University of Glasgow | School of Mathematics & Statistics

Parameter inference in dynamical systems with a nonparametric model

Supervisor: Dr. Mu Niu

Student: ZhongTian Zheng

March 2021

Contents

1 Introduction

In nature, there are many interactive relationships between quantities such as the population of predators and preys, where the preys decrease when the predators increase and the predators decrease when the preys decrease. To summarize this sort of discipline in a concise and accurate fashion, the concept of the dynamical system is invented to describe the evolution of a set of interactive states dependent on time (Strogatz, Steven H 1994). We note that many dynamical systems are given in the form of nonlinear ordinary differential equations [1] (ODEs), hence make the parameter inference computationally expensive using standard iterative inference procedures. To avoid this demanding computation, the gradient matching method is introduced.

Before going into details of the gradient matching method, it is worthwhile to understand the conventional approaches that have been widely used for parameter estimation in dynamical systems. In the classical methods, parameter estimation is treated as an optimization problem where we search for the optimal set of parameters that produce the values close to observations. Particularly, one optimization algorithm inspired by the process of natural selection is extremely popular, also known as the evolutionary algorithm (Mitchell M 1998). This idea is borrowed in many methods featuring Genetic algorithm (Holland JH 1975), Differential Evolution (Storn R, Price K 1997). Given a guess for parameters in a dynamical system, the goodness of the guess needs to be assessed by explicitly solving the ODEs of the dynamical system, then measuring the discrepancy between the solved values and the observed values in data. Therefore, the ODEs will be solved as many times as the guesses are made, which is a huge computational cost when the parameter space is huge and the ODEs are complex. And another tricky issue is, solving ODEs often requires initial values, but we only have noisy data. It means we need to also search initial values around the observed values. In this work, we will later look at the Differential Evolution's application in a specific example.

Compared with the classical methods, the mentioned gradient matching method measure does not measure the discrepancy between observations and predicted values of a certain state in a dynamical system, instead circumvents solving ODEs for predicting a certain state value in the dynamical system and compare the gradients instead. Firstly, it interpolates time series for each state in a dynamical system to get smooth lines, and then optimize the parameters in nonlinear ODE by minimizing the difference between the time derivatives predicted by ODE and those of the interpolants (the smooth lines) at the observed time-points.

Practically, under the scheme of gradient matching, it can be classified into non-

[1] In mathematics, an ordinary differential equation (ODE) is a differential equation containing one or more functions of one independent variable and the derivatives of those functions. The term ordinary is used in contrast with the term partial differential equation which may be with respect to more than one independent variable. –Wikipedia

parametric and parametric realizations for different interpolation methods, such as P-splines (Liang and Wu 2008), Parallel tempering (Campbell and Steele 2012), Gaussian process ((Dondelinger et al. (2013), Calderhead et al. (2009), Barber and Wang (2014)). This work primarily focuses on kernel ridge regression, which is another realization for interpolation, using the basis function from a Reproducing Hilbert Space (RKHS, Gonzalez et al 2013, 2014) and its programming implementation in R package KGode (Mu Niu 2020).

2 Methods

2.1 Dynamical system

Consider a dynamical system consisted of *r* states, its states at time *t* can be denoted by a time-dependent vector:

$$\vec{x}(t) = \left(x_1(t), x_2(t) \dots x_r(t)\right) \; r \geq 2 \tag{1}$$

And the system can be fully defined by non-linear ODEs, the time derivative of a certain state $\boldsymbol{s}$ can be expressed in the form:

$$\dot{x}_s(t) = \frac{dx_s(t)}{dt} = f_s(x(t), \boldsymbol{\theta}) \tag{2}$$

Where $\boldsymbol{\theta}$ can be either a constant or a vector of constants.
In this work, we assume for each state $\boldsymbol{x_s(t)}$, $\boldsymbol{y_s}$ are observed with additive noise at distinct n time point, that is:

$$\overrightarrow{y_s} = \overrightarrow{x_s} + \boldsymbol{\varepsilon_s} \tag{3}$$

Where $\overrightarrow{x_s} = (x_s(t_1) \dots, x_s(t_n))$, $\boldsymbol{\varepsilon_s} \sim N(0, \sigma^2 * I)$
In this work, we will particularly focus on Lotka-Volterra and bio-pathway systems.

