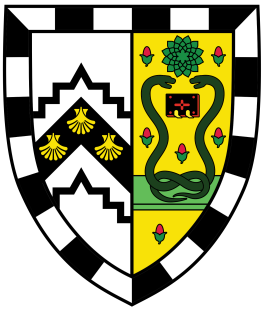
Modelling Transcriptional Regulation using Gaussian Processes

MPhil in Data Intensive Science

University of Cambridge



William Purvis

Gonville & Caius College

Submitted: June 30, 2024

Word Count: 6072

Supervised by Dr Henry Moss

Department of Applied Mathematics and Theoretical Physics

A dissertation submitted to the University of Cambridge in partial fulfilment for the degree of Master of Philosophy in Data Intensive Science.

Declaration

I, William Purvis of Gonville & Caius College, being a candidate for the MPhil in Data Intensive Science, hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains no more than 7000 words including the abstract, tables, figures, appendices*, and references. This dissertation contains 28 figures and 2 tables.

William Purvis
June 30, 2024

*Not including the Generative AI dislaimer.

Acknowledgements

This project has benefited significantly from the contributions, feedback, and support of many people whom I have greatly enjoyed working with, and whose support I have considerable appreciation for during my time at the University of Cambridge. Their contributions have been instrumental in shaping both my research and personal growth. I would like to highlight and acknowledge the following individuals for their support and guidance throughout this project:

Dr Henry Moss, for his mentorship, guidance, and support throughout this project. His expertise and insights have been invaluable in shaping the direction of this research and helping to reproduce the results presented in the original paper using GPJax. In addition to guiding me throughout my project, Henry has been extraordinarily supportive and encouraging of my post-Cambridge career aspirations.

Jacob Moss, for his specific help in obtaining the required data for this project and PyTorch implementations of similar models. His expertise in applying probabilistic machine learning to biological data has been invaluable in shaping the direction of this research.

The Data Intensive Science cohort, for their encouragement and support throughout the year. Their diverse backgrounds and expertise have been instrumental in shaping my thoughts and ideas throughout the project and regarding future career aspirations.

Abstract

This work addresses the challenge of inferring the latent activity profiles of transcription factors in gene regulatory networks, which is crucial for understanding the dynamics of gene expression. **(Barenco et al., 2006)** Using a Latent Force Model (LFM) implemented in GPJax - a GP library in JAX - we reproduced the results obtained by **(N. Lawrence et al., 2006)** concerning the p53 transcription factor network. This involved modelling the latent activity of p53 from gene expression levels of its target genes, treating the transcription factor activity as a GP prior. The model implemented in this work not only successfully replicated the original study but was also able to predict the gene expression levels of p53 targets accurately.

The LFM proposed by **(N. Lawrence et al., 2006)** and reproduced in this work was evaluated through a series of experiments on the same dataset used in the original study. Ablation studies were conducted to test the model's performance on subsets of the data, demonstrating that the model could accurately predict the latent activity of p53 even when trained on fewer target genes, challenging the necessity for large datasets under certain conditions. However, varying results across different gene replicas underscored the importance of open science practices, as consistent data sharing is crucial for replicating scientific findings.

Contents

List of Figures

List of Tables

List of Acronyms

cDNA complementary DNA

DNA deoxyribonucleic acid

GP Gaussian process

HVDM hidden variable dynamic modelling

LFM Latent Force Model

ML Machine learning

MCMC Markov Chain Monte Carlo

MLL marginal log-likelihood

MMGMOS modified gamma model for oligonucleotide signal

mRNA messenger Ribonucleic acid

RBF radial basis function

RNA ribonucleic acid

SIM single input motif

Introduction

Machine learning (ML) can be broadly categorised into two classes: data-driven and mechanistic modelling. Models such as neural networks and decision trees are examples of data-driven models, which learn patterns directly from the data without assuming any prior knowledge of the underlying phenomena. In contrast, mechanistic models involve imposing known mechanisms or theories into the system. These are often derived from physical laws or established relationships within the data, such as differential equations in biological systems or climate models.

For data-driven models, if the data is scarce relative to the complexity of the system, they may struggle to make accurate predictions on test data. Additionally, if the test data falls outside the training data's regime, forcing the model to extrapolate, it can lead to poor model performance. In contrast, mechanistic models retain a major advantage over data-driven models in that they enable accurate predictions even in regions where there may be little to no training data. However, a major challenge of purely mechanistic models is that accurately describing a complex system using such a model may not be possible: even if the underlying mechanisms are known, the system may be too complex to be accurately described by a mechanistic model. **(N. D. Lawrence, 2015)**

This project adopts a hybrid approach introduced by **(N. Lawrence et al., 2006)**, combining a (simplified) mechanistic model of the system with machine learning techniques to augment the system for analysis, known as a *Latent Force Model (LFM)*. The system of study is a subnetwork of genes where the transcription of the genes is driven by a single transcription factor (p53). **(Barenco et al., 2006)** proposed a first order differential equation (outlined in section 2.2.2) to model this system and used a parametric model to estimate both the parameters of the p53 targets and the hidden (*latent*) activity profile of p53 using Markov Chain Monte Carlo (MCMC) methods for Bayesian inference.

The aim of this project is to reproduce the results obtained by **(N. Lawrence et al., 2006)**, where a Gaussian process (GP) model was used for Bayesian inference in the same system. The mechanistic component of this latent force model is a cascade of biophysical differential equations modelling interactions between regulatory proteins and target messenger Ribonucleic acid (mRNA)s and the data-driven aspect comes from treating the transcription factor protein activities as latent variables with GP priors. The results were reproduced using GPJax, a GP library in JAX, **(Pinder and Dodd, 2022)** and the model was evaluated on the same dataset used by **(N. Lawrence et al., 2006)**.

The thesis begins by situating the project within the fields of gene regulation and molecular biology in Chapter 2. This chapter also introduces the mathematical frameworks foundational to the project, including Gaussian Processes and Latent Force Models. Chapter 3 details the methodology used to replicate the results of **(N. Lawrence et al., 2006)**, covering data preprocessing steps and model implementation. The results and discussion are presented in Chapter 4, which is followed by an extension of the work in Chapter 5. The thesis concludes in Chapter 6 with a summary of the key findings and a discussion of the original work that this project sought to reproduce.

Background & Literature Review

2.1 Biological Context

As this project focuses on gene expression, this section will briefly cover the relevant background in molecular biology and gene regulation. The central dogma of molecular biology describes the sequential flow of information through biological systems, often simplified as deoxyribonucleic acid (DNA) → ribonucleic acid (RNA) → protein. Figure 2.1 illustrates this process for a eukaryotic cell. The process of transcription is the first step in gene expression, where a gene is copied into an mRNA molecule by RNA polymerase. This mRNA molecule is then converted into a protein via translation. Gene expression regulation is a complex process involving multiple levels of control, including transcriptional, post-transcriptional, and translational control. Regulator proteins themselves can also be regulated, making gene regulation a complex network of interactions. **(Gibney and Nolan, 2010)**

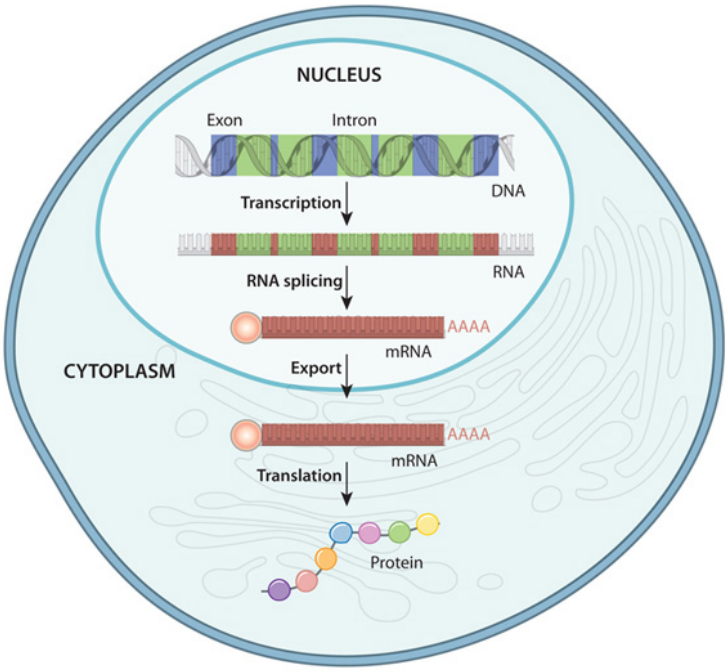


Figure 2.1: An overview of gene expression in a eukaryotic cell. **(Alberts et al., 2013)**

To understand how gene networks function, it is essential to identify their components and understand how they interact with one another. Experimental techniques such as DNA microarray assays or RNA sequencing provide insights into the dynamic relationships in transcriptional responses. This section will specifically focus on DNA microarray assays, as the experimental data for this project comes from time-series analyses using this technique.

DNA microarray assays consist of a grid of wells, each holding a single-stranded DNA probe attached to a solid support, representing specific genes. To perform the assay, mRNA from the cell of interest is isolated and reverse transcription is performed to obtain complementary DNA. These complementary DNA (cDNA)s are labelled with fluorescent tags and introduced to the array, allowing them to hybridise with any matchig DNA sequences they encounter on

the array. By combining fluorescence with a high-resolution digital camera, the presence of each gene can be determined. The relative abundance can also be established by looking at the intensity of the fluorescence. **(Bumgarner, 2013)**

The gene network studied in this project is the p53 network. This network is mediated by the p53 transcription factor, which plays a crucial role in triggering apoptosis (cell death) and mediating tumour suppression in response to cellular stress. The dataset used in this project was generated by **(Barenco et al., 2006)** through the irradiation of a human leukaemia cell line (MOLT4) and collecting protein and RNA at regular intervals after irradiation. Ionising radiation induces DNA damage, which in turn activates the transcription of several known p53 target genes. **(Fei and El-Deiry, 2003)**

2.2 Mathematical Frameworks

2.2.1 Gaussian Processes

A Gaussian process (GP) is formally defined as a collection of random variables, any finite number of which have a joint Gaussian distribution. **(Rasmussen and Williams, 2006)** GPs model distributions over functions of the form $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is any domain. A random function drawn from a GP, characterised by mean and covariance functions μ and k, respectively, is denoted as:

$$f \sim \mathcal{GP}(\mu, k) \tag{2.1}$$

Functions sampled from a GP are primarily defined by the covariance function, or kernel, with the mean function typically set to 0 for simplicity. The choice of kernel dictates the distribution over functions; for example, using the radial basis function (RBF) kernel* imposes specific smoothness and lengthscale properties on these functions. When evaluating a GP at a set of input points $\mathbf{X}^\star$, the resulting function values $f^\star$ are jointly Gaussian distributed with a covariance matrix given by the kernel function evaluated at the input points:

$$f^\star \sim \mathcal{N}(0, k(\mathbf{X}^\star, \mathbf{X}^\star)) \tag{2.2}$$

This formulation exploits the property of multivariate Gaussian distributions that conditional distributions of Gaussians are also Gaussian: a property which extends to Gaussian processes.

Gaussian Process Regression

GPs can be used in the context of regression by incorporating the knowledge that the training data provides about the function: given a set of n points $\mathbf{X} = [x_1, \ldots, x_n]$ and the noisy function values $\boldsymbol{y} = [f(x_1) + \epsilon_1, \ldots, f(x_n) + \epsilon_n]$ (where $\epsilon_n \sim \mathcal{N}(0, \sigma_n^2)$), the joint distribution of the function values at the training points and the noise-free function values at the test points

*The RBF kernel is defined in Equation (2.12)

$\boldsymbol{f}^\star$ is given by:

$$\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{f}^\star \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_X \\ \boldsymbol{\mu}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K}_\sigma & \mathbf{K}_{X,*} \\ \mathbf{K}_{X,*}^\top & \mathbf{K}_{*,*} \end{pmatrix} \right) \tag{2.3}$$

where $\mathbf{K}_\sigma = \mathbf{K}_{X,X} + \sigma_y \mathbf{I}_N$ is $N \times N$, $\mathbf{K}_{X,*} = k(\mathbf{X}, \mathbf{X}^\star)$ is $N \times N^\star$, and $\mathbf{K}_{*,*} = k(\mathbf{X}^\star, \mathbf{X}^\star)$ is $N^\star \times N^\star$.

Therefore, the mean and variance of the predictive distribution at the test points $\boldsymbol{f}^\star$ are given by:

$$\boldsymbol{\mu}_{*|X} = \boldsymbol{\mu}_* + \mathbf{K}_{X,*}^\top \mathbf{K}_\sigma^{-1} (\boldsymbol{y} - \boldsymbol{\mu}_X) \tag{2.4a}$$

$$\boldsymbol{\Sigma}_{*|X} = \mathbf{K}_{*,*} - \mathbf{K}_{X,*}^\top \mathbf{K}_\sigma^{-1} \mathbf{K}_{X,*} \tag{2.4b}$$

To conclude this overview of GPs and GP regression, it is useful to introduce the *marginal likelihood*, which is defined as the integral of the likelihood times the prior over the function values. The marginal log-likelihood conditioned on the hyperparameters $\boldsymbol{\theta}$ is defined in Equation (2.5): **(Rasmussen and Williams, 2006)**

$$\log p(\mathbf{y} \mid X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^\top K_y^{-1}\mathbf{y} - \frac{1}{2}\log |K_y| - \frac{n}{2}\log 2\pi \tag{2.5}$$

The gradient of the marginal log-likelihood with respect to the hyperparameters can be computed and then used in a gradient-based optimization algorithms such as Adam to find the optimal hyperparameters. **(Kingma and Ba, 2015)**

2.2.2 Latent Force Models

In the context of gene regulation, when multiple genes are regulated by a single transcription factor, this is known as a single input motif (SIM). **(Khanin et al., 2005)** To model the p53 SIM, **(Barenco et al., 2006)** introduced a method known as hidden variable dynamic modelling (HVDM), which was applied to the network by assuming a linear response model. Although the linear assumption is not verified in practice, it has the advantage of giving rise to an exactly tractable inference problem and is justified as follows: the rate at which p53-dependent mRNA transcripts accumulate (measured as the rate of gene expression) depends on the base transcription rate of a target gene (B), the sensitivity of that gene to p53 (S), the rate of mRNA decay (D), and the activity level of p53 itself. For a given gene j, these factors are summarised below:

1. Basal transcription rate of target gene, B_j: the fundamental rate at which gene j transcribes mRNA in the absence of p53 modulation.
2. Sensitivity of the gene to p53, S_j: How responsive the gene is to the presence of p53.
3. Level of activity of p53, $f(t)$: The dynamic activity level of p53 at any given time t.

