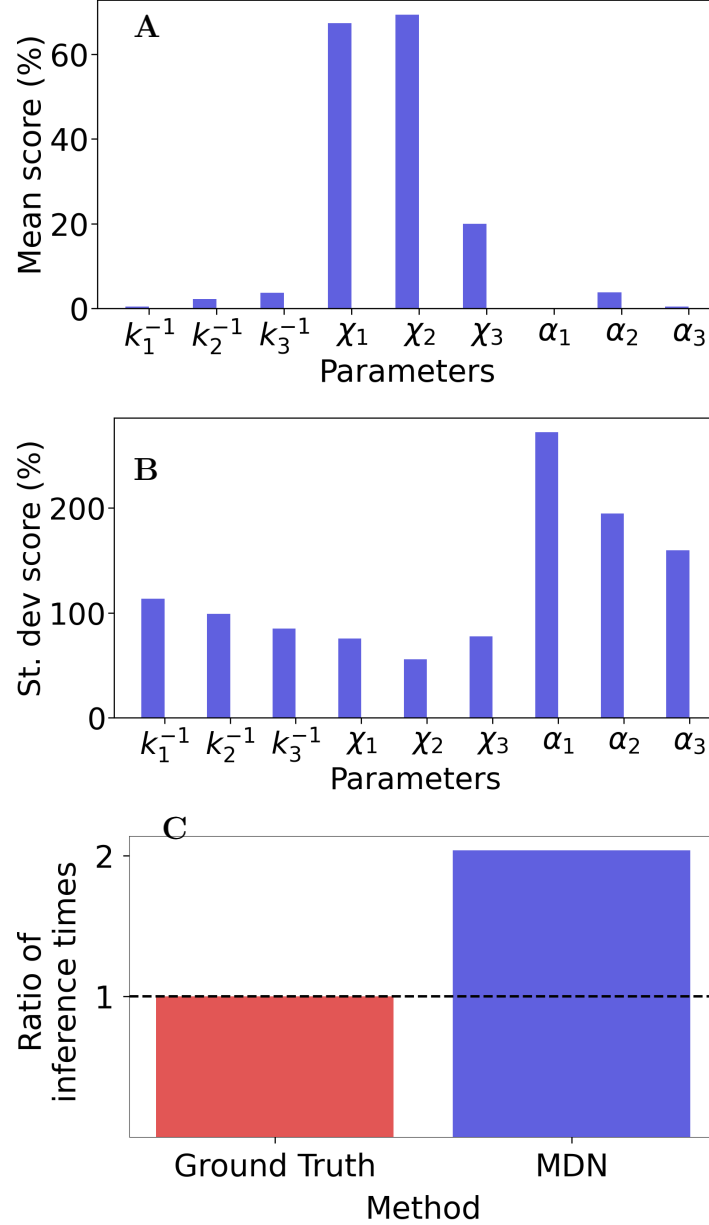


Figure 4.23: Scores and inference times for SNPE using MDN.

Based on the results so far, the following can be concluded:

1. MDN yields reasonable, conservative, fairly consistent and robust (sec. 4.7 estimates,
2. Mean scores can be consistently higher for some parameters over others, indicating



difficulty in their inference,

3. Amortisation is not shown yet,
4. Ratio of inference times for MDN remains higher than unity, falsifying the primary hypothesis
5. Ratio of inference times for MDN has further improved from the previous case, supporting the auxiliary hypothesis,

Thus, MDN is an accurate approximator of the ground truth for this model. Amortization results are underway. Inference time for the SNPE estimate falsifies the primary hypothesis, while supporting the auxiliary hypothesis. Thus, the objectives for this section are met.

## 4.8 Repressilator Model: Twelve-dimensional version

The complexity is increased further by adding the Hill coefficients,  $n_1, n_2, n_3$ , to the list of parameters from before ( $\{\alpha_1, \alpha_2, \alpha_3, \chi_1, \chi_2, \chi_3, 1/k_1, 1/k_2, 1/k_3\}$ ). Everything else was assumed to be constant. Thus, following is the simplified system of ODEs:

$$\frac{dm_i}{dt} = \frac{\alpha_i}{1 + (k_i \cdot p_j)^{n_i}} + \chi_i - 2.2 \cdot m_i \quad (4.13)$$

$$\frac{dp_i}{dt} = \sigma_i \cdot m_i - 0.8 \cdot p_i, \quad \{\sigma_1, \sigma_2, \sigma_3\} = \{19.0, 18.5, 17.0\} \quad (4.14)$$

Again as before, the following figure only shows the ‘observed’ dataset in terms of mRNA 1 and protein 1 trajectories. PPC for the ground truth on the **first dataset** indicating predictive trajectories for mRNA 1 ( $m_1$ ) and protein 1 ( $p_1$ ) is also shown.

### SNPE with hyperparameter optimisation leads to reasonable estimates for the dataset

Here, the results are shown only for MADE (MDN shown in (sec. 4.7)). The SNPE estimate is conservative and passes the PPC.

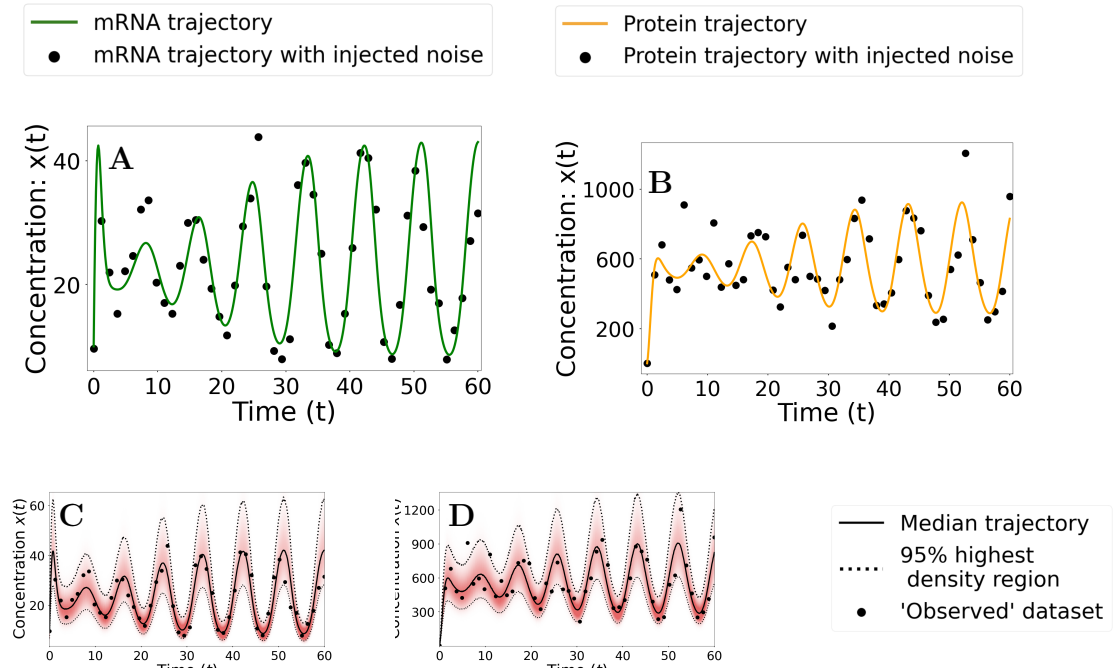


Figure 4.24: ‘Observed’ dataset alongside PPC from the ground truth for the twelve parameter model. **A** and **B** show ODE solutions injected with noise for mRNA 1 and protein 1, respectively. **C** and **D** show predicted trajectories for mRNA 1 and protein 1 from the PPC on ground truth.

## Scores and inference times for MDN and MADE

Benchmarking and the ratios of inference times shown for MDN and MADE. The scores are consistent with the previous results. Notably, the scores for MADE and MDN estimates are consistent with each other and the ratios of inference times have further reduced for both, supporting the auxiliary hypothesis.