2.2 Reproducing Kernel Hilbert Space

RKHS is a particular function Space such that the kernel can reproduce any function in the space including the kernel itself (details to follow). In this work, we focus on its application to interpolation. In this section, the definition of RKHS and its property will be introduced and how kernel ridge regression is conducted for interpolation. As this work discusses only ODEs concerning time, the notation of some definition will be adapted to our topic (Murphy (2012), Macdonald and Husmeier (2015) are referred for the following introduction).

Definition of Positive definite matrix: Let A be a real symmetric matrix. The quadratic form of A is $c^T Ac$.Then A is positive definite if for any $c \neq 0$, the quadratic form of A is positive

$$c^T A c > 0 \tag{3}$$

Definition of Kernel function: A kernel function is a real-valued function of two arguments, $K(x, x') \in \mathbb{R}$, for $x, x' \in \chi$, where χ is some abstract space, in the context of a dynamical system, x, x' represents time points within $[t_1, t_n]$

Definition of Mercer (positive definite) kernels:
The kernel ridge regression is based on Mercer (positive definite) kernels (Mercer (1909)) $K(t_i, t_j)$, which satisfy the requirement that the Gram matrix, defined by

$$\boldsymbol{K} = \begin{pmatrix} K(t_1, t_1) & \dots & K(t_1, t_n) \\ \dots & \dots & \dots \\ K(t_n, t_1) & \dots & K(t_n, t_n) \end{pmatrix} \tag{4}$$

be positive definite for any set of inputs $\{t_i\}_{i=1}^n$.

As Gram matrix is positive definite, we can compute an eigenvector decomposition of it as follows,

$$K = U^T \Lambda U \tag{5}$$

Where Λ is a diagonal matrix of eigenvalues $\lambda_i > 0$ (property of positive definite matrix). Consider an element of gram matrix:

$$K(t_i, t_j) = \left(\Lambda^{\frac{1}{2}} U_{:,i}\right)^T \left(\Lambda^{\frac{1}{2}} U_{:,j}\right) \tag{6}$$

Let us define $\phi(t_1) = \Lambda^{\frac{1}{2}} U_{:,i}$. Then we can write

$$K(t_i, t_j) = \phi(t_i)^T \phi(t_j) \tag{7}$$

Therefore, we see that the entries in the kernel matrix can be computed by performing an inner product of some feature vectors that are implicitly defined by the eigenvectors U. In other words, if the kernel is Mercer, then there exists a function ϕ mapping $t_i \in [t_1, t_n]$ to $\mathbb{R}^D$ (D is a potentially infinite-dimensional space) such that

$$K(t_i, t_j) = \phi(t_i)^T \phi(t_j) \tag{8}$$

Where ϕ depends on the eigen*functions* of K.
Note the above introduction is a simplification of Mercer's work and Murphy's elaboration and is enough to illustrate the RHKS interpolation in the context of the dynamical system.

Mercer's theorem reveals that any positive definite kernel function can be represented in terms of eigenfunctions:

$$K(t_i, t_j) = \sum_{k=1}^{\infty} \lambda_k \psi_k(t_i) \psi_k(t_j) \tag{9}$$

Where λ_k *is eigenvalues and* ψ_i *is eigenfunctions*[2].
These ψ_k *form an orthonormal basis for a function space*

$$\mathcal{H} = \{f: f(t) = \sum_{k=1}^{\infty} f_k \psi_k(t) \ , \sum_{k=1}^{\infty} \frac{{f_k}^2}{\lambda_k} < \infty\}.$$

The inner product between two functions $f(t) = \sum_{k=1}^{\infty} f_k \psi_k(t)$ *and* $g(t) = \sum_{k=1}^{\infty} g_k \psi_k(t)$ *in the space in* $\mathcal{H}$ *is defined as*

$$< f, g >_{\mathcal{H}} \triangleq \sum_{k=1}^{\infty} \frac{f_k g_k}{\lambda_k} \tag{10}$$

Which Murphy (2012) shows implies that

$$< K(t_i, \cdot), K(t_j, \cdot) >_{\mathcal{H}} = K(t_i, t_j) \tag{11}$$

Which is known as the reproducing property and the space of functions H is called a reproducing kernel Hilbert space (RKHS).