4. Transcript decay rate, D_j: The rate at which gene j's mRNA degrades.

The parameters listed above can be combined into a non-autonomous, first-order linear differential equation to describe the time evolution of each gene transcript. Equation (2.6) relates a given gene j's expression level, denoted as $x_j(t)$, at time t to the concentration of the regulating transcription factor protein $f(t)$.

$$\frac{\mathrm{d}x_j(t)}{\mathrm{d}t} = B_j + S_j f(t) - D_j x_j(t) \tag{2.6}$$

Given the initial condition $x_j(0) = A_j$, Equation (2.6) can be solved to recover (solved in Appendix A.1):

$$x_j(t) = \frac{B_j}{D_j} + \left(A_j - \frac{B_j}{D_j}\right) e^{-D_j t} + S_j e^{-D_j t} \int_0^t e^{D_j u} f(u) \mathrm{d}u \tag{2.7}$$

Examining Equation (2.7), three distinct components are identified:

1. A steady-state component (B_j/D_j), which represents the value towards which $x_j(t)$ converges as $t \to \infty$
2. A transient solution from initial conditions (second term), which shows how the system transitions from its initial state to the steady state. It accounts for the initial deviation from the steady-state value at $t = 0$
3. And a driven response (integral term) which accounts for the driving latent force function $f(t)$, which in this case is the activity level of the p53 transcription factor.

Setting the initial baseline expression to zero, $x_j(0) = 0$, which implies $A_j = B_j/D_j$, allows Equation (2.7) to be re-written as follows:

$$x_j(t) = \frac{B_j}{D_j} + L_j[f](t) \tag{2.8}$$

Where $L_j[f](t)$ is the linear operator (convolution integral) relating the latent function f to the mRNA abundance of gene j, $x_j(t)$:

$$L_j[f](t) = S_j e^{-D_j t} \int_0^t e^{D_j u} f(u) \mathrm{d}u \tag{2.9}$$

As Equation (2.8) involves a linear operator acting on the latent function $f(t)$, modelling the concentration of p53 as a latent function drawn from a GP prior implies that the mRNA abundance levels will also be modelled as a GP. Specifically, if the covariance function associated with $f(t)$ is $k_{ff}(t, t')$, then the covariance function of $x_j(t)$ can be calculated via Equation (2.8) (due to linear operator acting on f and linearity relations of covariance):

$$\mathrm{Cov}(L_j[f](t), L_k[f](t')) = L_j \otimes L_k k_{ff}(t, t') \tag{2.10}$$

Explicitly, this is given by the following formula: (derived in Appendix A.2.1)

$$k_{x_j x_k}(t, t') = S_j S_k e^{-D_j t - D_k t'} \int_0^t e^{D_j u} \int_0^{t'} e^{D_k u'} k_{ff}(u, u') \mathrm{d}u' \mathrm{d}u \tag{2.11}$$

Which defines the covariance function between the expression levels of genes j and k. If the prior over the latent function f is taken to be the RBF kernel:

$$k_{ff}(t, t') = \exp\left(-\frac{(t - t')^2}{2l^2}\right) \tag{2.12}$$

where l controls the width of the basis functions, the integrals in Equation (2.11) can be computed analytically to give the following expression for the covariance function:

$$k_{x_j x_k}(t, t') = S_j S_k \frac{\sqrt{\pi} l}{2} [h_{kj}(t, t') + h_{kj}(t', t)] \tag{2.13}$$

where

$$\begin{aligned} h_{kj}(t, t') = & \frac{\exp(\gamma_k)^2}{D_j + D_k} \Bigg\{ \exp[-D_k(t' - t)] \left[\mathrm{erf}\left(\frac{t' - t}{l_r} - \gamma_k\right) + \mathrm{erf}\left(\frac{t}{l_r} + \gamma_k\right) \right] \\ & - \exp\left[-(D_k t' + D_j)\right] \left[\mathrm{erf}\left(\frac{t'}{l_r} - \gamma_k\right) + \mathrm{erf}\left(\gamma_k\right) \right] \Bigg\} \end{aligned}$$

Here, $\mathrm{erf}(x)$ is the real-valued error function, $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-y^2) \mathrm{d}y$, and $\gamma_k = \frac{D_k l}{\sqrt{2}}$.

To infer the protein concentration levels, the 'cross covariance' terms between $x_j(t)$ and $f(t')$ are required. This can be obtained explicitly for squared exponential priors on the latent function f as: (solved in Appendix A.2.2)

$$k_{x_j f}(t, t') = \frac{S_{rj} \sqrt{\pi} l}{2} \exp(\gamma_j)^2 \exp(-D_j(t' - t)) \left[\mathrm{erf}\left(\frac{t' - t}{l} - \gamma_k\right) + \mathrm{erf}\left(\frac{t}{l} + \gamma_k\right) \right] \tag{2.14}$$

In summary, three different kernels have been defined:

1. The covariance function of gene expressions at times t' and t: $k_{x_j x_k}(t, t')$
2. The covariance function for the latent force function $k_{ff}(t, t')$ (taken as an RBF prior)
3. The cross-covariance between measured gene expression at time t and the latent force function at time t': $k_{x_j f}(t, t')$

The latent p53 activity profile can then be modelled as a GP with a zero mean function and covariance function $k_{ff}(t, t')$:

$$f \sim \mathcal{GP}(0, k_{ff}(t, t')) \tag{2.15}$$

This GP can then be related to a GP describing the gene expressions $x_j(t)$ via Equation (2.8) to give:

$$x_j(t) \sim \mathcal{GP}(m_{x_j}(t), k_{x_j x_j}(t, t')) \tag{2.16}$$

Where $m_{x_j}(t) = \frac{B_j}{D_j}$ and $k_{x_j x_j}(t, t')$ is given by Equation (2.11).

The joint distribution of the latent p53 activity profile and measured gene expressions can then be obtained by combining the three kernels defined above:

$$\begin{pmatrix} \mathbf{x}(t_1) \\ \vdots \\ \mathbf{x}(t_N) \\ f^* \end{pmatrix} \sim \mathcal{N}_{N+1} \left(\begin{pmatrix} \boldsymbol{\mu}_{\mathbf{x}} \\ 0 \end{pmatrix}, \left(\begin{array}{ccc:c} k_{xx}(t_1,t_1) & \dots & k_{xx}(t_1,t_N) & k_{xf^*}(t_1) \\ \vdots & \ddots & \vdots & \vdots \\ k_{xx}(t_N,t_1) & \dots & k_{xx}(t_N,t_N) & k_{xf^*}(t_N) \\ \hdashline k_{xf^*}(t_1)^T & \dots & k_{xf^*}(t_N)^T & k_{f^*f^*}(t_N,t_N) \end{array} \right) \right) \tag{2.17}$$

where $\mathbf{x}(t_i)$ represents the gene expressions measured at time t_i, f^* represents the latent function (p53 activity profile), and $\boldsymbol{\mu}_{\mathbf{x}}$ the mean function evaluations of gene expressions at times $t_1, \dots, t_N$.

Assuming that the gene expression measurements are noisy, i.e. $y_i = x_i + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$, the GP regression techniques outlined in Section 2.2.1 can then be applied to infer the mean and covariance function of the posterior process on f (latent activity profile of p53 transcription factor).

Reproduction

3.1 Data Preprocessing

The dataset used in this study consists of expression level measurements for 50 genes across three replicas. These measurements, obtained experimentally by **(Barenco et al., 2006)**, were taken at 2-hour intervals for 12 hours after irradiation using an Affymetrix U133A oligonucleotide microarray.* In addition to the gene expressions, the authors used the modified gamma model for oligonucleotide signal (MMGMOS) algorithm to obtain uncertainties for each gene expression measurement.

The original paper then restricted their interest to five known targets of p53:

Target	Probeset	Aliases
DDB2	`203409_at`	
p21	`202284_s_at`	p21CIP1, CDKN1A
SESN1/hPA26	`218346_s_at`	
BIK	`205780_at`	
TNFRSF10b	`209294_x_at`, `209295_at`, `210405_x_at`	

Table 3.1: Target genes of interest in the p53 dataset.

The hidden variable dynamic modelling (HVDM) model employed by **(Barenco et al., 2006)** was made identifiable by experimentally measuring the mRNA decay of the p21 target gene using polymerase chain reaction. In addition, **(N. Lawrence et al., 2006)** fixed the scale of the sensitivities by choosing p21's sensitivity to equal 1.0, and $f(0)$ was constrained to be zero. To validate their model, **(Barenco et al., 2006)** made explicit protein concentration measurements by growing mutant cell lines where the p53 gene had been knocked out. These *ground truth* values are used to validate the model implemented in this work.

The `load_barenco_data()` function pre-processed the data by extracting the relevant target genes listed in Table 3.1 and log-normalising their corresponding expression levels and uncertainties. This resulted in the following dataset:

$$\mathcal{D} = \{(x_j(t_i), \sigma_j(t_i)) \mid j = 1, 2, 3, 4, 5;\ i = 1, 2, 3, 4, 5, 6, 7\} \tag{3.1}$$

Calculating the covariance between different genes at different timepoints via $k_{x_j x_k}(t, t')$, as well the covariance between different genes and the latent function via $k_{x_j f}(t, t')$, requires the Gaussian processes to learn vector-valued functions. Since GPJax does not natively support vector-valued functions, the final step of data preprocessing involved augmenting the dimensionality of the dataset to include the timepoints and an additional *flag* dimension in addition to the gene expression measurements so that the data points are three-dimensional: $d = (t, j, z)$ where t is the timepoint, j is the gene index, and $z \in \{0, 1\}$. Using this flag, a GP can be used

*Experimental details regarding data acquisition can be found in the Barenco paper.

to learn the function $g(\bullet)$, which is defined as follows:

$$g(t,i,z) = \begin{cases} x(t,i) & \text{if } z = 1, \\ f(t) & \text{otherwise.} \end{cases} \tag{3.2}$$

where $x(\bullet)$ is the gene expression as a function of the gene index and timepoint, and $f(\bullet)$ is the latent function at time t. Therefore, the training data for $k_{x_j x_k}(t,t')$ is $x_j = (t,j,1)$ and $x_k = (t',k,1)$. To make a prediction for the latent force, a data point such as $d^* = (t^*,j^*,1)$ would be used. In summary:

$$X_{test} = (t,i,1) \rightarrow x(t,i) \tag{3.3a}$$
$$X_{test} = (t,i,0) \rightarrow f(t) \tag{3.3b}$$

3.2 GPJax

This project was implemented in python using the GPJax library, a didactic GP library in JAX. **(Pinder and Dodd, 2022)** The fundamental concept of GPJax is that every object is represented as a JAX PyTree, which allows for easy manipulation of the objects using JAX's automatic differentiation capabilities. A PyTree is a data structure that is composed of other data structures, where each node is either a leaf (a simple data structure) or another PyTree. For example, to define a GP prior in GPJax, the user must define a mean function and a kernel function, which are both PyTrees. The prior object is then a PyTree with the mean and kernel functions as it's leaves (as shown in Code Block 3.1).

```
kernel = gpx.kernels.RBF()
meanf = gpx.mean_functions.Zero()
prior = gpx.gps.Prior(mean_function=meanf, kernel=kernel)
```

Code Block 3.1: Defining a GP prior in GPJax (from GPJax documentation)

The posterior is then defined as a PyTree with the prior and likelihood as its leaves. In summary, every model object in GPJax is a PyTree or a transformation on a PyTree, providing full integrability with the broader JAX ecosystem and leveraging JAX's automatic differentiation capabilities.

3.2.1 Comparison with original implementation

The initial application of GPs to model transcriptional regulation, presented by **(N. Lawrence et al., 2006)**, was implemented in MATLAB. Unfortunately, the code was not made publicly available, and the authors did not provide any details on the specific implementation or data pre-processing steps. As a substitute comparison, the GPJax implementation carried out in this work was compared to a GPyTorch implementation of the same model from the ALFI package. **(Moss et al., 2022)** Similarly to GPJax, GPyTorch also uses automatic differentiation to perform inference on GPs, although it is built on PyTorch rather than JAX, and does not use PyTrees. The ALFI library features multiple latent force models and the specific model used on the p53 dataset was extensively refactored to minimise abstraction and included in the project

repository for comparison.

3.2.2 GPJax Implementation

Given the augmented data outlined in Section 3.1 and the kernels defined in Section 2.2.1, the first step of the GPJax implementation involved defining a custom piecewise kernel whose output depends on the z label of the inputs. For two inputs $\mathbf{T} = (t, j, z)$ and $\mathbf{T}' = (t', k, z')$, the combined kernel function $k(\mathbf{T}, \mathbf{T}')$ is defined as follows:

$$k(\mathbf{T}, \mathbf{T}') = \begin{cases} k_{xx}((t,j),(t',k)) & \text{if } z = z' = 1 \\ k_{xf}((t,j),t') & \text{if } z \neq z' \\ k_{ff}(t,t') & \text{if } z = z' = 0 \end{cases} \tag{3.4}$$

Where $k_{xx}((t,j),(t,k))$ and $k_{xf}((t,j),t')$ are modifications of $k_{x_j x_k}(t,t')$ and $k_{x_j f}(t,t')$, respectively, to account for the additional gene index dimension.