Based on the results so far, the following can be concluded:

1. MDN and MADE both yield reasonable, conservative and consistent estimates,
2. Patterns in mean and standard deviation scores are consistent across the models,
3. Ratios of inference times for MDN and MADE remain higher than unity, falsifying the primary hypothesis,
4. Ratios of inference times for both MADE and MDN have further improved from the previous case, supporting the auxiliary hypothesis,

Thus, MDN and MADE are accurate approximators of the ground truth for this model. Amortization results are underway. Inference time for the SNPE estimate falsifies the primary hypothesis, while supporting the auxiliary hypothesis. Thus, the objectives for this section are met.

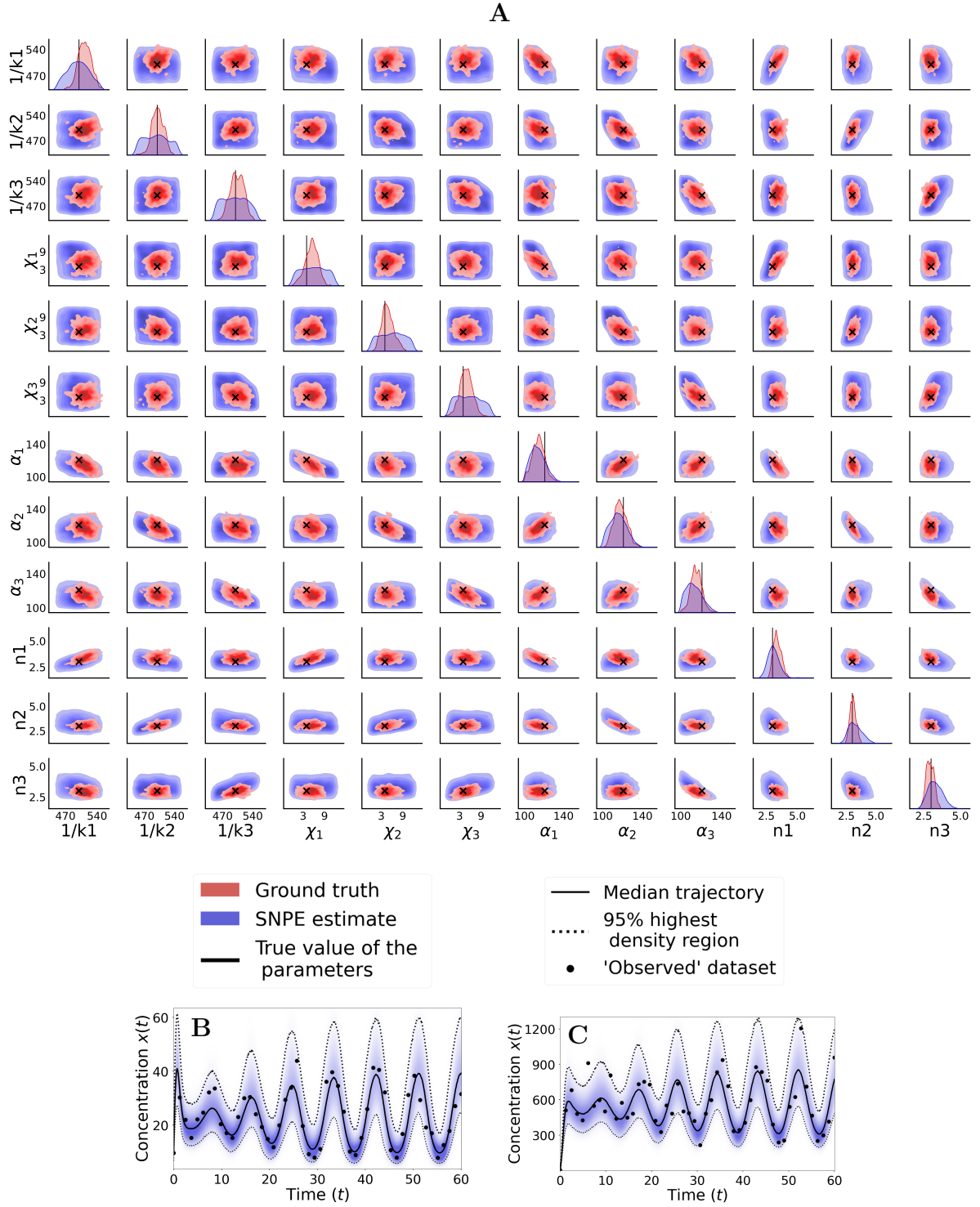


Figure 4.25: Visual comparison between the SNPE estimate using MADE and the ground truth for the ‘observed’ dataset. **A** shows pair plots of the SNPE estimate compared with the ground truth for the inverse binding affinities, leaky expression rates, maximum increments over leaky expression and Hill’s coefficients ( $\chi_1$ ,  $\chi_2$ ,  $\chi_3$ ,  $1/k_1$ ,  $1/k_2$ ,  $1/k_3$ ,  $\alpha_1$ ,  $\alpha_2$ ,  $\alpha_3$ ,  $n_1$ ,  $n_2$ ,  $n_3$ ). **B** and **C** show predicted trajectories for mRNA 1 and protein 1, respectively, from PPC on the SNPE estimate.

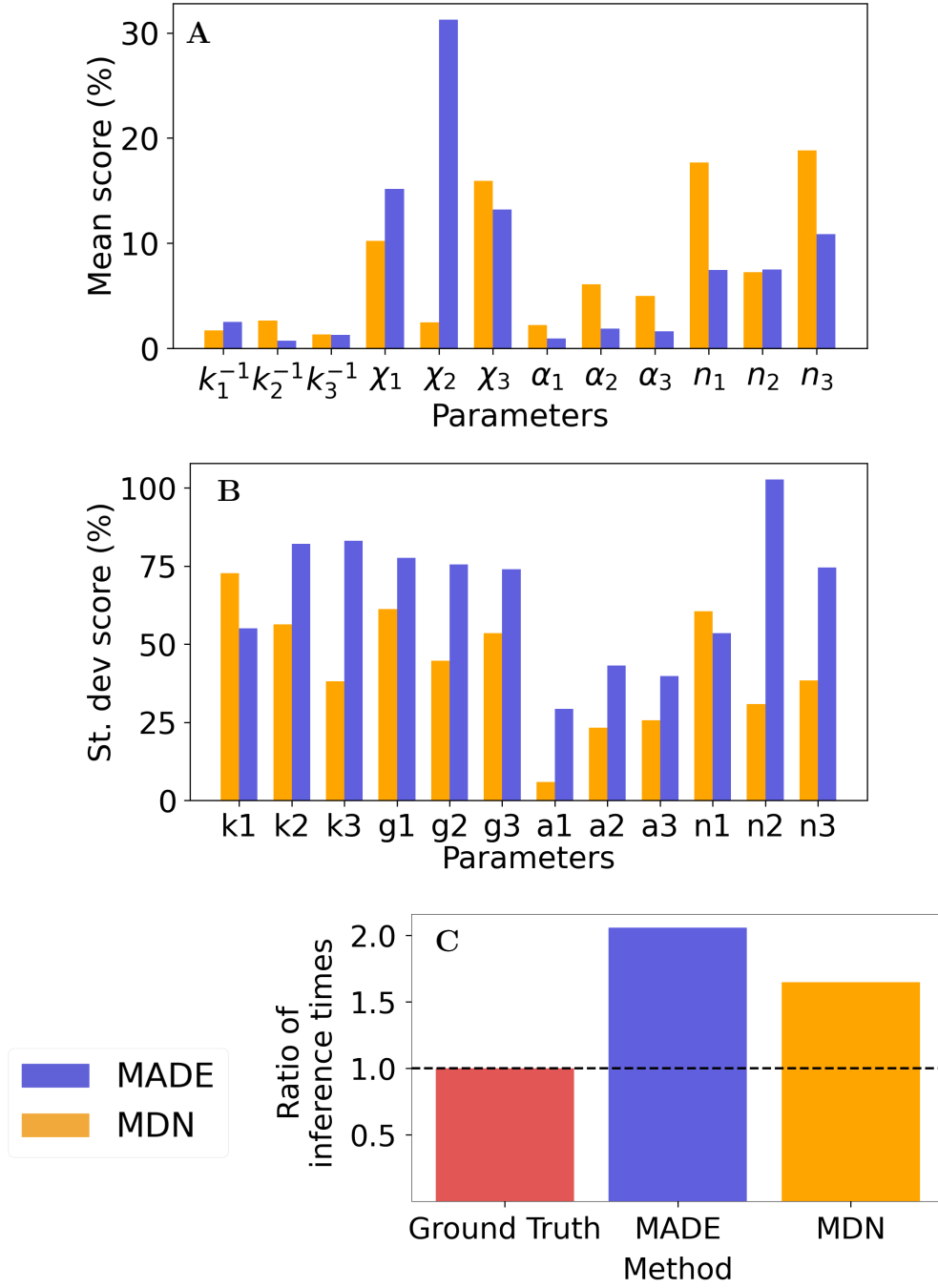


Figure 4.26: Visual comparison between the scores and inference times for SNPE using two architectures.