2.3 Kernel-based interpolation

The theoretical basis of the method - Representer Theorem
The interpolation is done by finding a function $f \in \mathcal{F}$, which minimize a regularized risk function $R_{reg}[f]$. The usage of regularization is firstly proposed by Tikhonov and V. Y. Arsenin in 1977, without which minimizing the empirical risk can lead to numerical instabilities and bad generalization performance. It restricts the class of possible minimizers $\mathcal{F}$ (with $f \in \mathcal{F}$) of the empirical risk function $R_{emp}[f]$ such that $\mathcal{F}$ becomes a compact set. In addition, $R_{emp}[f]$ is assumed to be continuous in f.

Theorem Operator Inversion Lemma: Let X be a compact set and let the map $f: \mathrm{X} \to Y$ *be continuous. Then there exists an inverse map* $f^{-1}: f(X) \to X$ *that is also continuous.* ***(Proof seen in Schöelkopf, B., and Smola, A. (2002))***

The operator inversion lemma shows that for compact $\mathcal{F}$, the inverse map from the minimum of the empirical risk functional $R_{emp}[f]: \mathcal{F} \to \mathbb{R}$ to its minimizer f_m is continuous and the search for f_m becomes an optimization problem.

Instead of specifying a compact set $\mathcal{F}$ which may be unrealistic in practice, the regularization term $\Omega(f)$ is added to the $R_{emp}[f]$. That is,

$$R_{reg}[f] = R_{emp}[f] + \lambda\Omega(f) \tag{12}$$

Where $\lambda > 0$ is the regularization parameter which specifies the trade-off between minimization of $R_{emp}[f]$ and the smoothness or simplicity which is enforced by small $\Omega(f)$). The explicit form of form a minimizer $R_{reg}[f]$ is given by the theorem

[2] Wikipedia: Eigenfunction

Theorem Representer Theorem: Denote by $\Omega: [0,\infty) \to \mathbb{R}$ *a strictly monotonic increasing function, by* χ *a set, and by c:* $(\chi \times \mathbb{R}^2)^n \to \mathbb{R} \cup \{\infty\}$ *an arbitrary loss function. Then each minimizer* $f \in H$ *of the regularized risk*

$$c\big((x_1, y_1, f(x_1), \dots, (x_n, y_n, f(x_n)\big) + \Omega\big(||f||\big) \tag{13}$$

Admits a representation of the form

$$f(t) = \sum_{i=1}^{n} \alpha_i \, K(t, t_i) \tag{14}$$

Where n corresponds to the number of observations in the data. ***(Proof seen in in Schöelkopf, B., and Smola, A.(2002))***

Kernel ridge regression

To interpolate ***r*** time series for corresponding ***r*** states in our dynamical system, we intend to find ***r*** independent minimizers in ***r*** Reproducing Kernel Hilbert Space. As per the representer theorem, when the regularized risk meets the required form, the smooth interpolant of state ***s***, $g_s(t)$ can be expressed as a linear combination of kernel functions centered around ***n*** time points $\{t_i\}_{i=1}^{n}$ in our data:

$$g_s(t; \boldsymbol{b_s}) = \sum_{i=1}^{n} b_{si} \, K_s(t, t_i) \tag{15}$$

Where $b_{si} \in R$, and $K_s(t, t_i)$ is the chosen type of kernel with state-specific hyper-parameters.