To implement this approach in GPJax, a custom kernel was defined with k_{xx}, k_{xf}, and k_{ff} as its methods. The correct kernel to use is then selected based on the z labels of the input data in the `__call__` method shown in Code Block 3.2.

```
    def __call__(self, t: Int[Array, "1 3"], t_prime: Int[Array, "1 3"]) ->
ScalarFloat:

        # Get flag from input
        f1 = jnp.array(t[2], dtype=int)
        f2 = jnp.array(t_prime[2], dtype=int)

        # Use switches to select correct kernel
        kxx_switch = f1 * f2
        kff_switch = (1 - f1) * (1 - f2)
        kxf_switch = f1 * (1 - f2)
        kxf_t_switch = (1 - f1) * f2

        final_kernel = (
            kxx_switch * self.kernel_xx(t, t_prime)
            + kff_switch * self.kernel_ff(t, t_prime)
            + kxf_switch * self.kernel_xf(t, t_prime)
            + kxf_t_switch * self.kernel_xf(t_prime, t)
        )

        return final_kernel
```

Code Block 3.2: Call method for combined kernel

It is worth highlighting that during training, only the hyperparameters of the k_{xx} sub-kernel (B, S, D, and ℓ) and mean function (defined as $m_{x_j}(t) = \frac{B_j}{D_j}$) are optimised for, as the training dataset contains the target gene expressions. These hyperparameters are summarised in Table 3.2 below.

Hyperparameter	Description	Optimised in
B	Basal transcription rate of target genes	mean function
S	Sensitivity of target genes to p53	kernel
D	Decay rate of target genes	kernel and mean function
ℓ	Lengthscale of RBF prior	kernel

Table 3.2: Hyperparameters of the GP model

GPJax's architecture, which treats the mean function and kernel as separate PyTrees, presented a significant challenge when optimising the decay rate D. This hyperparameter, shared by both the mean and covariance functions, could not be optimised directly by GPJax through maximising the marginal likelihood of the training data directly. To overcome this limitation, a custom LFM was implemented in GPJax, inheriting from the `gpx.base.Module` class to enable parameter sharing between the mean function and kernel. This custom model is comparable to the `gpjax.gps.ConjugatePosterior` PyTree from the GPJax library, and therefore contains the following methods:

- `kernel`: Combined kernel defined in code block 3.2 (including sub-kernels k_{xx}, k_{xf}, and k_{ff}).
- `mean_function`: Mean function defined as $m_{x_j}(t) = \frac{B_j}{D_j}$ for gene expressions, and $m_f(t) = 0$ for the latent function.
- `gram`: Computes the Gram matrix of the kernel.
- `cross_covariance`: Computes the $N \times M$ gram matrix on an a pair of input matrices with shape $N \times D$ and $M \times D$
- `latent_predict`: Predicts the latent function
- `gene_predict`: Predicts the gene expressions

To train this custom model and learn Type-2 maximum likelihood estimates of the hyperparameters listed in Table 3.2, a custom trainer was also implemented. The custom `JaxTrainer` leverages JAX's automatic differentiation capabilities to compute the gradients of the marginal likelihood with respect to the hyperparameters and uses a gradient-based optimiser to update the hyperparameters. In addition, the `JaxTrainer` class includes an `after_epoch` method which fixes the scale of the sensitivities by choosing p21's sensitivity to equal 1.0 and p21's decay rate to equal 0.8, as implemented by **(N. Lawrence et al., 2006)**.

The trained model was then used to predict the latent function and gene expressions for the five target genes at a set of test points between 0 and 12 hours. To validate the latent function, predictions for p53's activity profile from **(Barenco et al., 2006)** were used as the ground truth. Similarly, for the gene expressions, the protein concentration levels from the same paper were used as the ground truth. The optimised sensitivity, basal transcription rates, and decay rate hyperparameters were also compared to experimentally measured values from the **(Barenco et al., 2006)**.

Discussion & Results

4.1 Original results

The predicted latent activity profile for the p53 transcription factor obtained by **(N. Lawrence et al., 2006)** is shown in Figure 4.1 below. The Solid line is the posterior mean prediction, dashed lines are 95% credibility intervals, and experimental predictions are shown as crosses.

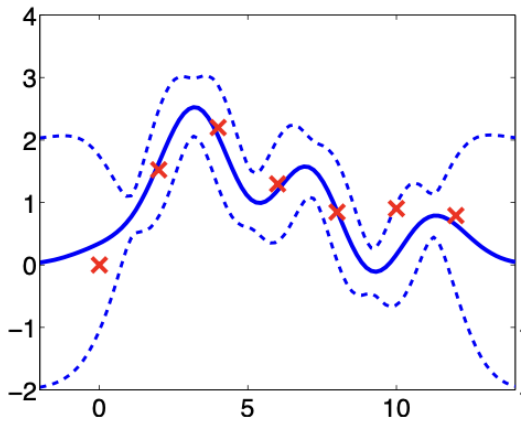


Figure 4.1: Predicted p53 activity profile from **(N. Lawrence et al., 2006)**.

In addition to the predicted latent function, the inference results on the hyperparameters of the p53 target genes are shown in Figure 4.2 below. The grey bars are estimates obtained by **(N. Lawrence et al., 2006)** and the white are estimates obtained by **(Barenco et al., 2006)**.

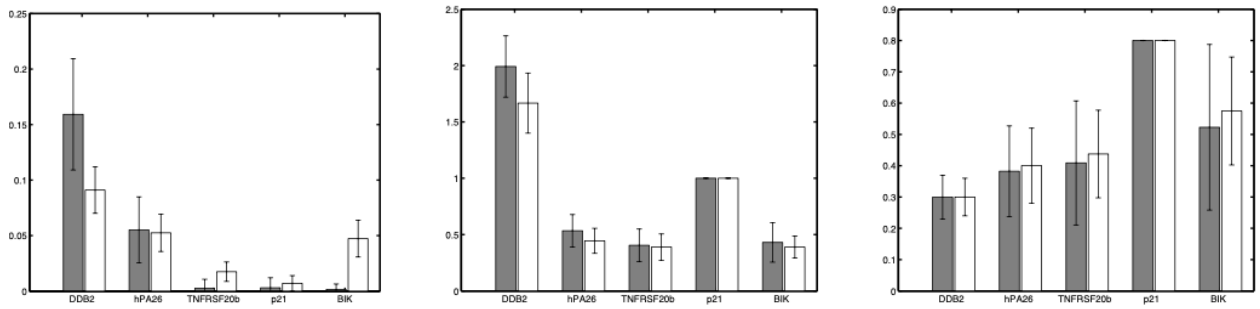


Figure 4.2: Inferred hyperparameters for p53 target genes from **(N. Lawrence et al., 2006)**. The first plot shows the basal transcription rates, the middle plot shows the sensitivities, and the last plot shows the decay rates.

The posterior mean function obtained by **(N. Lawrence et al., 2006)** aligns closely with the experimental predictions, although significant oscillatory behaviour is observed in the posterior variance at the left and right-hand tails. The authors attribute this to an artifact caused by the RBF covariance function, with the steep rise between time zero and time two forcing the lengthscale to be small, leading to the oscillatory behaviour. **(Rasmussen and Williams, 2006)**

Hyperparameters were assigned a gamma prior distribution ($a = b = 0.1$, corresponding to a

mean of 1 and variance of 10) and samples from the posterior distribution were obtained by **(N. Lawrence et al., 2006)** using Hybrid Monte Carlo. The results shown in Figure 4.2 show decent accordance with the experimental estimates, with the decay and sensitivity rate of p21 clearly shown to be fixed at values of 0.8 and 1.0, respectively.

As mentioned in Section 3.2.1, the authors did not provide any details on the specific implementation details so Figures 4.1 and 4.2 are used simply as reference points for the results obtained in this work. In-depth comparisons will be made against results obtained from ALFI, which are presented in the following section. **(Moss et al., 2022)**

4.1.1 ALFI Results

The latent activity profile for the p53 transcription factor obtained from the ALFI package using the first data replica is shown in Figure 4.3 below. The solid line is the posterior mean prediction, dashed lines represent two standard deviations, and experimental predictions are shown as crosses.

The predicted latent activity profile for the p53 transcription factor from the first replica obtained with ALFI is shown in Figure 4.3 below. The Solid line is the mean prediction, dashed lines represent two standard deviations, and experimental predictions from **(Barenco et al., 2006)** are shown as crosses.

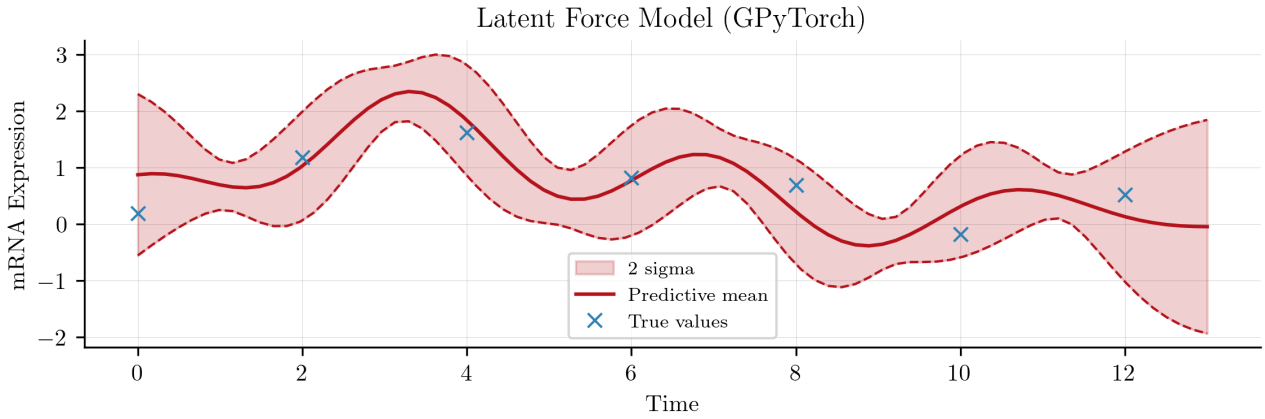


Figure 4.3: Predicted p53 activity profile from ALFI.

ALFI uses GPyTorch to implement the GPs, and the results obtained are similar to those obtained by **(N. Lawrence et al., 2006)**. For numerical stability, a jitter term of 1×10^{-4} was added to the diagonal of the covariance matrix. To optimise the marginal log-likelihood with respect to the hyperparameters the Adam optimiser was used with a learning rate of 0.01 over 150 iterations. **(Kingma and Ba, 2015)**

It is worth highlighting that the experimental predictions differ in Figures 4.1 and 4.3, which is likely due to different pre-processing steps taken to prepare the data. The results of inference on the hyperparameters for the p53 target genes are shown in Figure 4.4 below. The red bars are estimates obtained directly from the posterior distribution, and the blue bars are estimates obtained by **(Barenco et al., 2006)**.

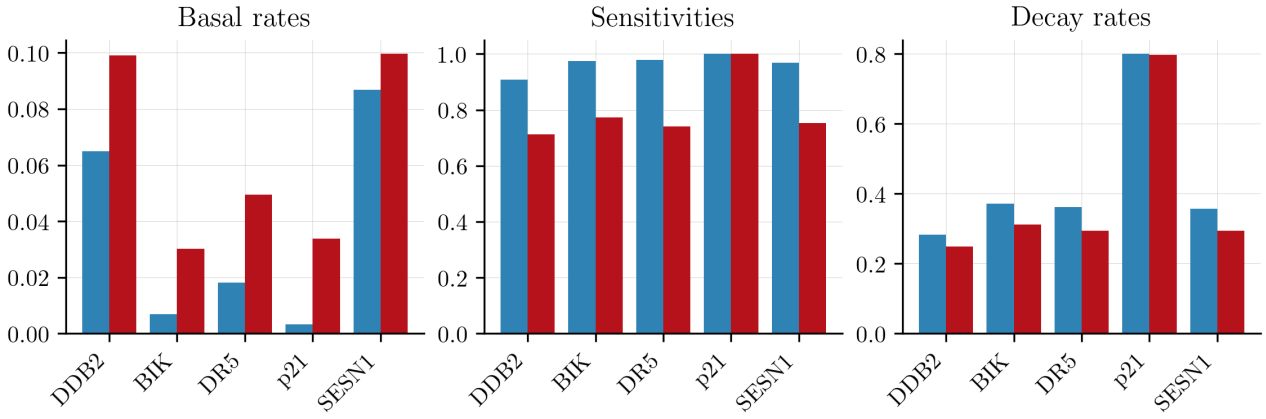


Figure 4.4: Inferred hyperparameters for p53 target genes from ALFI.

The estimates for target gene sensitivities and decay rates are in good agreement with the experimental estimates, with the decay rate and sensitivity of p21 fixed at 0.8 and 1.0, respectively. However, the estimated basal transcription rates are higher than the experimental estimates, a discrepancy likely due to not fixing any of the target gene basal rates. Similarly to the experimentally inferred activity profile of p53, the experimental estimates for the hyperparameters of the target genes in Figure 4.4 differ from those presented in Figure 4.2. This is attributed to the pre-processing steps taken to prepare the data, but could also be due to a different gene replica used by **(N. Lawrence et al., 2006)** and **(Barenco et al., 2006)**.

In addition to the predicted latent function and hyperparameters, ALFI was used to predict the gene expressions for the five target genes at the test points. The predicted gene expressions for the five target genes are shown in Figure 4.5 below. The solid line is the posterior mean prediction, dashed lines represent two standard deviations, and experimental predictions are shown as crosses.