## 4.9 Repressilator Model: 15-dimensional version

Lastly, the protein translation rates (parameters of the three dimensional version) are added back to the existing parameters ( $\{n_1, n_2, n_3, \alpha_1, \alpha_2, \alpha_3, \chi_1, \chi_2, \chi_3, 1/k_1, 1/k_2, 1/k_3\}$ ). Everything else was assumed to be constant. The following is the simplified system of ODEs:

$$\frac{dm_i}{dt} = \frac{\alpha_i}{1 + \left(\frac{1}{k_i} \cdot p_j\right)^{n_i}} + \chi_i - 2.2 \cdot m_i \quad (4.15)$$

$$\frac{dp_i}{dt} = \sigma_i \cdot m_i - 0.8 \cdot p_i. \quad (4.16)$$

Finally, the following figure only shows the ‘observed’ dataset in terms of mRNA 1 and protein 1 trajectories. PPC for the ground truth indicating predictive trajectories for mRNA 1 ( $m_1$ ) and protein 1 ( $p_1$ ) is also shown.

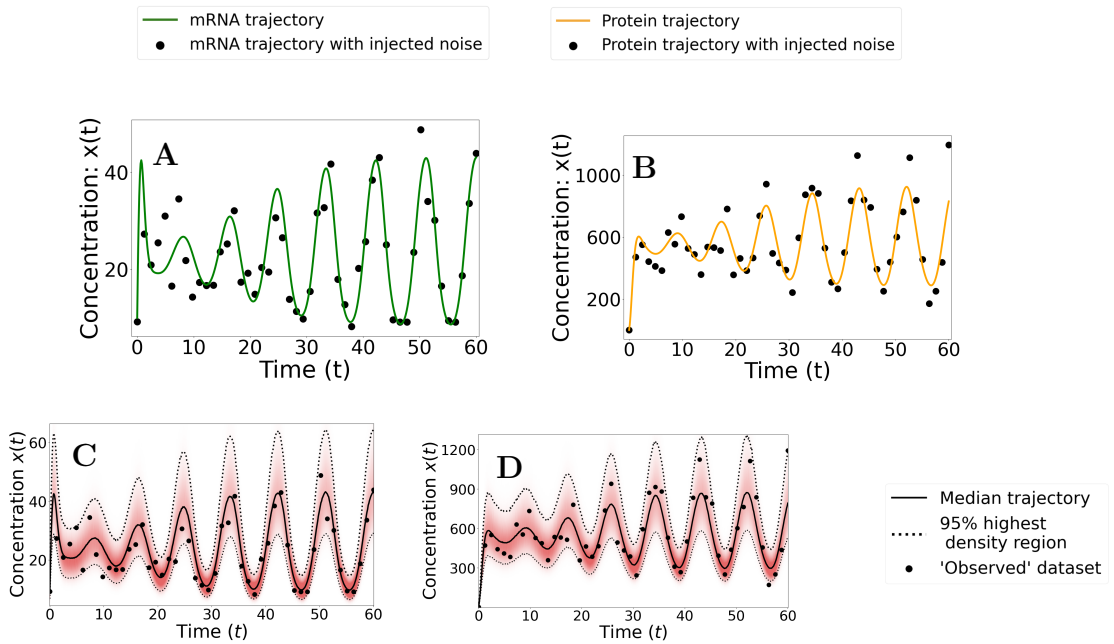


Figure 4.27: ‘Observed’ dataset alongside PPC from the ground truth for the 15 parameter model. **A** and **B** show ODE solutions injected with noise for mRNA 1 and protein 1, respectively. **C** and **D** show predicted trajectories for mRNA 1 and protein 1 from the PPC on ground truth.

## SNPE with hyperparameter optimisation leads to reasonable estimates for the dataset

Here, only the PPC pots are shown for MADE (results for MDN are underway). Pair plots can be found in sec. 6.4. Again, the SNPE estimate passes the PPC.

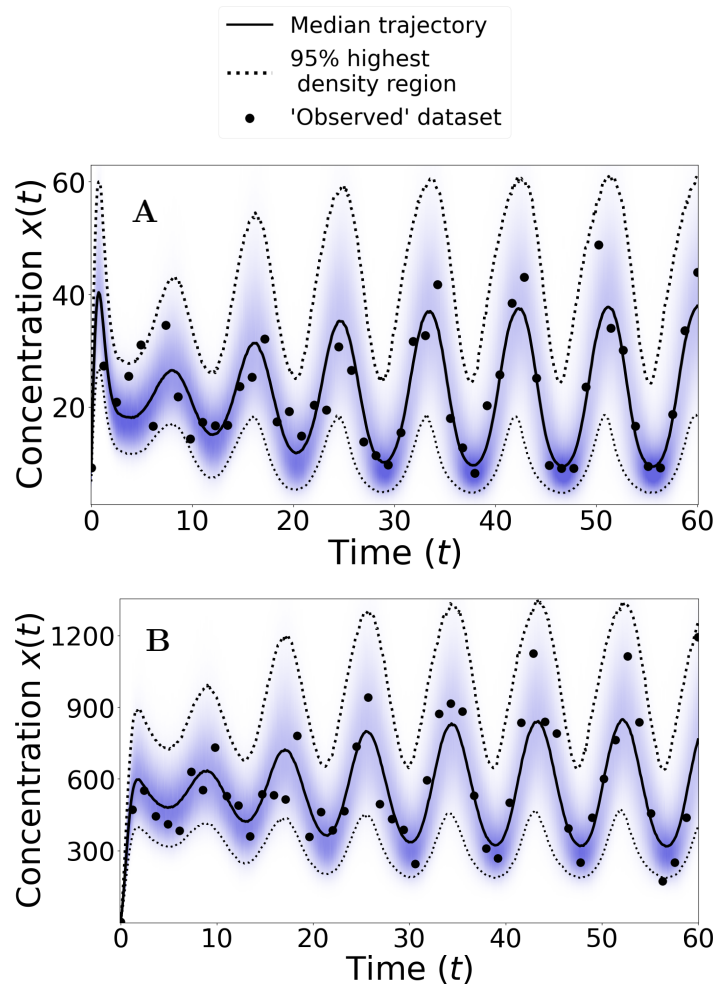


Figure 4.28: Visual comparison between the SNPE estimate using MADE and the ground truth for the 'observed' dataset. **A** and **B** show predicted trajectories for mRNA 1 and protein 1, respectively, from PPC on the SNPE estimate.

## Inference times for all architectures and scores

Benchmarking and the ratios of inference times shown MADE (results for MDN are underway). The scores are consistent with the previous results. Notably, the scores for MADE and MDN estimates are consistent with each other and the ratios of inference times have further reduced for both, supporting the auxiliary hypothesis.