The coefficients and hyper-parameters can be estimated by minimizing a regularized risk function with a range of choices for regularization, in this project, the squared norm of the Hilbert space is used. The regularized risk function is given by

$$\mathcal{L}(\boldsymbol{b_s}, \boldsymbol{\varphi_s}; \boldsymbol{\lambda_s}) = \sum_{i=1}^{n} \big(g_s(t_i) - y_s(t_i)\big)^2 + \boldsymbol{\lambda_s} \|q_s\|^2 \tag{16}$$

Where $\boldsymbol{b_s}$ is a vertical n-element-vector containing $\boldsymbol{b_{si}}$ as coefficients, $\boldsymbol{\lambda_s}$ is the strength of the regularization for each time series of state ***s*** and $\|q_s\|^2$ is the regularization term, the squared norm of $\mathcal{H}_s$:

$$\|q_s\|^2 = < \boldsymbol{b_s}^{\boldsymbol{T}} \begin{pmatrix} K_s(t, t_1) \\ \dots \\ \dots \\ K_s(t, t_n) \end{pmatrix}, \big(K_s(t, t_1) \dots K_s(t, t_n)\big) \, \boldsymbol{b_s} \geq \boldsymbol{b_s}^{\boldsymbol{T}} \, \boldsymbol{K_s} \, \boldsymbol{b_s} \tag{17}$$

Where $\boldsymbol{K_s}$ is the Gram matrix,

$$\boldsymbol{K_s} = \begin{pmatrix} K_s(t_1, t_1) & \dots & K_s(t, t_n) \\ \dots & \dots & \dots \\ K_s(t_n, t_1) & \dots & K_s(t_n, t_n) \end{pmatrix} \tag{18}$$

This kernel-based interpolation is called kernel ridge regression. The minimization of $\mathcal{L}(\boldsymbol{b_s}, \boldsymbol{\varphi_s}; \boldsymbol{\lambda_s})$ with respect to $\boldsymbol{b_s}$ given $\boldsymbol{\varphi_s}$ and $\boldsymbol{\lambda_s}$ can be solved by setting the gradients w.r.t $\boldsymbol{b_s}$ to 0, giving the solution

$$\boldsymbol{b_s} = (\boldsymbol{K_s} + \boldsymbol{\lambda_s I})^{-1} \boldsymbol{y_s} \tag{19}$$

Where the regularization parameter $\boldsymbol{\lambda_s}$ is chosen by k-fold cross-validation (k=3 by default in KGode) and $\boldsymbol{K_s}$ dependent on the hyper-parameter $\boldsymbol{\varphi_s}$. That is, the problem of minimizing $\mathcal{L}(\boldsymbol{b_s}, \boldsymbol{\varphi_s}; \boldsymbol{\lambda_s})$ in equation 16 equals to find the optimized state-specific hyper-parameters given $\boldsymbol{\lambda_s}$. When the $\boldsymbol{b_s}, \boldsymbol{\varphi_s}, \boldsymbol{\lambda_s}$ are determined, the interpolation for a certain state ***s*** is completed.