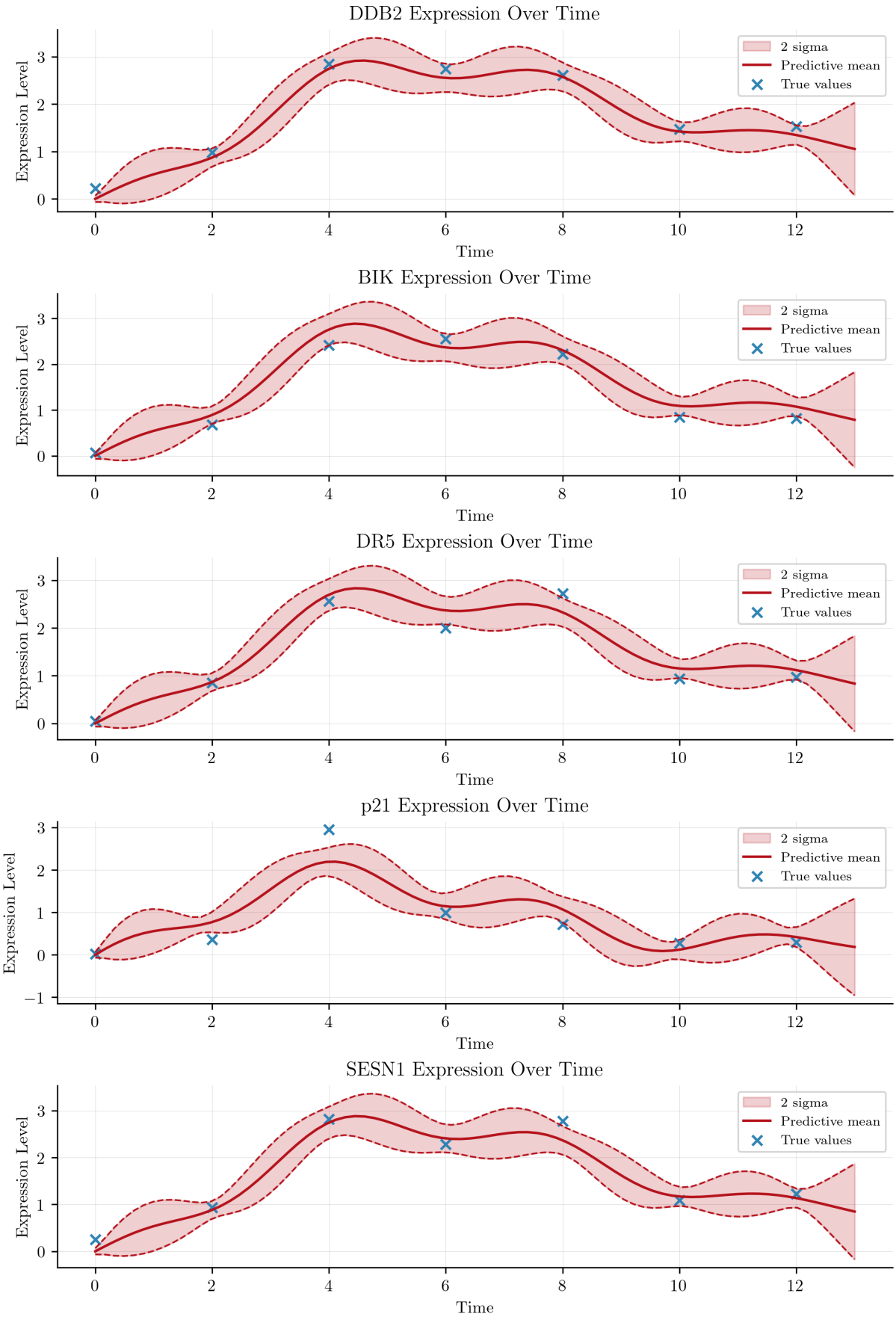


Figure 4.5: Predicted gene expressions for p53 target genes from ALFI.

4.2 GPJax results

The latent activity profile of p53 predicted by the GPJax implementation is shown in Figure 4.6 below. The solid line is the posterior mean prediction, dashed lines represent two standard deviations, and experimental values are shown as crosses.

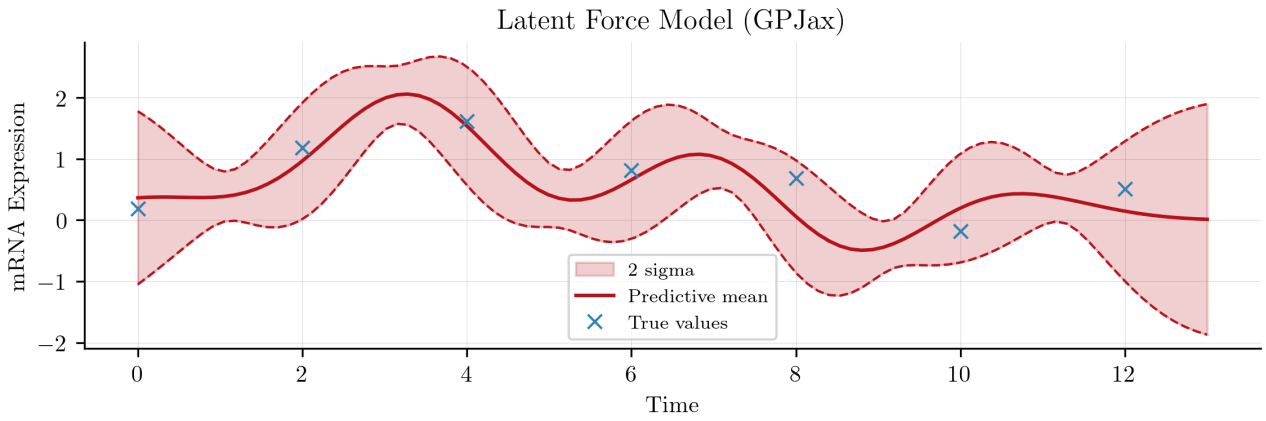


Figure 4.6: Predicted p53 activity profile from GPJax.

The posterior mean function aligns closely with the experimental predictions, especially in the region between time zero and time six. To ensure a fair comparison with the results obtained using the ALFI LFM, the same data replica was used to train the GPJax model, a jitter term of 1×10^{-4} was added to the diagonal of the covariance matrix to ensure numerical stability, and the Adam optimiser was used to optimise the marginal log-likelihood with a learning rate of 0.01 over 150 iterations.

To enforce the $f(0) = 0$ constraint proposed by **(N. Lawrence et al., 2006)**, an artificial data point at $(0, 0)$ was added to the training data. This resulted in an improved fit to the latent prediction at the first timepoint compared with the ALFI implementation. Investigation into the kernel behaviour showed that although this artificial data point resulted in a zero mean for the test and training data ($\boldsymbol{\mu}_\star$ and $\boldsymbol{\mu}_X$ in Equation (2.4a)), the resulting posterior mean did not exactly equal zero due to inaccuracies in matrix inversion.

Hyperparameter estimates were obtained directly from the optimised model, and the results are shown in Figure 4.7 below. The red bars are estimates obtained from the posterior distribution and the blue bars are experimental values obtained by **(Barenco et al., 2006)**.

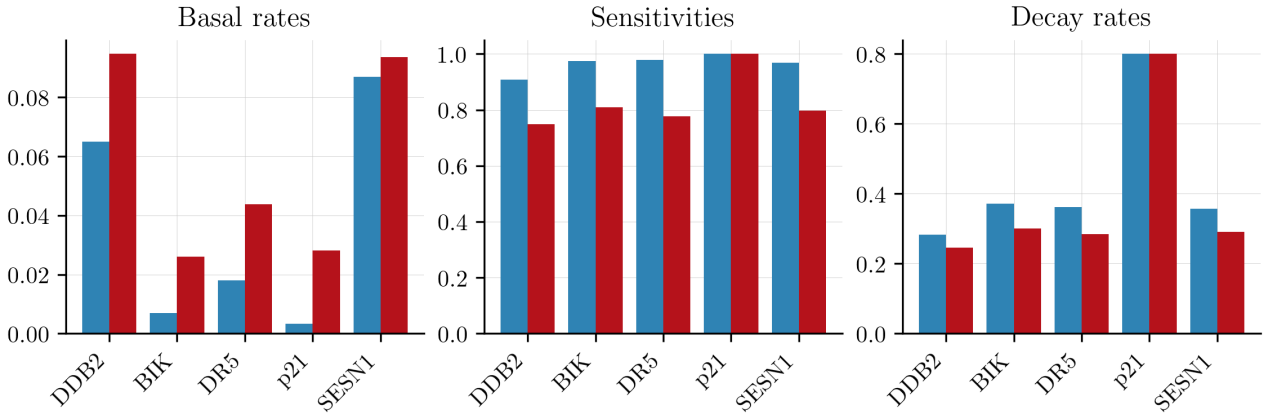


Figure 4.7: Inferred hyperparameters for p53 target genes from GPJax.

Similar to the findings obtained with ALFI, the estimates for target gene sensitivities and decay rates demonstrate good agreement with the experimental data, while the estimates for the basal transcription rates are inconsistent with the experimental results. Figure 4.8 compares the inferred hyperparameters for the p53 target genes obtained using GPJax (blue) and ALFI (red). The results are nearly identical, though GPJax yields marginally higher estimates for basal transcription rates and slightly lower estimates for sensitivities on average.

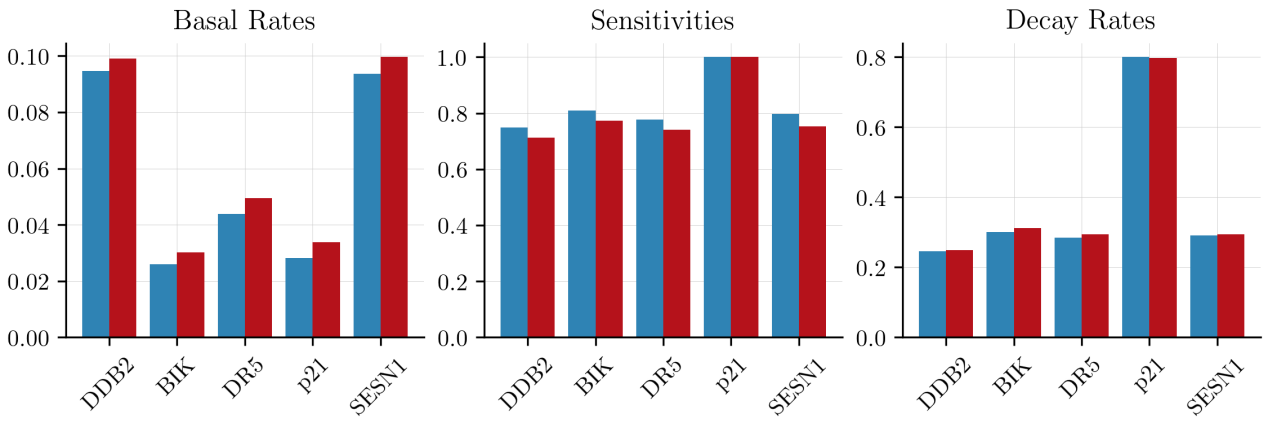


Figure 4.8: Comparison of inferred hyperparameters across GPJax and ALFI.

As a final step in the reproduction, the trained model was also used to predict the gene expressions for the five target genes at the test points. The predicted gene expressions for the five target genes are shown in Figure 4.9 The solid line is the posterior mean prediction, dashed lines represent two standard deviations, and experimental measurements are shown as crosses.

The predicted gene expressions using the LFM implemented in GPJax are consistent with the results obtained using ALFI. Across all posterior distributions for target gene expressions, the GPJax predictions show higher variance compared to the ALFI predictions, and neither model shows zero variance at the experimental data points. This discrepancy is likely due to the small dataset size and the high noise levels in the experimental data. To verify this, the model noise was fixed to 0, trained under the same conditions, and the results are shown in Figure 4.10 below.

Figure 4.10 shows much smaller variance in the posterior distribution at the experimental data points, highlighting the impact of the noise level on the model's predictions. The persistent lack of zero variance at these points, however, suggests that other factors may be contributing to the model's inability to fit the data exactly. This can be attributed either to the small dataset size or to the kernel's construction, which was designed to model the latent activity of p53 rather than the target gene expressions directly.

In conclusion, the results obtained using GPJax are consistent with those obtained using ALFI and comparable to those obtained by **(N. Lawrence et al., 2006)**, with the model accurately predicting the latent activity profile of p53 and the gene expressions of the target genes. The hyperparameter estimates for the basal transcription rates, sensitivities, and decay rates of the target genes are in good agreement with the experimental data, with the decay rate of p21 fixed at 0.8 and the sensitivity of p21 fixed at 1.0. In the following section, the performance of the GPJax model is explored further by ablating training data and running the model on different gene replicas.

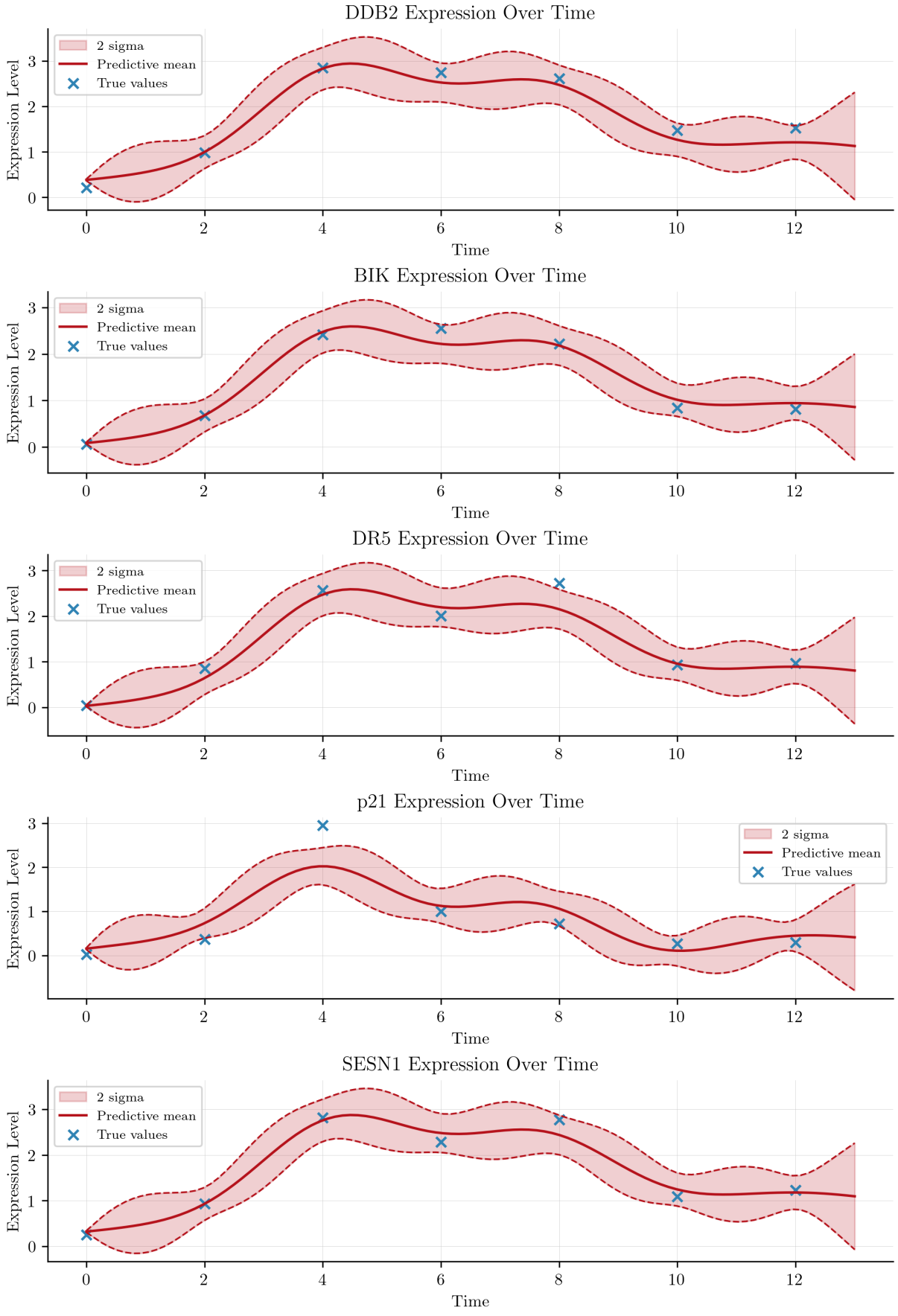


Figure 4.9: Predicted gene expressions for p53 target genes from GPJax.