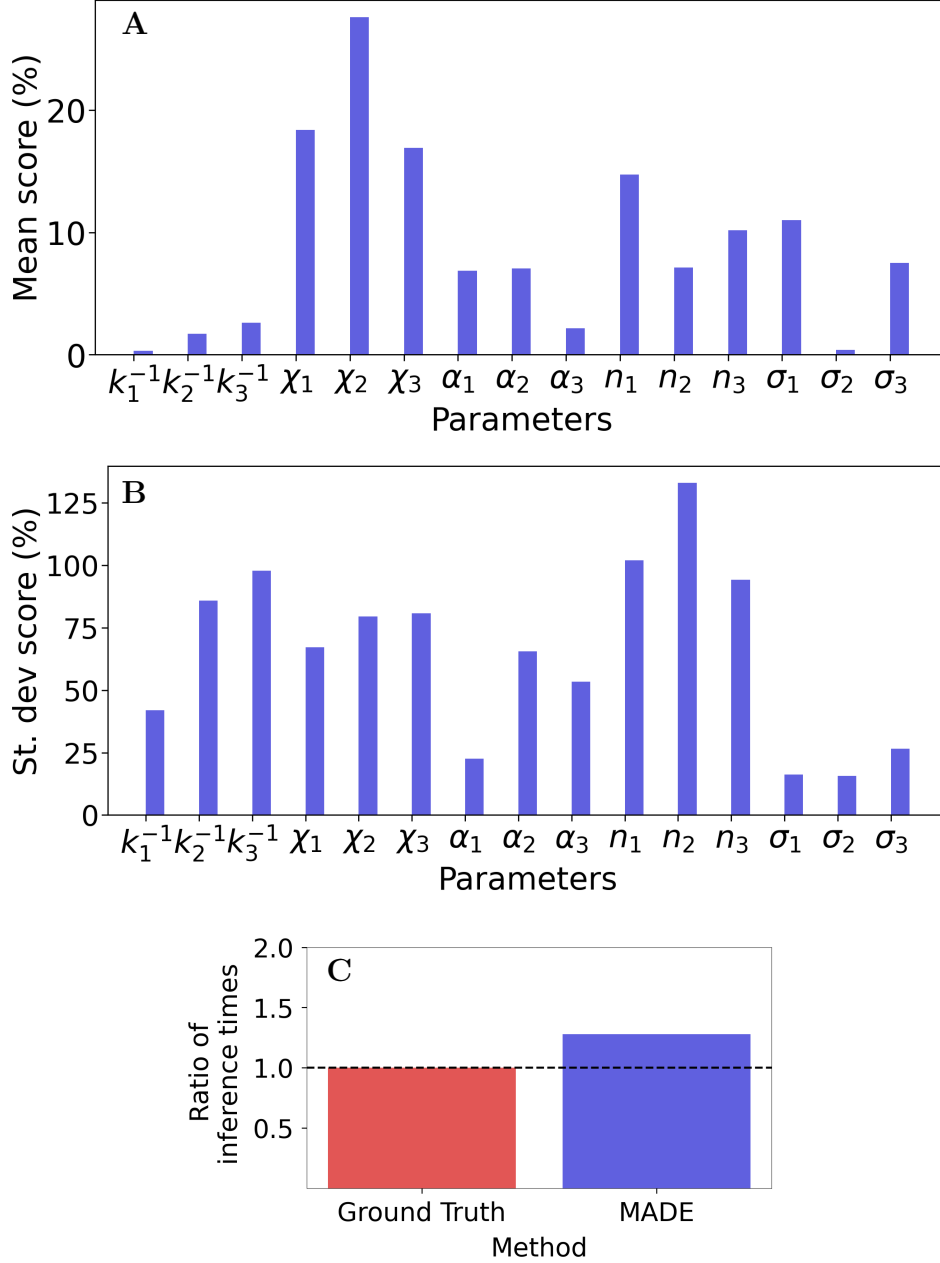


Figure 4.29: Visual comparison between the scores and inference times for SNPE using MADE.

1. MADE yields reasonable, conservative and robust (sec. 4.9) estimates,
2. Patterns in mean and standard deviation scores are consistent across the models,
3. Ratio of inference times for MADE remains higher than unity, falsifying the primary hypothesis,

4. Ratio of inference times MADE has further improved from the previous case, supporting the auxiliary hypothesis,

Thus, MADE is an accurate approximator of the ground truth for this model. Amortization results are underway. Inference time for the MADE estimate falsifies the primary hypothesis, while supporting the auxiliary hypothesis. Thus, the objectives for this section are met.

## 4.10 Inference time comparison across models

This chapter concludes with comparison of the ratios of inference times of the fastest estimates for each of the simpler models.

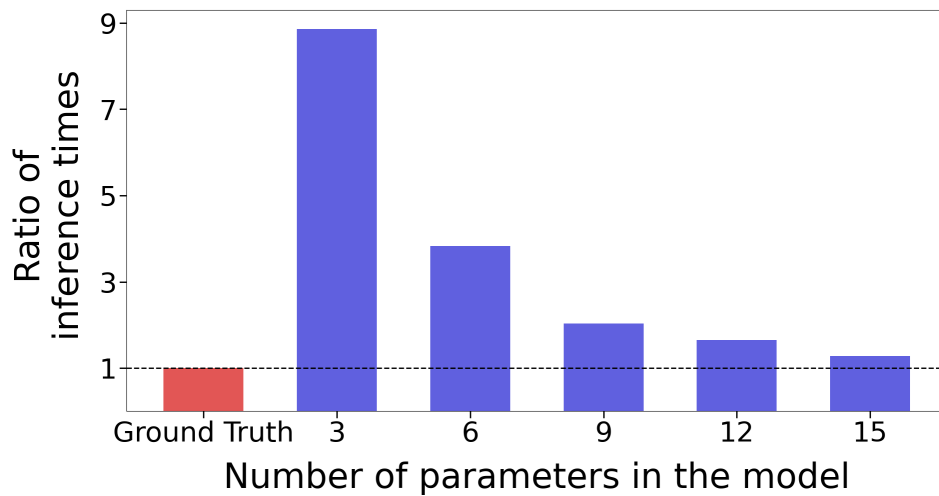


Figure 4.30: Visual comparison between the inference times across models

This figure supports the auxiliary hypothesis and gives confidence in MDN and MADE for being extended to the 21 parameter model (results are underway).



# Chapter 5

## Discussion

As a proof of concept, this study explored the establishment of a parameter inference pipeline for the Repressilator model using synthetically generated data. It was hypothesized that the neural network based algorithms can accurately infer the joint posterior, faster than the state-of-the-art algorithms. Starting with the exponential model (one dimensional, sec. 4.2 and two dimensional sec. 4.3 versions), parameter inference was subsequently carried out for the various parameterizations of the repressilator model (sec. 4.5, 4.6, 4.7, 4.8 and 4.9). With numerous positive results in hand, the conclusions, limitations, future work and scope are discussed.

### 5.1 Conclusions

This study finds MDN and MADE as promising candidates for GRN inference for a system of many parameters, given a topology, a system of ODEs and prior distributions over the parameters. They hold promise to reduce the ratios of inference times below zero for the 21 parameter model and average inference time per dataset in the case of amortization.

### 5.2 Limitations

Following are the limitations of this study:

1. One of the major limitations of this study is the inability to establish an ‘accurate’ enough joint posterior estimate from SNPE, while the scores help with comparison, they by themselves are not sufficient to indicate whether an SNE estimate is accurate enough,
2. Another major limitation is difficulty with diagnostics for poor estimates/those which fail the PPCs (sec. 4.3). It becomes challenging to troubleshoot such estimates in a principled manner,
3. Noise is not inferred for repressilator models, which could pose additional computational cost related challenges,
4. The main inference phase is about five-ten times slower than the hyperparameter optimization phase, indicating slow training and convergence for the neural

network algorithms. This could be due to unoptimized implementation of the python scripts,

5. Effects of number of simulations and the number of rounds was not systematically tested. Thus, it is possible that estimates as good as shown in the study could be obtained faster,
6. `pymc` yields the MCMC chain for a specified number of samples. If the number of samples is too low, the algorithm does not converge. However, if the number of samples is set higher than necessary, the algorithm does converge, but takes longer. This was not systematically tested. Thus, it is possible that the ratios of inference times shown are actually higher because `pymc` was run longer than necessary,
7. The current implementation of these algorithms requires a lot of resources ( $\sim 100$  CPUs,  $\sim 100\text{gb}$  memory). This precludes their accessibility and dissemination, even if successfully implemented.
8. In the previous work, various methods were used to pre-process the data and simulations to improve the inference, only one of such methods was used in this study.