2.4 Gradient matching

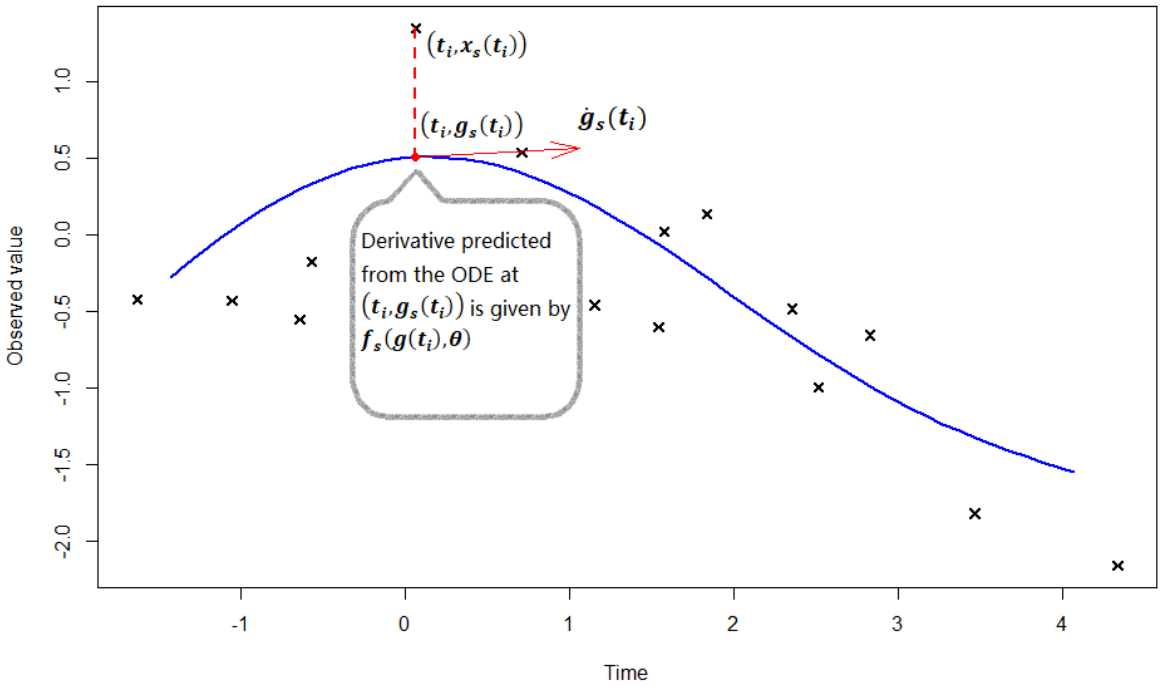


Figure 1- A sketch map to illustrate the idea of gradient matching. The cross signs represent observations for a certain state ***s***. The blue line represents the interpolant $g_s(t)$ for a certain state ***s***, its time derivatives $\dot{g}_s(t)$ is the slope of the tangent at $(t_i, g_s(t_i))$), also known as "interpolant gradient". The derivatives predicted from the ODEs at the $(t_i, g_s(t_i))$ is given by $f_s(g(t_i), \theta)$, also known as "gradient from ODE".

When an interpolant $g_s(t)$ is achieved, its time derivatives $\dot{g}_s(t)$ is readily available

$$\dot{g}_s(t; \boldsymbol{b_s}) = \sum_{i=1}^{n} b_{si} \frac{\partial K_s(t, t_i)}{\partial t} \tag{20}$$

The gradient from ODE is evaluated by the ODE function $f_s(g(t_i), \theta)$ defined in

equation 2. Finally, $\boldsymbol{\theta}$ is estimated to be the one that gives the least sum squares of the difference between $\dot{g}_s(t)$ and $f_s(g(t_i), \theta)$ across all states, that is,

$$\hat{\theta} = argmin_\theta \left\{ \sum_{s=1}^{r} \sum_{i=1}^{n} \left(\dot{g}_s(t_i) - f_s(g(t_i), \theta) \right) \right\} \tag{21}$$

Where $\dot{g}_s(t_i) = \sum_{j=1}^{n} b_{sj} \dot{K}_s(t_i, t_j)$ is readily available, and $\dot{x}_s(t_i)$ can be derived from the ODEs.

The idea of gradient matching itself is straightforward. It presumes the interpolant for each state is close to the true function implied by the ODEs, thus have gradients close to those predicted from ODEs at the observed time points. However, the interpolant's quality as one can expect determines the quality of following gradient matching, which requires us to ensure the goodness of fit of the kernel ridge regression.

2.5 Kernel available in KGode package

There are many positive definite kernels with reproducing properties, two popular ones are applied in this work. They are radial basis function (RBF) and multilayer perceptron (MLP).

In the context of a dynamical system, each state has its kernels with state-specific hyper-parameter, taking input time-point **t** and a given time-point $\mathbf{t_i}$ of the observations in data. In the KGode package, the RBF kernel for a certain state **s** is defined as,

$$\forall\, \mathrm{t_i} \in \mathrm{T},\ \mathrm{K_s(t, t_i)} = \ \mathrm{e}^{-\frac{(\mathrm{t}-\mathrm{t_i})^2}{{\varphi_s}^2}} \tag{22}$$

where φ_s is a state-specific hyperparameter greater than zero, also known as length scale of RBF.