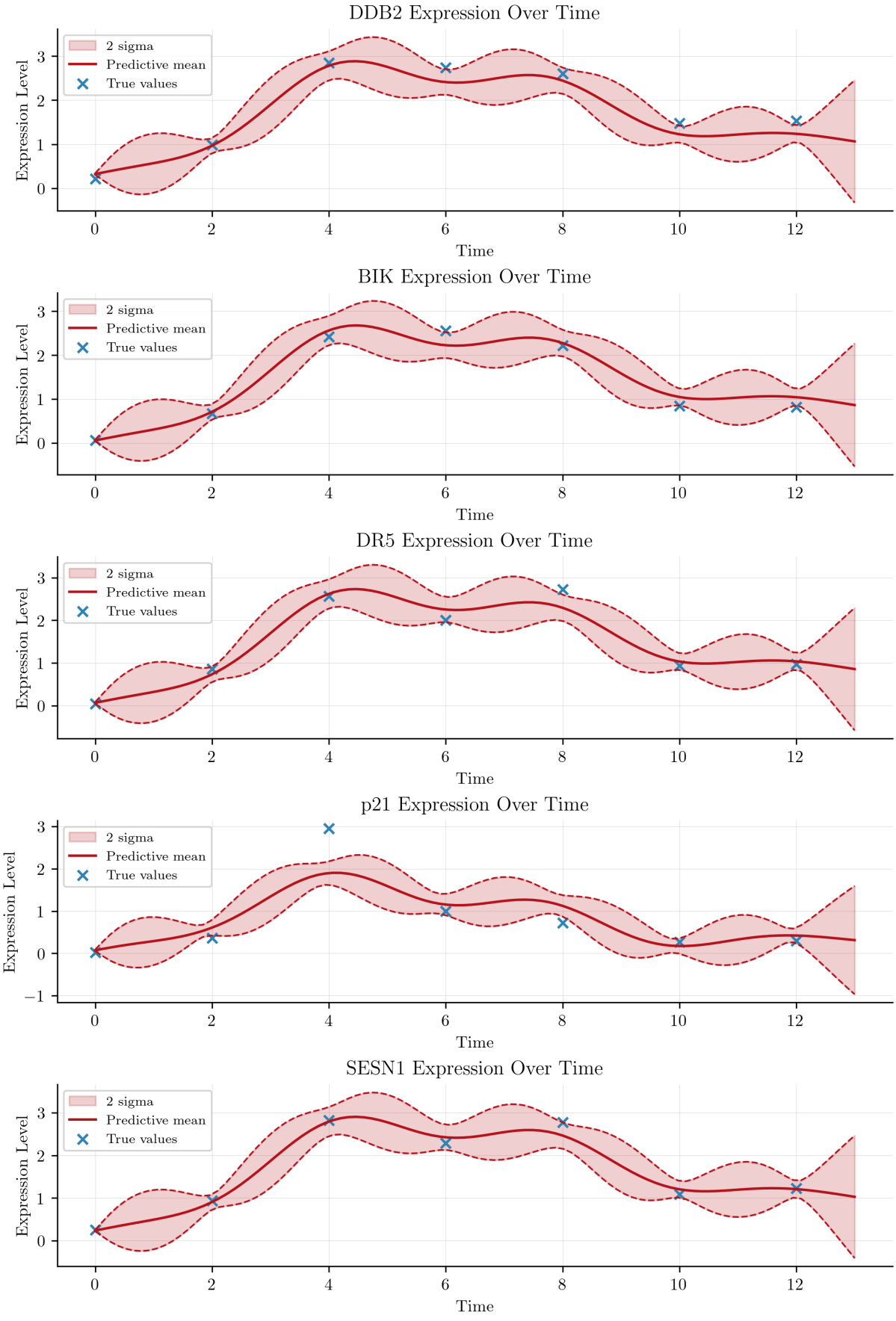


Figure 4.10: Predicted gene expressions for p53 target genes from GPJax with zero noise.

Going Further

5.1 Ablation Study

In both the original LFM paper by **(N. Lawrence et al., 2006)** and the HVDM paper by **(Barenco et al., 2006)**, the authors focused on modelling the latent activity of the p53 transcription factor using gene expression data from five known target genes. The custom LFM implemented in GPJax dynamically initialises the hyperparameters of the mean function and kernel based on the number of genes present in the training data. Therefore, to investigate the model's performance when trained on fewer genes, the model was trained on the following ablated datasets:

- **Subset 1** - four genes: DDB2, BIK, DR5, and p21.
- **Subset 2** - four genes (without p21): DDB2, BIK, DR5, and SESN1.
- **Subset 3** - three genes: BIK, DR5, and p21.
- **Subset 4** - two genes: DR5, and p21.
- **Subset 5**: p21.
- **Subset 6**: DR5.

The aim of this ablation study was to evaluate the accuracy of the LFM when trained with a reduced number of genes in predicting p53's latent activity. The study also explored scenarios where no hyperparameters were fixed, in the ablated datasets omitting p21. All posterior distributions presented in the following sections were obtained using the same training conditions as the original model, except for the fixed hyperparameters.* Gene expression predictions for each ablation subset are provided in Appendix B, along with a brief discussion of these findings.

5.1.1 Inference with four target genes

The posterior mean function of the LFM trained on four target genes in Figures 5.1 and 5.2 shows a good fit to the experimental data. These posterior distributions capture the latent activity of p53 with accuracy comparable to that achieved when trained on the entire dataset. Qualitatively, the posterior mean function trained on DDB2, BIK, DR5, and SESN1 (no hyperparameter fixing) fits the experimental data better than when trained on DDB2, BIK, DR5, and p21 (with hyperparameter fixing), although the posterior variance of the latter matches the posterior variance observed in Figure 4.6.

*The marginal log-likelihood (MLL) was optimised using the Adam optimiser with a learning rate of 0.01 over 150 iterations and a jitter term of 1×10^{-4} was added to the diagonal of the covariance matrix.

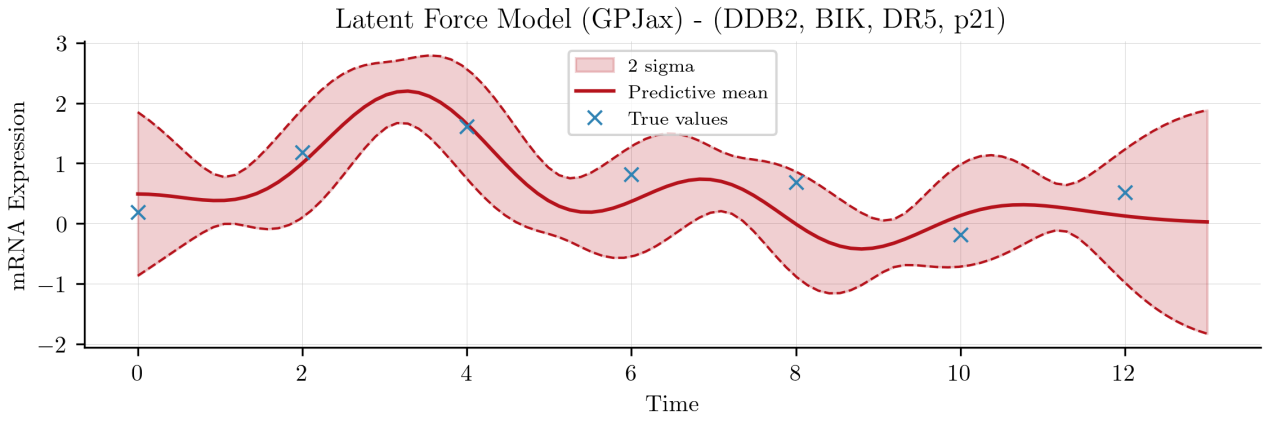


Figure 5.1: Predicted p53 activity profile from GPJax with four target genes.

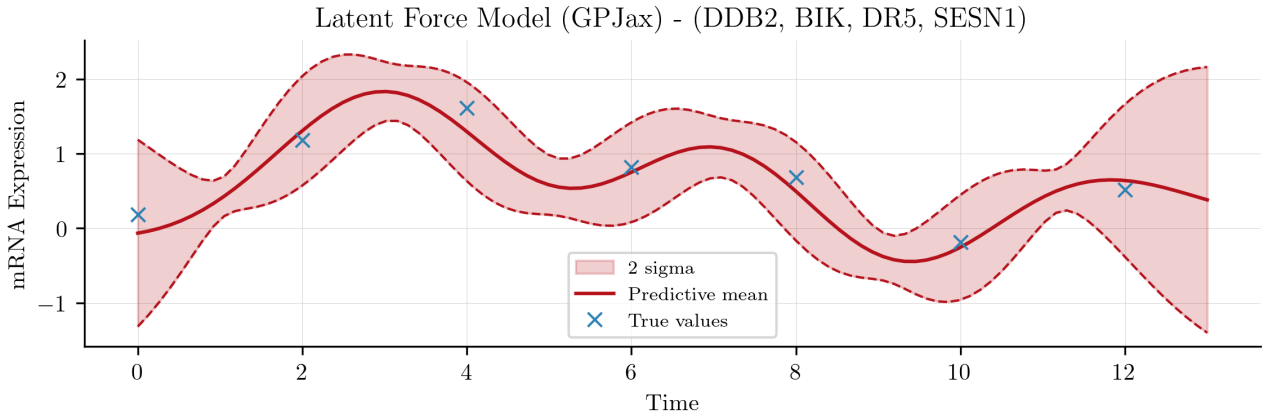


Figure 5.2: Predicted p53 activity profile from GPJax with four target genes.

5.1.2 Inference with three target genes

Next, the training dataset was reduced to only contain three target genes. The posterior mean function of the LFM trained on BIK, DR5, and p21 is shown in Figure 5.3. The model captures the latent activity of p53 with a degree of accuracy comparable to that achieved when trained on the entire dataset, although the posterior mean shows less oscillatory behaviour around the later timepoints than that obtained from the full dataset.

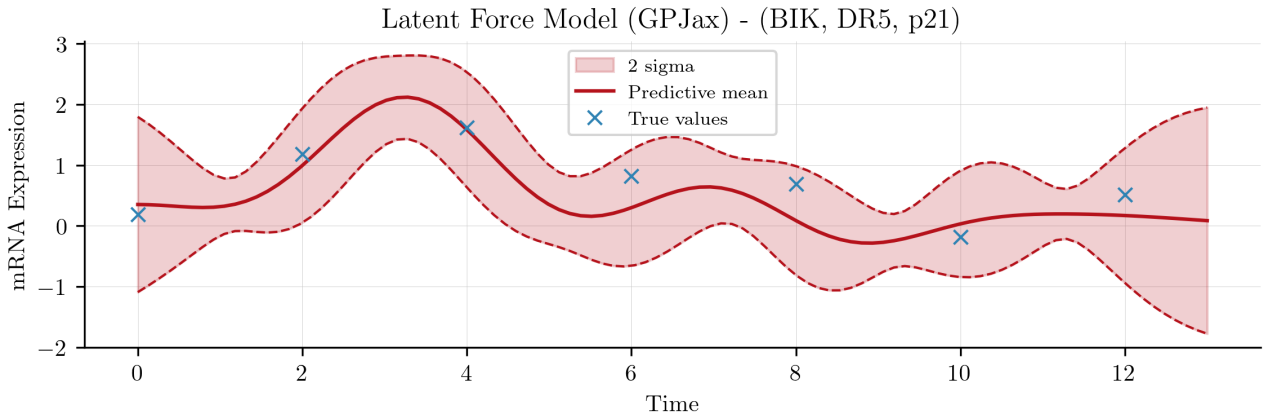


Figure 5.3: Predicted p53 activity profile from GPJax with three target genes.

5.1.3 Inference with two target genes

Using two p53 target genes, DR5 and p21, to train the LFM resulted in the posterior distribution shown in Figure 5.4. This distribution is qualitatively identical to the posterior distribution obtained when trained on the BIK, DR5, and p21 target genes, and maintains a relatively good fit to the experimental data for the first three timepoints.