## 5.3 Future work

The following directions constitute future work:

1. Extending the inference procedure to the 21 parameter model, using both MDN and MADE for SNPE and NPE (enabling amortisation),
2. A more thorough analysis of the performance of MDN and MADE for every one of the simpler models, with benchmarking for SNPE and NPE,
3. Evaluating the degree to which the ratio of inference times per dataset could be improved for amortization, by using numerous ‘observed’ datasets,
4. Analyzing the effects of the number of simulations and rounds on the SNPE inference,
5. Finding the shortest ground truth inference times for all the models for fair comparisons with the neural methods,
6. Employing other ways to pre-process the ‘observed’ data such as using convolutional neural networks as was done in the previous work,
7. Utilizing more optimized frameworks and libraries to implement the same inference procedure, using lesser memory and CPUs,
8. With a successful pipeline, carrying out inference for real biological data of concentrations of the species such as for natural GRNs, with known topologies and hypothesized systems of ODEs describing the dynamics.

## 5.4 Scope

This is where this study gets embedded in the landscape of biology. The scope of the results for this project is two-fold, assuming a well-established, successful inference pipeline:

1. Engineering of artificial GRNs: With the uncertainties on various parameters quantified, there are direct implications for engineering efforts such as for the repressilator. The mean value of the inferred joint posterior along with the 95% highest posterior density region provides a credible interval over the parameters. This could act as the specification of the engineered system, preventing the need to tinker around with the various parameter values such as binding affinities by trying out different promoters through hit and trial. This could greatly improve the reproducibility and robustness of the engineered systems as it is particularly pertinent to the biological systems. This could be done in a sequential manner, enabling the design-build-test engineering cycle, which is one of the aims of synthetic biology. For instance, in the paper Elowitz and Leibler (2000), the authors started with rough ranges of the parameter values to engineer the repressilator. With the pipeline of this study in place, engineering of GRNs could be done in a more principled manner by incorporating experimental to further refine the parameter estimates by using credible intervals and quantifying the noise.
2. Deeper understanding natural GRNs: Firstly, a joint posterior on the parameters could give information on correlations between the different parameters. These could be utilized to understand which parameters vary with the others. This has implications for which parameter sets are most sensitive mutations. For example, say the binding affinity of a transcription changes due to mutations in the promoter, its translation rate may change as well, compromising the behaviour of interest. The joint posterior essentially encodes all possible information about the system, based on data and our assumptions. This can be used to draw numerous kinds of other downstream inferences. Secondly, using SNPE enables model selection. In particular, given a network topology, various hypothesized systems of ODEs can be individually tested. Given a real dataset, SBI can be implemented for each hypothesized model. Once the inference is done, PPCs can be conducted on estimates from each model to falsify the unrealistic ones. Further more, since SNPE yields probability values for the joint posterior estimate, given the data, the ratios of probabilities of different models given the data can be employed as a part of Bayesian model selection to select the most probable model explaining observed data.

# References

- U. Alon. *An introduction to systems biology: design principles of biological circuits*. Chapman and Hall/CRC, 2019.
- R. C. Aster, B. Borchers, and C. H. Thurber. *Parameter estimation and inverse problems*. Elsevier, 2018.
- A.-L. Barabasi and Z. N. Oltvai. Network biology: understanding the cell’s functional organization. *Nature reviews genetics*, 5(2):101–113, 2004.
- M. A. Beaumont, W. Zhang, and D. J. Balding. Approximate bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.
- Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
- S. A. Benner and A. M. Sismour. Synthetic biology. *Nature reviews genetics*, 6(7):533–543, 2005.
- J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.
- C. M. Bishop. Mixture density networks. 1994.
- M. G. Blum. Approximate bayesian computation: a nonparametric perspective. *Journal of the American Statistical Association*, 105(491):1178–1187, 2010.
- S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.
- P. Cahan, H. Li, S. A. Morris, E. L. Da Rocha, G. Q. Daley, and J. J. Collins. Cellnet: network biology applied to stem cell engineering. *Cell*, 158(4):903–915, 2014.
- Caltech Archives. Caltech Archives Digital Collections: Image 1.10-29. Online resource. URL <https://digital.archives.caltech.edu/collections/Images/1.10-29/>. Accessed: 1988-05-11.
- K. C. Chen, L. Calzone, A. Csikasz-Nagy, F. R. Cross, B. Novak, and J. J. Tyson. Integrative analysis of cell cycle control in budding yeast. *Molecular biology of the cell*, 15(8):3841–3862, 2004.
- S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995.

- K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.
- K. Cranmer, J. Brehmer, and G. Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.
- A. Der Kiureghian and O. Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.
- A. J. Dobson and A. G. Barnett. *An introduction to generalized linear models*. Chapman and Hall/CRC, 2018.
- M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 2000.
- A. Gelman and D. B. Rubin. Markov chain monte carlo methods in biostatistics. *Statistical methods in medical research*, 5(4):339–355, 1996.
- M. Germain, K. Gregor, I. Murray, and H. Larochelle. Made: Masked autoencoder for distribution estimation. In *International conference on machine learning*, pages 881–889. PMLR, 2015.
- D. K. Goode, N. Obier, M. Vijayabaskar, M. Lie-A-Ling, A. J. Lilly, R. Hannah, M. Lichtinger, K. Batta, M. Florkowska, R. Patel, et al. Dynamic gene regulatory networks drive hematopoietic specification and differentiation. *Developmental cell*, 36(5):572–587, 2016.
- I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.
- D. Greenberg, M. Nonnenmacher, and J. Macke. Automatic posterior transformation for likelihood-free inference. In *International conference on machine learning*, pages 2404–2414. PMLR, 2019.
- S. Guo, Q. Jiang, L. Chen, and D. Guo. Gene regulatory network inference using pls-based methods. *BMC bioinformatics*, 17:1–10, 2016.
- K. Gurney. *An introduction to neural networks*. CRC press, 2018.
- L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray. From molecular to modular cell biology. *Nature*, 402(Suppl 6761):C47–C52, 1999.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- M. Helmstaedter, K. L. Briggman, S. C. Turaga, V. Jain, H. S. Seung, and W. Denk. Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, 500(7461):168–174, 2013.
- J. Hermans, V. Begy, and G. Louppe. Likelihood-free mcmc with amortized approximate ratio estimators. In *International conference on machine learning*, pages 4239–4248. PMLR, 2020.

- J. Hermans, A. Delaunoy, F. Rozet, A. Wehenkel, V. Begy, and G. Louppe. A trust crisis in simulation-based inference? your posterior approximations can be unfaithful. *arXiv preprint arXiv:2110.06581*, 2021.
- G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.
- S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- M. D. Hoffman, A. Gelman, et al. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.
- K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
- D. Huang, A. Bharti, A. Souza, L. Acerbi, and S. Kaski. Learning robust statistics for simulation-based inference under model misspecification. *Advances in Neural Information Processing Systems*, 36:7289–7310, 2023.
- P. Hunter. Biology is the new physics. *EMBO Reports*, 11(5):350–352, 2010. doi: 10.1038/embor.2010.55.
- G. Karlebach and R. Shamir. Modelling and analysis of gene regulatory networks. *Nature reviews Molecular cell biology*, 9(10):770–780, 2008.
- S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein. Random boolean network models and the yeast transcriptional network. *Proceedings of the National Academy of Sciences*, 100(25):14796–14799, 2003.
- D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- H. Kitano. Computational systems biology. *Nature*, 420(6912):206–210, 2002.
- E. Klipp, R. Herwig, A. Kowald, C. Wierling, and H. Lehrach. *Systems biology in practice: concepts, implementation and application*. John Wiley & Sons, 2005.
- M. Köppen. The curse of dimensionality. In *5th online world conference on soft computing in industrial applications (WSC5)*, volume 1, pages 4–8, 2000.
- A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.
- J. Kruschke. Doing bayesian data analysis: A tutorial with r, jags, and stan. 2014.
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