The MLP kernel is defined as,

$$\forall\, \mathrm{t_i} \in \mathrm{T}, \mathrm{K_s(t, t_i)} = \arcsin\left(\frac{\omega_s \mathrm{t_i t} + \mathrm{l_s}}{\sqrt{\omega_s {\mathrm{t_i}}^2 + \mathrm{l_s} + 1}\sqrt{\omega_s \mathrm{t}^2 + \mathrm{l_s} + 1}} \right) \tag{23}$$

where ω_s, $\mathrm{l_s}$ are hyper-parameters.

3 Application to simulated data

3.1.1 Parameter inference for Lotka–Volterra model

Lotka-Volterra model is famous for describing the dynamics of the biology system

where two species' populations (Preys and predators) interact. Put in the framework of the dynamical system, the population of the prey is regarded as state 1, denoted by $x_1(t)$ while the population of the predators is regarded as state 2 denoted by $x_2(t)$. The model can be summarized by two ODEs:

$$\dot{x}_1(t) = \alpha x_1(t) - \beta x_1(t)x_2(t)$$
$$\dot{x}_2(t) = \delta x_1(t)x_2(t) - \gamma x_2(t) \quad (24)$$

Where $\dot{x}_1(t)$, $\dot{x}_2(t)$ are time derivatives for the two states in the model, $\alpha, \beta, \delta, \gamma$ are parameters in the model.

In the following section, kernel ridge regression with both MLP and RBF kernels are fitted to compare their performance in capturing the patterns and accuracy in the following parameter inference. For the sake of testing performance, data simulated from a known parameter set is used and is added to independent Gaussian noise ($noise \overset{i.i.d}{\sim} N(0,0.2)$) which result from the random measuring error from data collection and unconsidered factor in the model.

In Figure 2 and Figure 3, observations distribute evenly around the interpolants. Despite some discrepancies between the interpolants and the true functions caused by noise, the interpolation captures the general pattern well and makes the gradient matching practical.