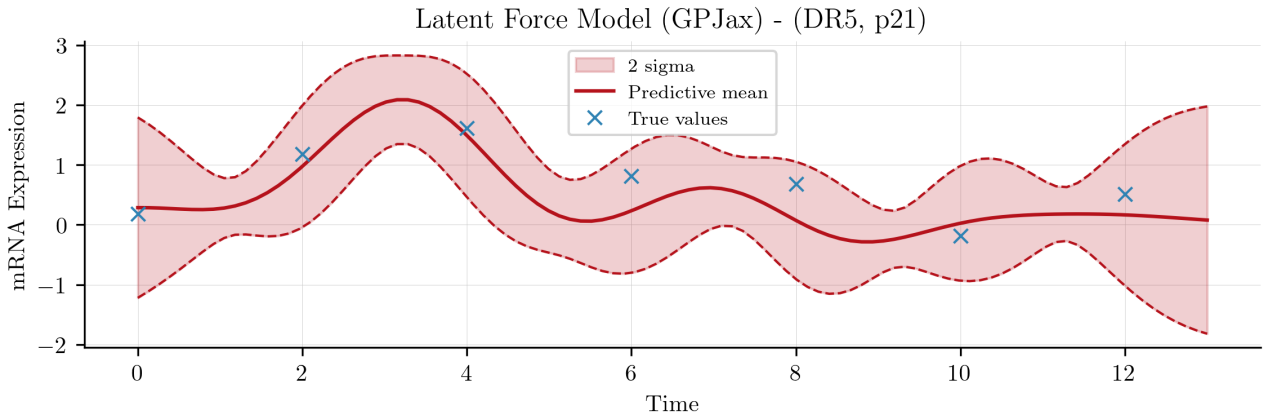


Figure 5.4: Predicted p53 activity profile from GPJax with two target genes.

5.1.4 Single gene inference

The posterior distributions of the linear force model trained on a single gene differ significantly from those obtained from multiple genes. Figure 5.5 shows the posterior distribution of the LFM trained on p21 gene expression data (with its hyperparameters fixed) and Figure 5.6 shows the posterior distribution of the LFM trained on DR5 gene expression data alone. When trained only on p21 data, the LFM's posterior mean function fails to capture the latent activity of p53 accurately. After time 6, the posterior variance remains almost uniform, and the posterior mean approaches zero. In contrast, the posterior mean function obtained from training on DR5 data alone shows a qualitatively better fit to that obtained from training on p21 data. This is attributed to the fixing of p21's hyperparameters, which reduces the model's flexibility.

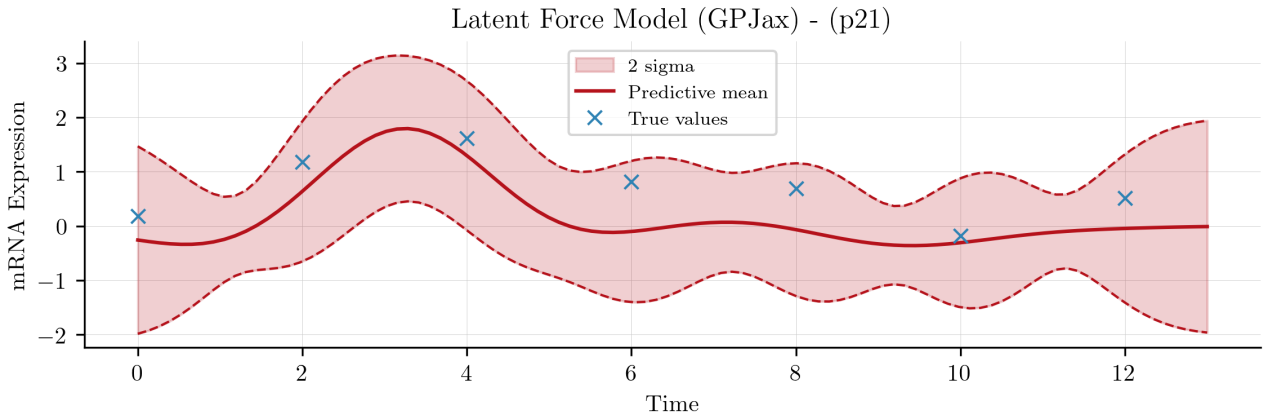


Figure 5.5: Predicted p53 activity profile from GPJax with p21 as the target gene.

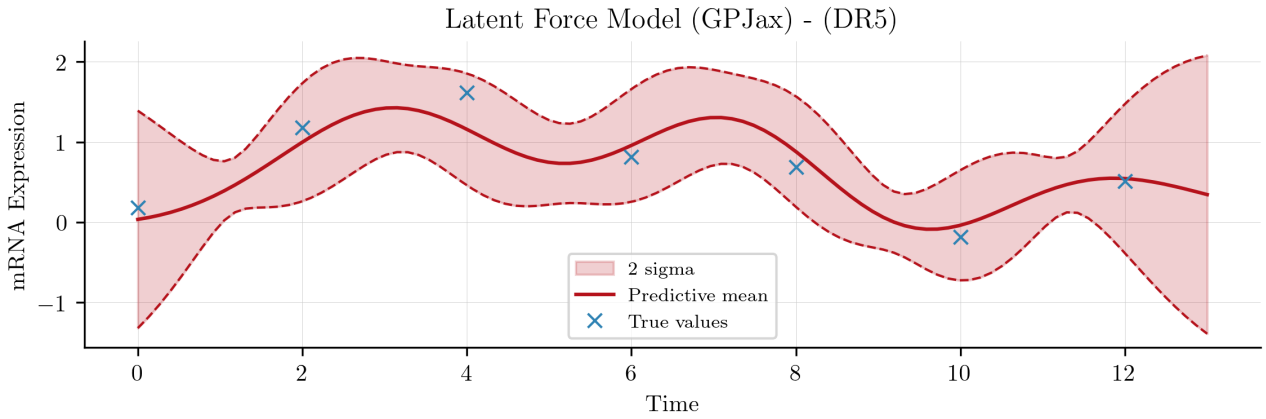


Figure 5.6: Predicted p53 activity profile from GPJax with DR5 as the target gene.

5.2 Gene Replicas

So far, both the results obtained using ALFI and those from the custom GPJax model have used only the first replica of the gene expression data. The similarity of these results to those reported by **(N. Lawrence et al., 2006)** suggests that these authors also used the first replica for their analysis. As an extension to the work carried out by **(Moss et al., 2022)** and **(N. Lawrence et al., 2006)**, the custom GPJax model was trained on the second and third replicas of the gene expression data to investigate the model's performance on different gene replicas.

The predicted latent activity profile for the p53 transcription factor from the second and third gene replicas using GPJax are shown in Figures 5.7 and 5.8 below.

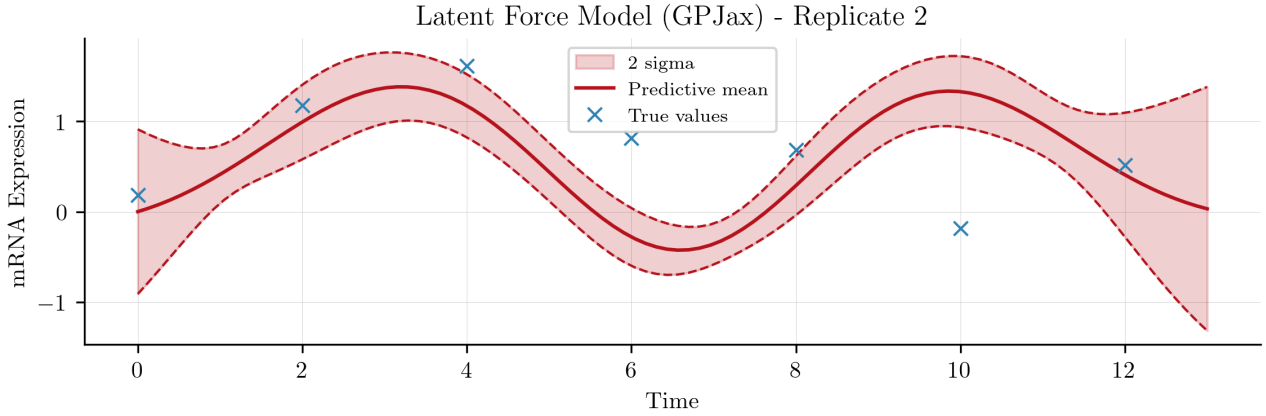


Figure 5.7: Predicted p53 activity profile from GPJax with the second gene replica.

For the second replica, the posterior distribution significantly deviates from that of the first replica and does not align with the experimentally inferred activity profile of p53. Training the LFM using the third replica yields a posterior distribution that more closely aligns with the experimental data. Although the posterior mean of this distribution does not match the experimental data as closely as the first replica, the two-sigma confidence interval captures the latent activity of p53 with an accuracy comparable to that of the first replica.

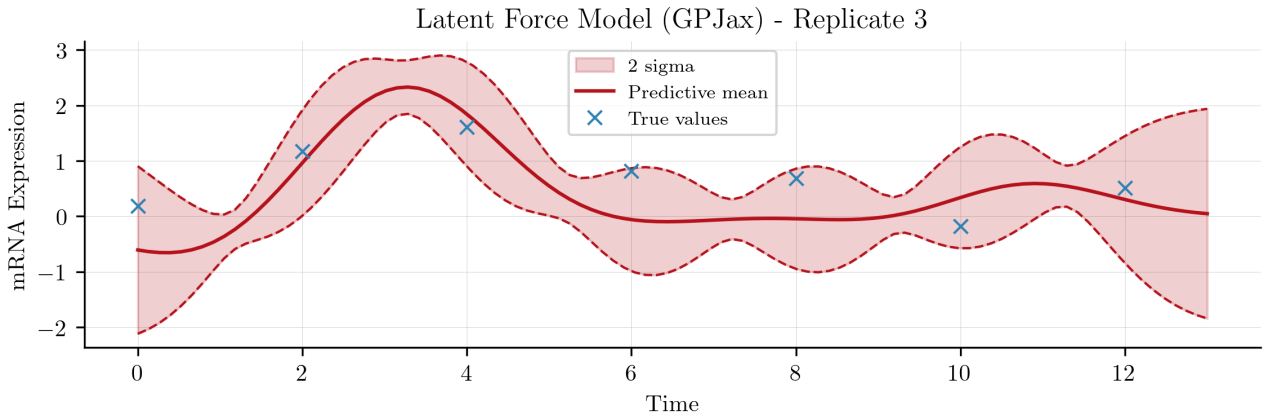


Figure 5.8: Predicted p53 activity profile from GPJax with the third gene replica.

To investigate whether these poor results were due to the implementation in GPJax, the same analysis was carried out using ALFI. The results obtained using ALFI are shown in Figures 5.9 and 5.10 below.

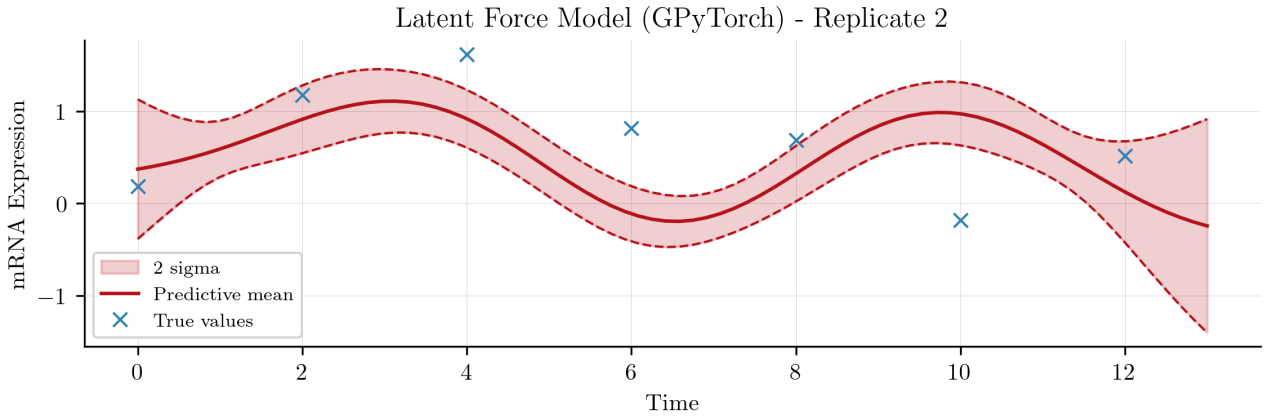


Figure 5.9: Predicted p53 activity profile from ALFI with the second gene replica.

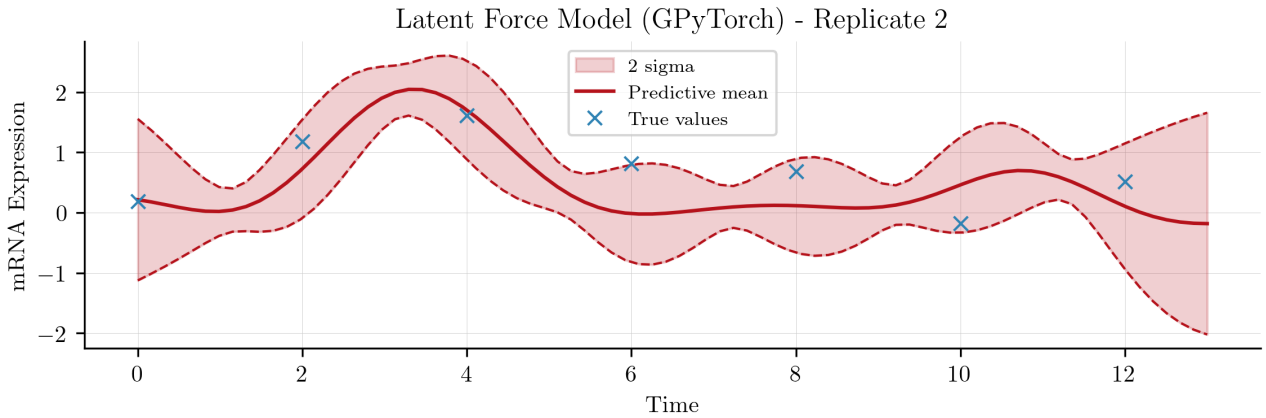


Figure 5.10: Predicted p53 activity profile from ALFI with the third gene replica.

The results obtained using ALFI are consistent with those obtained using GPJax: The posterior distribution obtained from the second gene replica does not align with the experimentally inferred activity profile of p53, while the posterior distribution obtained from the third gene

replica more closely aligns with the experimental data. This suggests that the poor results obtained from the second gene replica are not due to the implementation in GPJax, but rather to the quality of the data itself.