- M. K. Leung, H. Y. Xiong, L. J. Lee, and B. J. Frey. Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129, 2014.
- J. Li, D. J. Nott, Y. Fan, and S. A. Sisson. Extending approximate bayesian computation methods to high dimensions via a gaussian copula model. *Computational Statistics & Data Analysis*, 106:77–89, 2017.
- W. A. Link and M. J. Eaton. On thinning of chains in mcmc. *Methods in ecology and evolution*, 3(1):112–115, 2012.
- J. Lintusaari, M. U. Gutmann, R. Dutta, S. Kaski, and J. Corander. Fundamentals and recent developments in approximate bayesian computation. *Systematic biology*, 66(1):e66–e82, 2017.
- L. López-Maury, S. Marguerat, and J. Bähler. Tuning gene expression to changing environments: from rapid responses to evolutionary adaptation. *Nature Reviews Genetics*, 9(8):583–593, 2008.
- J.-M. Lueckmann, J. Boelts, D. Greenberg, P. Goncalves, and J. Macke. Benchmarking simulation-based inference. In *International conference on artificial intelligence and statistics*, pages 343–351. PMLR, 2021.
- S. Mangan and U. Alon. Structure and function of the feed-forward loop network motif. *Proceedings of the National Academy of Sciences*, 100(21):11980–11985, 2003.
- L. R. Medsker, L. Jain, et al. Recurrent neural networks. *Design and Applications*, 5(64-67):2, 2001.
- J. D. Murray. *Mathematical biology: I. An introduction*, volume 17. Springer Science & Business Media, 2007.
- N. M. Nasrabadi. Pattern recognition and machine learning christopher m. bishop, editor. *Journal of electronic imaging*, 16(4):049901, 2007.
- R. M. Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.
- G. Papamakarios and I. Murray. Fast  $\varepsilon$ -free inference of simulation models with bayesian conditional density estimation. *Advances in neural information processing systems*, 29, 2016.
- G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.
- G. Papamakarios, D. Sterratt, and I. Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd international conference on artificial intelligence and statistics*, pages 837–848. PMLR, 2019.
- L. Pinheiro Cinelli, M. Araújo Marins, E. A. Barros da Silva, and S. Lima Netto. Variational autoencoder. In *Variational methods for machine learning with applications to deep networks*, pages 111–149. Springer, 2021.
- P. Probst, A.-L. Boulesteix, and B. Bischl. Tunability: Importance of hyperparameters of machine learning algorithms. *Journal of Machine Learning Research*, 20(53):1–32, 2019.

- Ž. Pušnik, M. Mraz, N. Zimic, and M. Moškon. Computational analysis of viable parameter regions in models of synthetic biological systems. *Journal of biological engineering*, 13:1–21, 2019.
- PyMC Development Team. Pymc: Bayesian statistical modeling in python. <https://github.com/pymc-devs/pymc>, 2024. Accessed: 2024-03-16.
- sbi Development Team. Simulation-based inference (sbi): flexible likelihood-free inference in python. <https://github.com/sbi-dev/sbi>, 2024. Accessed: 2024-03-16.
- S. A. Sisson, Y. Fan, and M. Beaumont. *Handbook of approximate Bayesian computation*. CRC press, 2018.
- M. students. Group project, 2024.
- R. Thomas. Boolean formalization of genetic control circuits. *Journal of theoretical biology*, 42(3):563–585, 1973.
- T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. Stumpf. Approximate bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31):187–202, 2009.
- B. L. Trippe and R. E. Turner. Conditional density estimation with bayesian normalising flows. *arXiv preprint arXiv:1802.04908*, 2018.
- B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*, 17(205):1–37, 2016.
- L. Wasserman. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013.
- L. Yang and A. Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.



# Chapter 6

## Appendix

### 6.0.1 Autocorrelation plots

## 6.1 Supplementary results: Repressilator model (Three-Dimensional)

Estimates using MADE

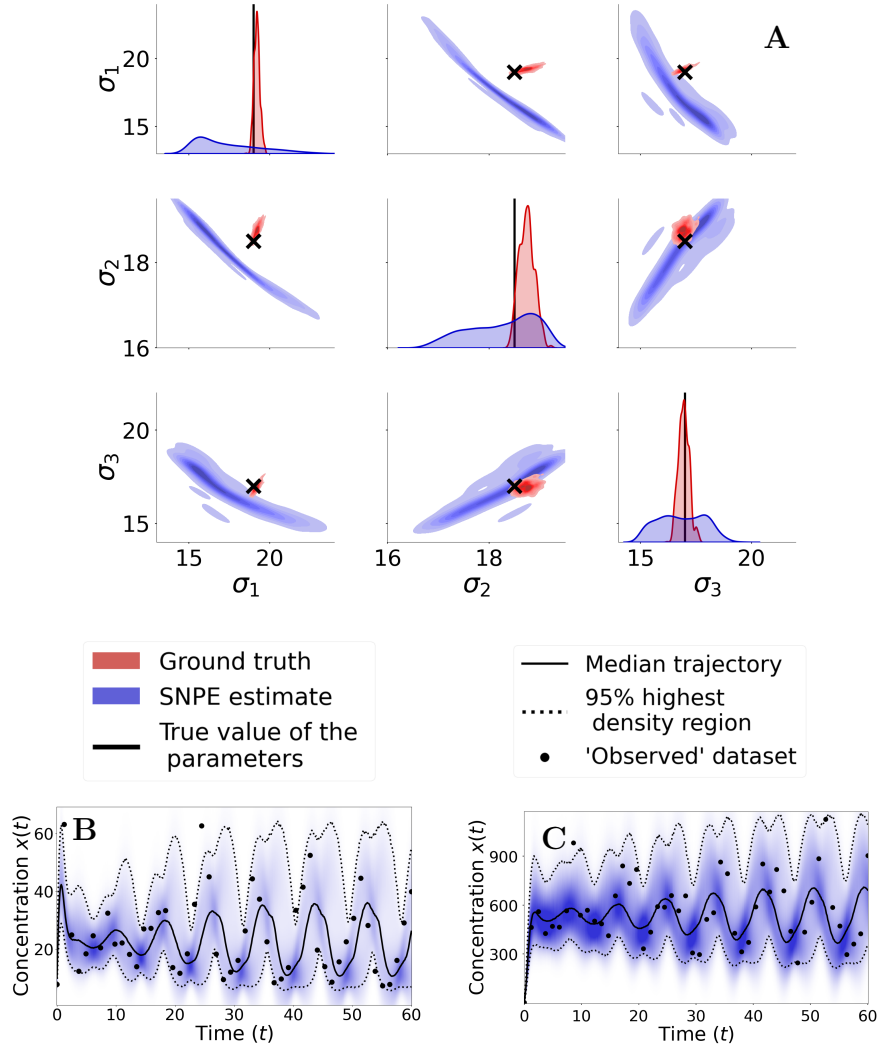


Figure 6.1: Visual comparison between the SNPE estimate using MDN and the ground truth for the ‘observed’ dataset. **A** shows pair plots of the SNPE estimate compared with the ground truth for the three protein translation rates ( $\sigma_1$ ,  $\sigma_2$ ,  $\sigma_3$ ). **B** and **C** show predicted trajectories for mRNA 1 and protein 1, respectively, from PPC on the SNPE estimate.