Conclusion

In this work, a latent force model implemented in GPJax was used to reproduce the results obtained by **(N. Lawrence et al., 2006)** and **(Barenco et al., 2006)** in modelling the latent activity of the p53 transcription factor from the expression levels of five target genes. The requirement for specific kernels based on the assumption of a linear response model was addressed by significantly customising the existing functions provided by the GPJax library and delving deeper into the nature of JAX pytrees. Bypassing many of GPJax's out-of-the-box features enabled the implementation of a custom latent force model which was able to accurately reproduce the results obtained by **(N. Lawrence et al., 2006)**. The results obtained with GPJax were thoroughly validated and compared in detail with those from ALFI, a latent force model library built on GPyTorch. **(Moss et al., 2022)**

The LFM implemented in this work was then used to predict the gene expressions of the target genes, showing good agreement with the experimental data and producing near-identical posterior distributions to those obtained using ALFI. The hyperparameters of the model were also compared with the experimental estimates, showing good agreement for the sensitivities and decay rates of the target genes.

An ablation study was conducted to investigate the model's performance when trained on a reduced number of genes. The results showed that the model was able to accurately predict the latent activity of p53 when trained on as few as three target genes, with the decay rate of p21 fixed at 0.8 and the sensitivity of p21 fixed at 1.0. This finding challenges the need to train the model with all five target genes and suggests that the initial dataset of five genes may not have provided substantial additional information.

The model's performance was also evaluated on different gene replicas, and results showed accurate predictions of the latent activity of the p53 transcription factor when trained on the first replica. However, this performance was inconsistent across different gene replicas, with the posterior distributions obtained from the second and third replicas deviating significantly from the experimental data. Such inconsistency, observed with both GPJax and ALFI, underscores the critical role of open science. The absence of a publicly available codebase and detailed methodological descriptions in the original study complicates the verification of whether the selection of data was influenced by performance metrics. Sharing data and code is essential for ensuring reproducibility and transparency in scientific research. This facilitates the validation of results and fosters a collaborative environment that enhances the reliability and generalisability of scientific findings.

References

Fei, Peiwen and Wafik S El-Deiry (Aug. 2003). "P53 and radiation responses". In: *Oncogene* 22.37, pp. 5774–5783. ISSN: 1476-5594. DOI: 10.1038/sj.onc.1206677. URL: http://dx.doi.org/10.1038/sj.onc.1206677.

Khanin, Raya et al. (Sept. 2005). "Computational inference of regulator activity in a single input motif from gene expression data". In: *BMC Bioinformatics* 6.S3. ISSN: 1471-2105. DOI: 10.1186/1471-2105-6-s3-s7. URL: http://dx.doi.org/10.1186/1471-2105-6-S3-S7.

Barenco, Martino et al. (2006). In: *Genome Biology* 7.3, R25. ISSN: 1465-6906. DOI: 10.1186/gb-2006-7-3-r25. URL: http://dx.doi.org/10.1186/gb-2006-7-3-r25.

Lawrence, Neil, Guido Sanguinetti, and Magnus Rattray (2006). "Modelling transcriptional regulation using Gaussian Processes". In: 19. URL: https://proceedings.neurips.cc/paper_files/paper/2006/hash/f42c7f9c8aeab0fc412031e192e2119d-Abstract.html.

Rasmussen, Carl Edward and Christopher K. I. Williams (2006). *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, pp. I–XVIII, 1–248. ISBN: 026218253X.

Gibney, E R and C M Nolan (May 2010). "Epigenetics and gene expression". In: *Heredity* 105.1, pp. 4–13. ISSN: 1365-2540. DOI: 10.1038/hdy.2010.54. URL: http://dx.doi.org/10.1038/hdy.2010.54.

Alberts, Bruce et al. (2013). "Differential Control of Transcription and Translation Underlies Changes in Cell Function". In: *Essential Cell Biology*. 4th ed. Philadelphia, PA: Garland Publishing. Chap. 2.3, p. 227. ISBN: 9780815344551.

Bumgarner, Roger (Jan. 2013). "Overview of DNA Microarrays: Types, Applications, and Their Future". In: *Current Protocols in Molecular Biology* 101.1. ISSN: 1934-3647. DOI: 10.1002/0471142727.mb2201s101. URL: http://dx.doi.org/10.1002/0471142727.mb2201s101.

Kingma, Diederik and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)*. San Diego, CA, USA.

Lawrence, Neil D. (July 2015). *Latent Force Models: Bridging the Divide between Mechanistic and Data Modelling Paradigms*. Talk presented at Max Planck Institute for Intelligent Systems. Stuttgart. URL: https://staffwww.dcs.shef.ac.uk/people/N.Lawrence/talks/lfm_stuttgart15.pdf.

Moss, Jacob D. et al. (Jan. 24, 2022). *Approximate Latent Force Model Inference*. DOI: 10.48550/arXiv.2109.11851. arXiv: 2109.11851[cs]. URL: http://arxiv.org/abs/2109.11851.

Pinder, Thomas and Daniel Dodd (2022). "GPJax: A Gaussian Process Framework in JAX". In: *Journal of Open Source Software* 7.75, p. 4455. DOI: 10.21105/joss.04455. URL: https://doi.org/10.21105/joss.04455.

Derivations

A.1 Solving the Differential Equation

Equation (2.6), which relates a given gene j's expression level $x_j(t)$ at time t to the concentration of the regulating transcription factor protein $f(t)$ can be solved using the method of integrating factors:

$$\frac{\mathrm{d}y}{\mathrm{d}t} + P(t)y = Q(t)$$

Where $P(t) = D_j$ and $Q(t) = B_j + S_j f(t)$. The integrating factor $\mu(t)$ is found by integrating $P(t)$ with respect to t:

$$\mu(t) = e^{\int P(t)\mathrm{d}t} = e^{\int D_j \mathrm{d}t} = e^{D_j t} \tag{A.1}$$

Next, the product of the integrating factor and the differential equation is taken:

$$e^{D_j t}\frac{\mathrm{d}x_j(t)}{\mathrm{d}t} + e^{D_j t} D_j x_j(t) = e^{D_j t}(B_j + S_j f(t)) \tag{A.2}$$

The left-hand side of Equation (A.2) can be rewritten as the derivative of the product of the integrating factor and the dependent variable (i.e. The derivative of $e^{D_j t}x_j(t)$):

$$\frac{\mathrm{d}}{\mathrm{d}t}(e^{D_j t}x_j(t)) = e^{D_j t}(B_j + S_j f(t)) \tag{A.3}$$

Equation (A.3) can be integrated with respect to t from 0 to t to obtain the general solution:

$$\int_0^t \frac{\mathrm{d}}{\mathrm{d}u}(e^{D_j u}x_j(u))\mathrm{d}u = \int_0^t e^{D_j u}(B_j + S_j f(u))\mathrm{d}u \tag{A.4a}$$

$$e^{D_j t}x_j(t) - e^{D_j 0}x_j(0) = \int_0^t e^{D_j u}(B_j + S_j f(u))\mathrm{d}u \tag{A.4b}$$

Note that $e^{D_j \cdot 0}x_j(0)$ simplifies to $x_j(0)$.

$$e^{D_j t}x_j(t) = x_j(0) + \int_0^t e^{D_j u}(B_j + S_j f(u))\mathrm{d}u \tag{A.5}$$

Assuming that the initial condition is $x_j(0) = A_j$:

$$
\begin{aligned}
e^{D_j t} x_j(t) &= \int_0^t e^{D_j u}(B_j + S_j f(u))\mathrm{d}u + A_j \\
e^{D_j t} x_j(t) &= \int_0^t e^{D_j u} B_j \mathrm{d}u + \int_0^t e^{D_j u} S_j f(u)\mathrm{d}u + A_j \\
e^{D_j t} x_j(t) &= \frac{B_j}{D_j}(e^{D_j t} - 1) + \int_0^t e^{D_j u} S_j f(u)\mathrm{d}u + A_j \\
x_j(t) &= \frac{B_j}{D_j}(1 - e^{-D_j t}) + e^{-D_j t}\int_0^t e^{D_j u} S_j f(u)\mathrm{d}u + A_j e^{-D_j t} \\
x_j(t) &= \frac{B_j}{D_j} + \left(A_j - \frac{B_j}{D_j}\right) e^{-D_j t} + S_j e^{-D_j t}\int_0^t e^{D_j u} f(u)\mathrm{d}u
\end{aligned} \tag{A.6a}
$$

Where Equation (A.6a) is the final solution for $x_j(t)$ (Equation (2.7)).

A.2 Convolution integrals & Covariance functions

A linear operator L has tow main properties:

- **Additivity**: $L(f + g) = L(f) + L(g)$
- **Homogeneity**: $L(cf) = cL(f)$

where f and g are functions and c is a constant.

For two functions $f(t)$ and $g(t)$, their convolution $(f * g)(t)$ is defined as:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)\mathrm{d}\tau$$

The proof that a convolution integral is a linear operator can be shown trivially by proving the above equation is additive and homogeneous.

Equation (2.9) can be re-written as:

$$L_j[f](t) = S_j e^{-D_j t}\int_0^t e^{D_j(u-t)} f(u)\mathrm{d}u$$

Which matches the structure of the convolution integral $(g * f)(t)$, where $g(t) = e^{-D_j t}$.

A.2.1 Gene Expression Covariance

Applying Equation (2.10) to Equation (2.8) to obtain the covariance function of $x_j(t)$ yields:

$$\begin{aligned}
\mathrm{Cov}(L_j[f](t), L_k[f](t')) &= \mathrm{Cov}\left(S_j e^{-D_j t}\int_0^t e^{D_j u} f(u)\mathrm{d}u, S_k e^{-D_k t'}\int_0^{t'} e^{D_k u} f(u)\mathrm{d}u\right) \\
&= S_j e^{-D_j t} S_k e^{-D_k t'} \mathrm{Cov}\left(\int_0^t e^{D_j u} f(u)\mathrm{d}u, \int_0^{t'} e^{D_k u} f(u)\mathrm{d}u\right) \\
&= S_j S_k e^{-D_j t - D_k t'} \int_0^t \int_0^{t'} \mathrm{Cov}\left(e^{D_j u} f(u), e^{D_k u'} f(u')\right) \mathrm{d}u'\mathrm{d}u \\
&= S_j S_k e^{-D_j t - D_k t'} \int_0^t e^{D_j u} \int_0^{t'} e^{D_k u'} k_{ff}(u, u')\mathrm{d}u'\mathrm{d}u \\
&= k_{x_j x_k}(t, t') \qquad \text{(A.7a)}
\end{aligned}$$

A.2.2 Cross covariance

$$\begin{aligned}
k_{x_j f}(t, t') = \mathrm{Cov}(L_j[f](t), f(t')) &= \mathrm{Cov}\left(S_j e^{-D_j t}\int_0^t e^{D_j u} f(u)\mathrm{d}u, f(t')\right) \\
&= S_j e^{-D_j t} \int_0^t \mathrm{Cov}\left(e^{D_j u} f(u), f(t')\right) \mathrm{d}u \\
&= S_j e^{-D_j t} \int_0^t e^{D_j u} \mathrm{Cov}(f(u), f(t'))\mathrm{d}u \\
&= S_j e^{-D_j t} \int_0^t e^{D_j u} k_{ff}(u, t')\mathrm{d}u \qquad \text{(A.8a)}
\end{aligned}$$

Ablation Study Gene expressions

Training the LFM on four genes to predict their expressions yielded similar results to training on the entire dataset. When p21 was included, with it's sensitivity and decay rate fixed, the predictions show in Figure B.1 display higher variance around the experimental data points compared to when p21 was ablated (Figure B.2. Further, in Figure B.1, the gene expression for DR5 is inconsistent with the experimental results. This inconsistency is also observed in Figure B.3, when the model was trained only on BIK, DR5, and p21. The gene expressions obtained when the LFM was trained on two genes (DR5 and p21) are shown in Figure B.4, and are comparable to the predictions from using three target genes in Figure B.3, qualitatively showing slightly higher variance.

When a single gene was used to train the LFM model, the resulting posterior distribution for the gene expression of that gene varies wildly depending on whether the hyperparameters of that gene were fixed or not. Figure B.5 shows the prediction for p21's expression when its sensitivity and decay rate were fixed, showing a poor fit and high variance, and Figure B.6 shows the same prediction without fixing its hyperparameters. In the latter figure, the posterior mean fits the experimental data much better, with two standard deviations capturing the expression at time 4. A similar fit is observed in Figure B.7 for DR5, indicating that fixing the hyperparameters in the latent force model results in worse predictions for gene expressions.

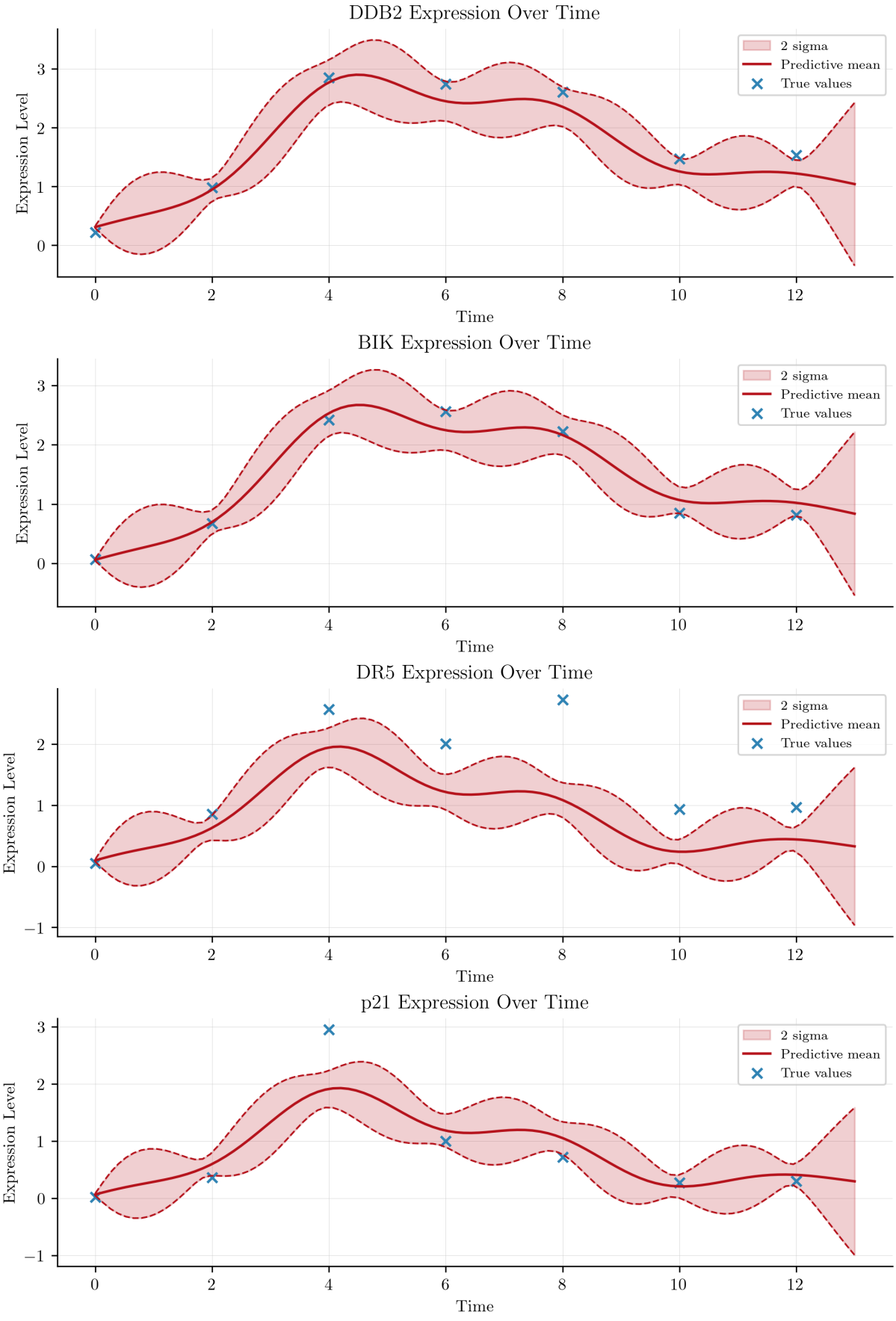


Figure B.1: Predicted gene expressions for four genes with p21.

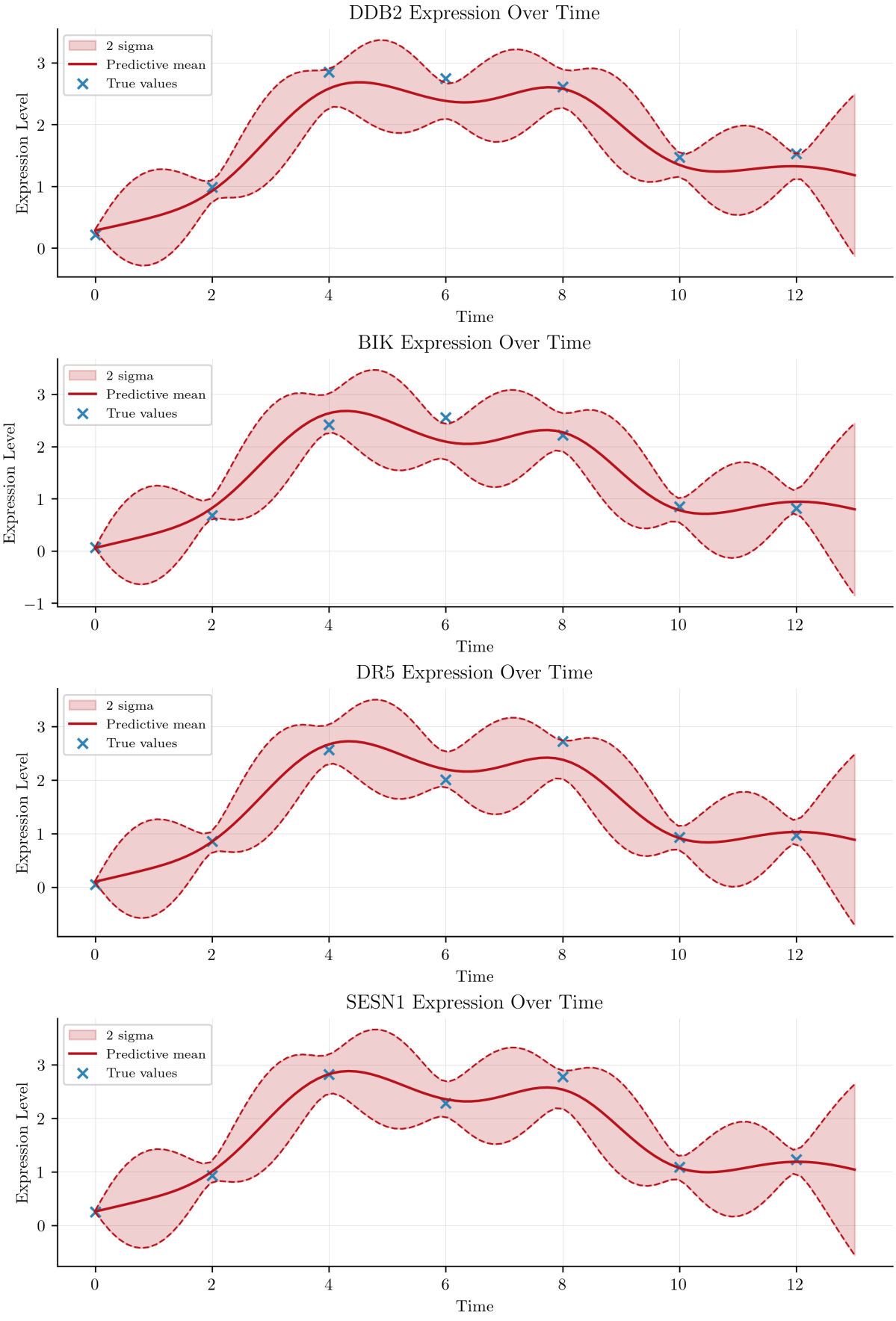


Figure B.2: Predicted gene expressions for four genes without p21.

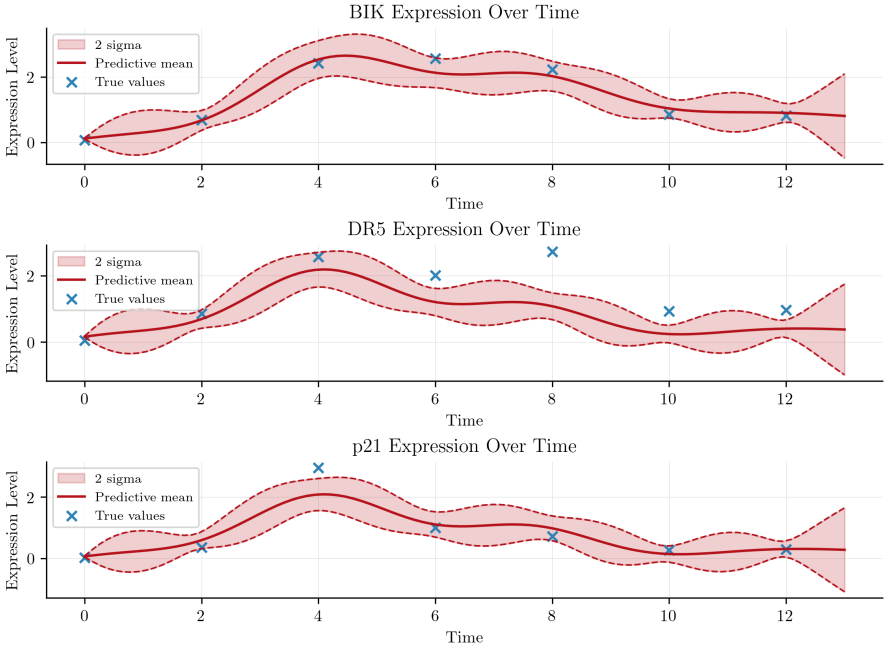


Figure B.3: Predicted gene expressions for three genes.

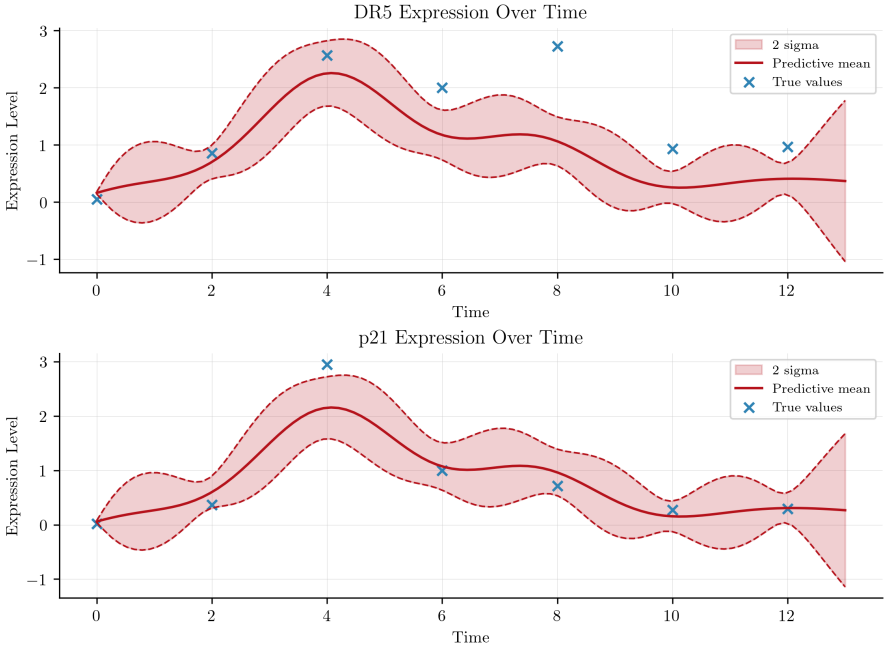


Figure B.4: Predicted gene expressions for two genes.

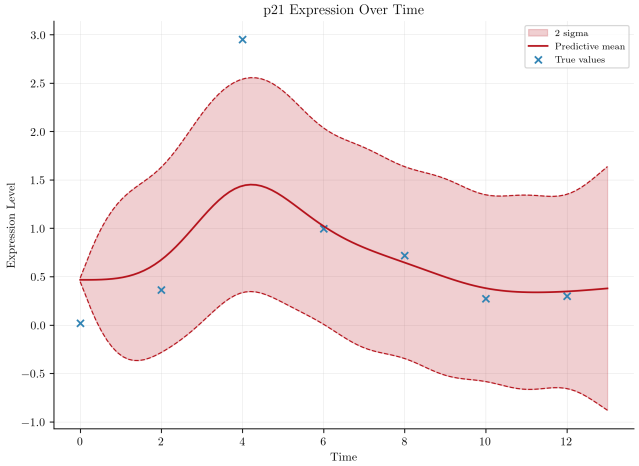


Figure B.5: Predicted gene expressions for p21 (fixed sensitivity and decay).

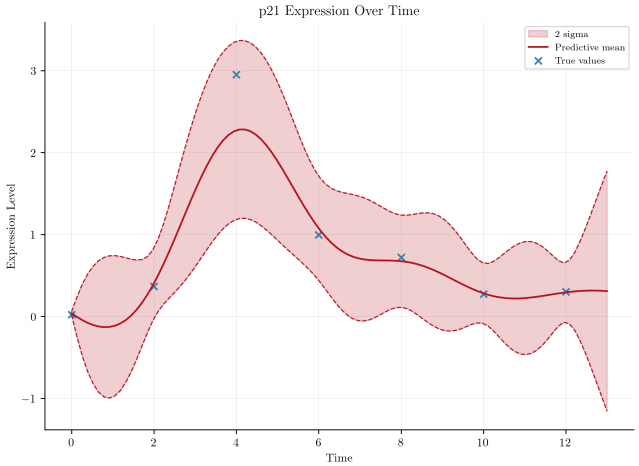


Figure B.6: Predicted gene expressions for p21.

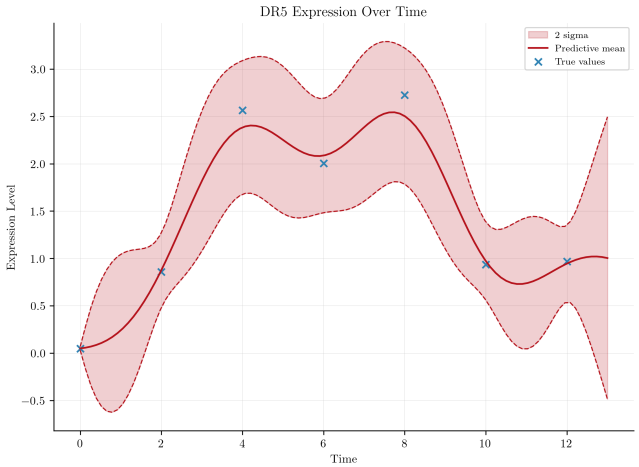


Figure B.7: Predicted gene expressions for DR5.

Generative AI Disclaimer

GitHub Copilot was used throughout the project as it is installed on my IDE (VS Code). It's use was limited to auto-completing lines of code where it made sense to do so (i.e, it was not used to blindly implement functionality, simply to complete code and/or docstring sentences). The use of GenAI LLMs such as ChatGPT and PerplexityAI was minimised throughout the development of the codebase and examples of some prompts used are shown below:

1. "Explain why I am getting a tracer error in JAX here"
2. "Generate NumPy style docs for this function"
3. "How can I use tabulate to format my results nicely on the CL?"

In regards to report writing, GenAI LLMs such as ChatGPT and PerplexityAI were used as editing tools to help with grammar and sentence structure. The use of these tools was limited to editing and proofreading the report and no content was generated by these tools. Examples of prompts used are shown below:

1. "Improve the grammar, structure, and coherency of this text"
2. "Check this text for spelling mistakes"
3. "Why does this LaTeX code not compile?"