## 6.2 Supplementary results: Repressilator model (Six-dimensional)

SNPE estimate using MADE

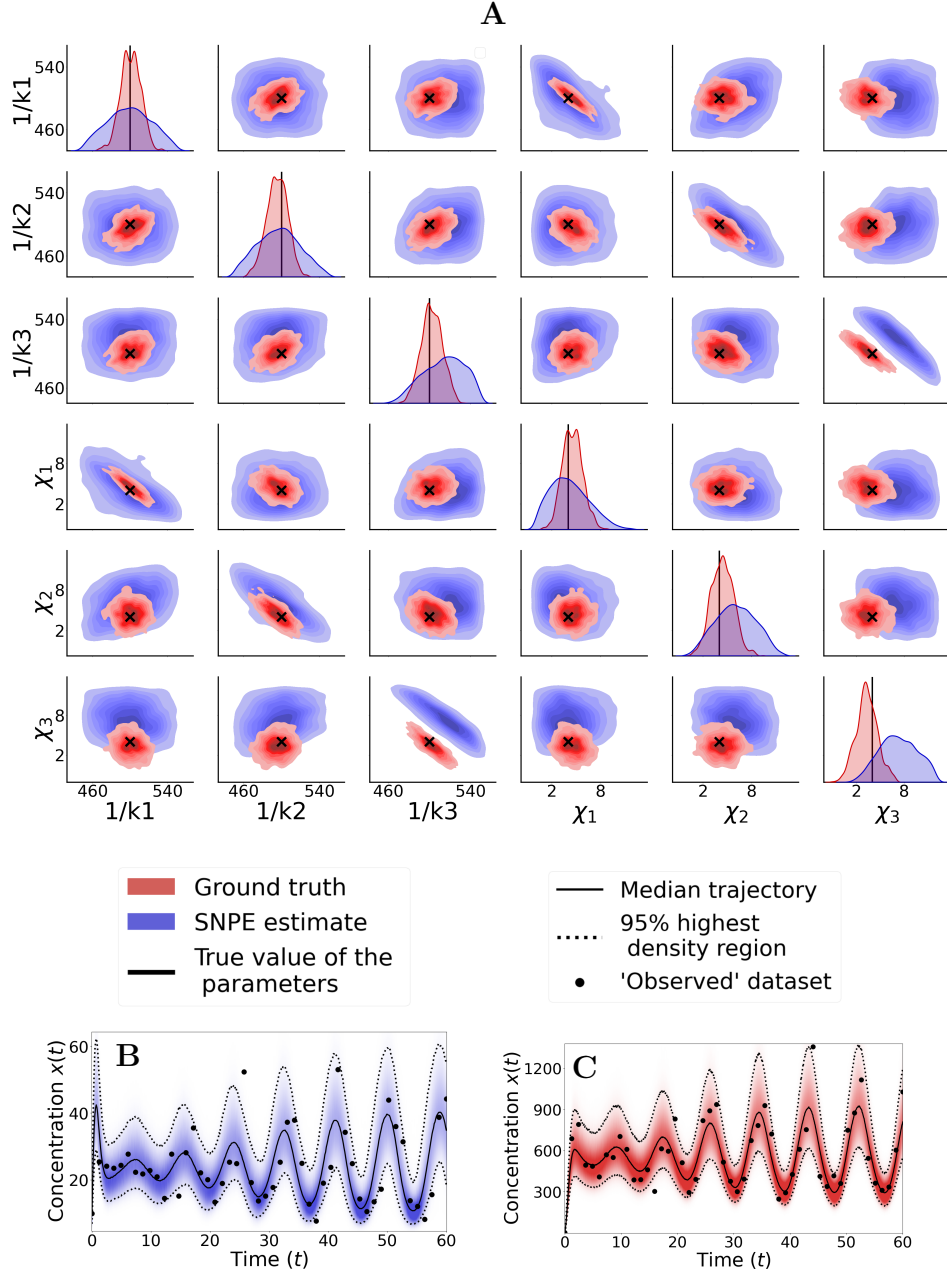


Figure 6.2: Visual comparison between the SNPE estimate using MDN and the ground truth for the 'observed' dataset. **A** shows pair plots of the SNPE estimate compared with the ground truth for the inverse binding affinities and leaky expression rates ( $\chi_1$ ,  $\chi_2$ ,  $\chi_3$ ,  $1/k_1$ ,  $1/k_2$ ,  $1/k_3$ ). **B** and **C** show predicted trajectories for mRNA 1 and protein 1, respectively, from PPC on the SNPE estimate.

## 6.3 Supplementary results: Repressilator model (Twelve-Dimensional)

SNPE estimate from MDN

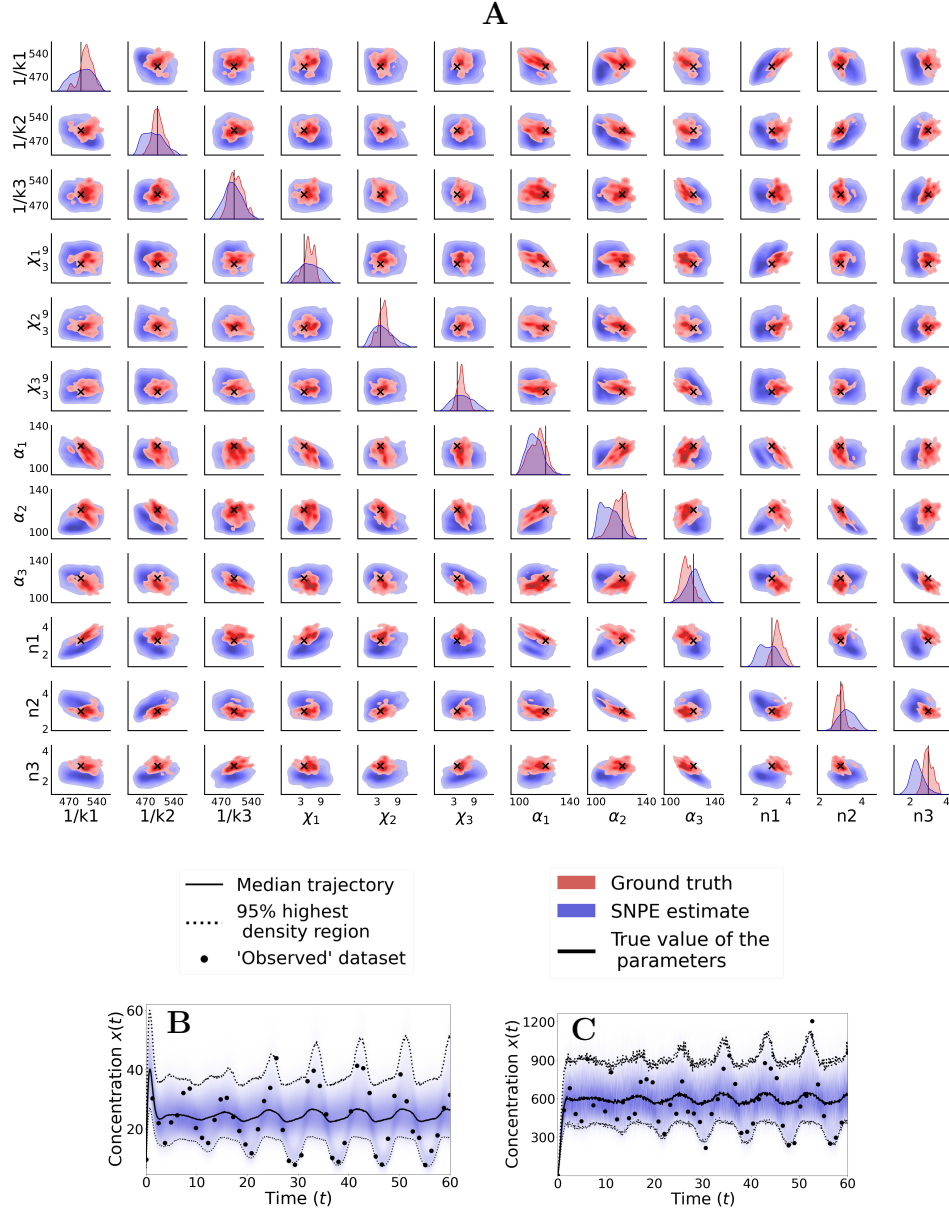


Figure 6.3: Visual comparison between the SNPE estimate 2 using MDN and the ground truth for the ‘observed’ dataset. **A** shows pair plots of the SNPE estimate compared with the ground truth as before.

## 6.4 Supplementary results: Repressilator model (15-Dimensional)

MADE: Estimate 1

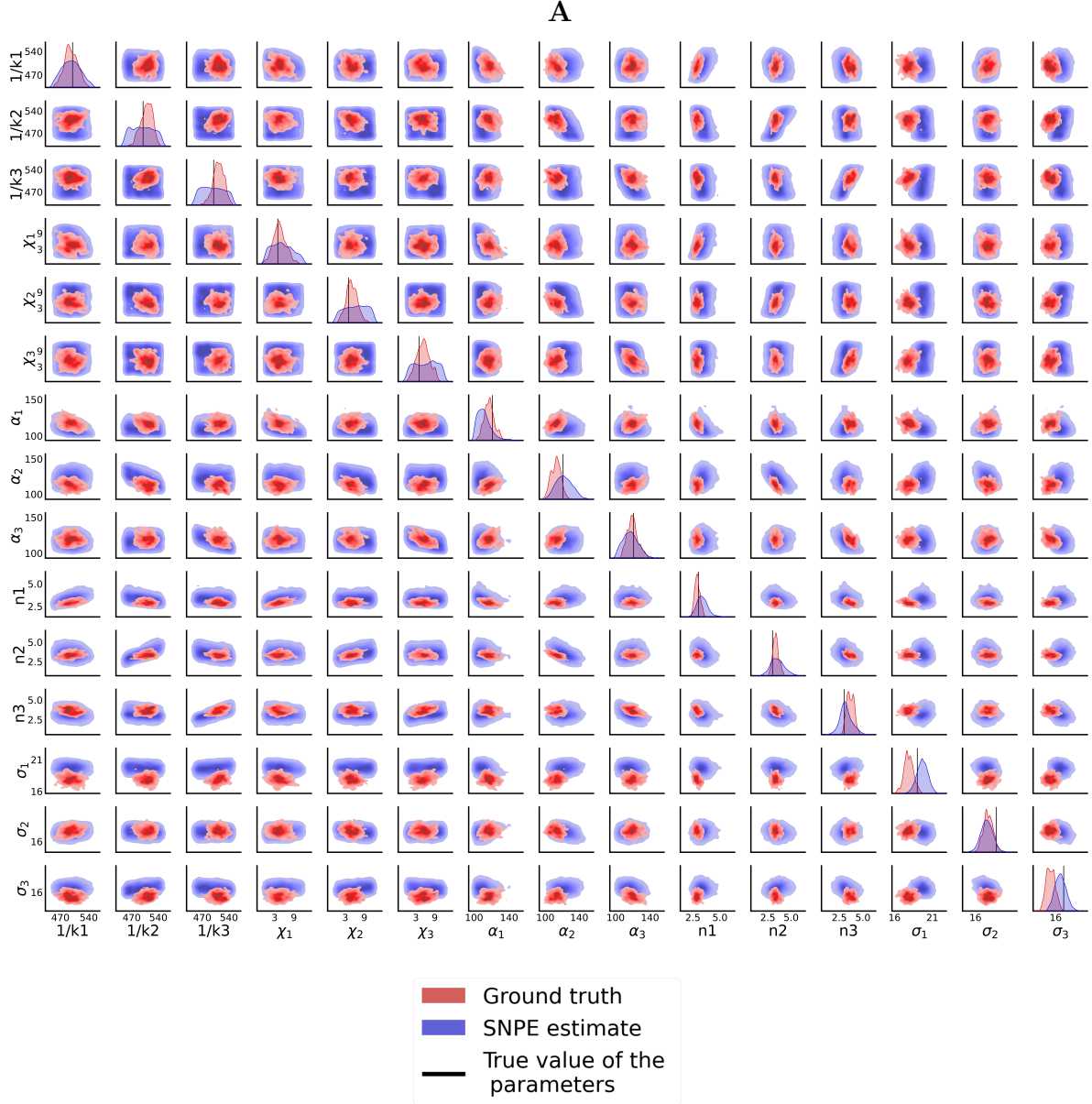


Figure 6.4: Visual comparison between the SNPE estimate using MDN and the ground truth for the ‘observed’ dataset. **A** shows pair plots of the SNPE estimate compared with the ground truth for the inverse binding affinities, leaky expression rates, maximum increments over leaky expression, Hill’s coefficients and protein translation rates ( $\chi_1$ ,  $\chi_2$ ,  $\chi_3$ ,  $1/k_1$ ,  $1/k_2$ ,  $1/k_3$ ,  $\alpha_1$ ,  $\alpha_2$ ,  $\alpha_3$ ,  $n_1$ ,  $n_2$ ,  $n_3$ ,  $\sigma_1$ ,  $\sigma_2$ ,  $\sigma_3$ ).

MADE: Estimate 2

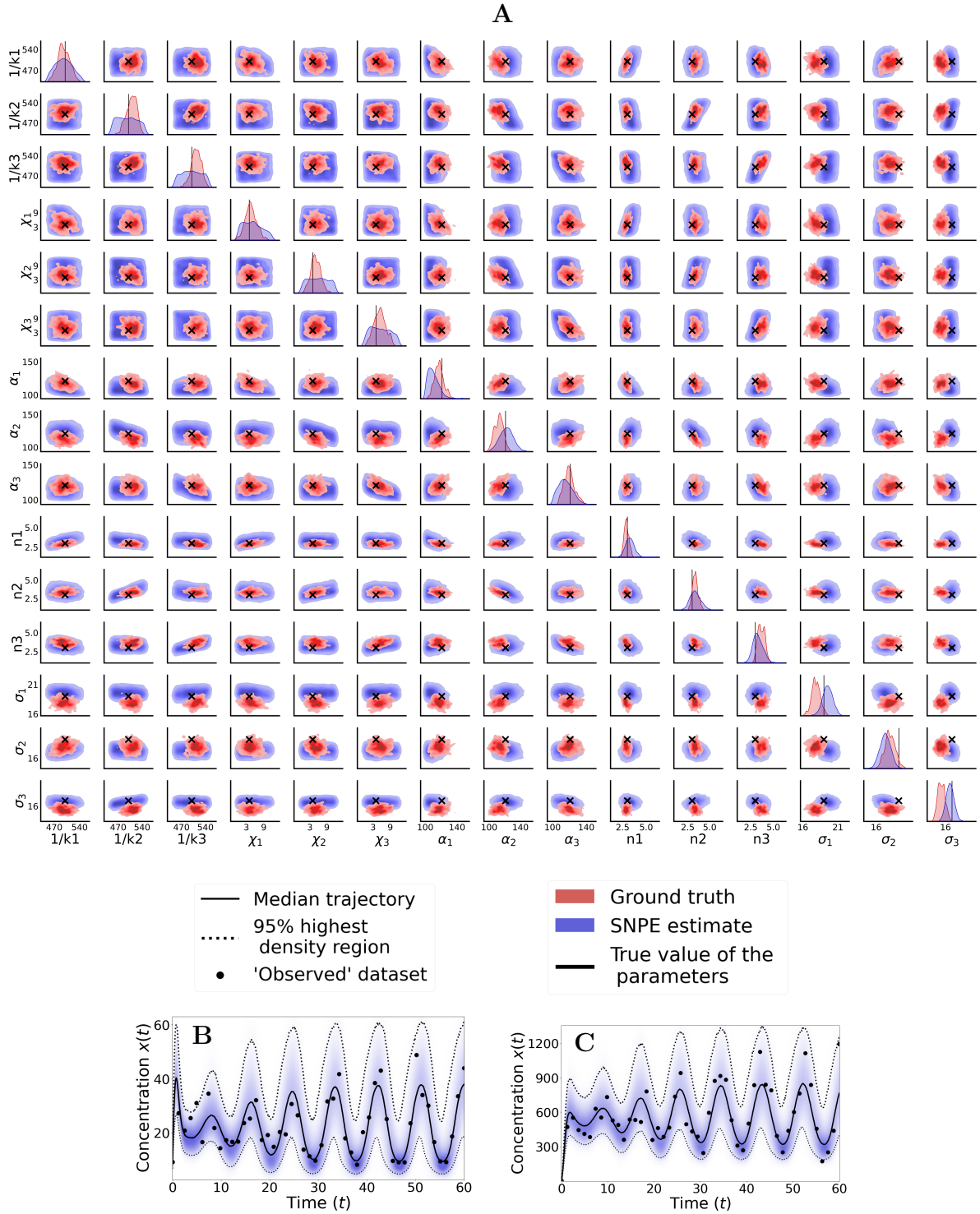


Figure 6.5: Visual comparison between the SNPE estimate 2 using MDN and the ground truth for the ‘observed’ dataset. **A** shows pair plots of the SNPE estimate compared with the ground truth as before.