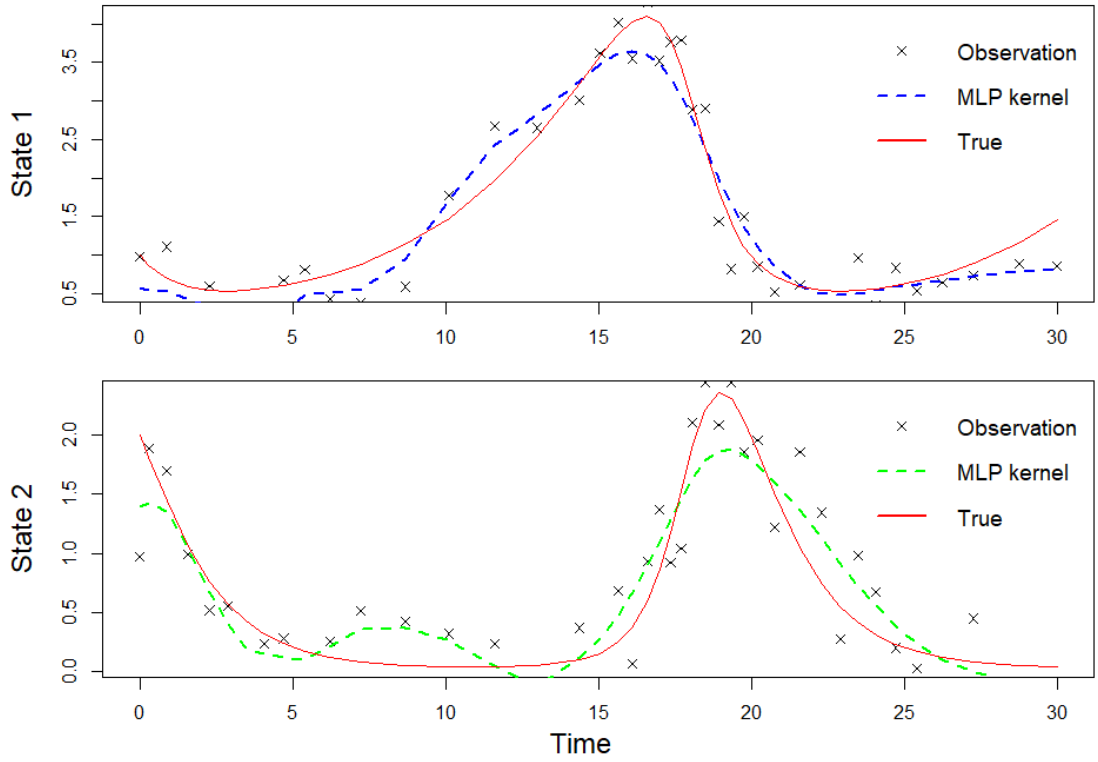


Figure 2-The interpolation of state 1, 2 based on MLP kernel. The observations are marked by symbol "×", the true functions of state 1 and state 2 are red lines and the dashed colored lines are the smooth interpolants based on MLP kernels

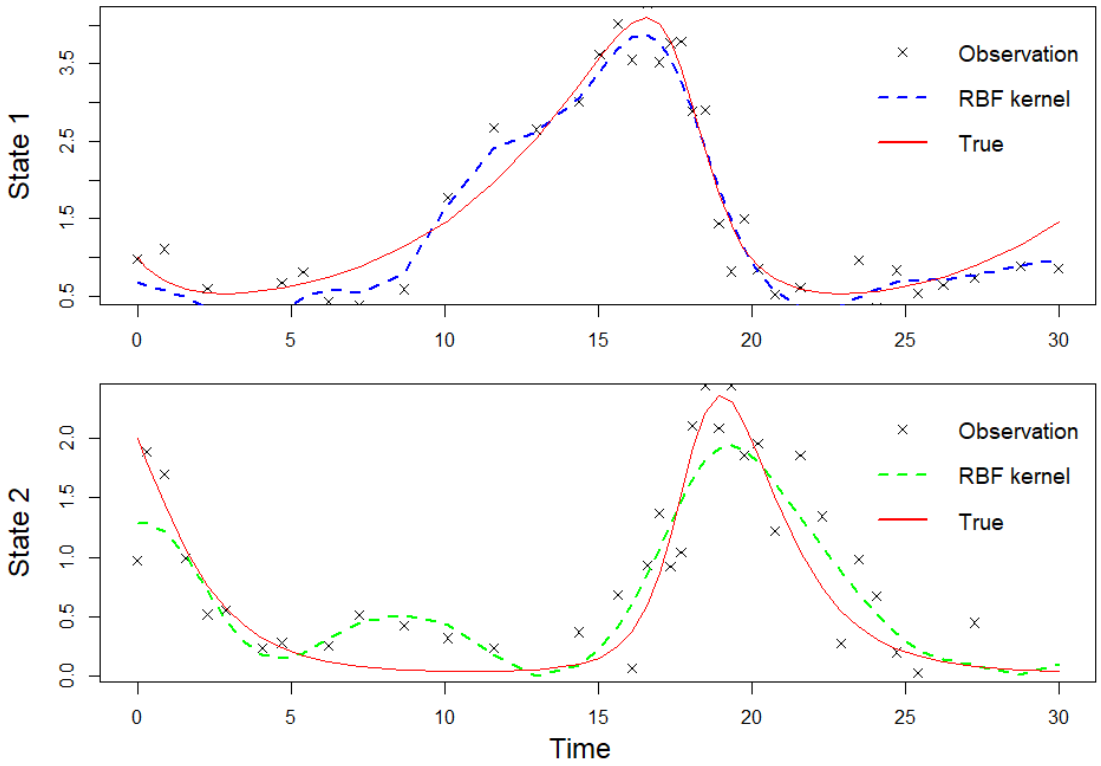


Figure 3-The interpolation of state 1,2 based on RBF kernel. The observations are marked by symbol "×", the true functions of state 1 and state 2 are red lines and the dashed colored lines are the smooth interpolants based on RBF kernels

Because under the assumption of normal error with zero mean and constant variance, a model with good fitness will have residuals approximately following a normal distribution and distributing evenly around zero across the fitted values, we can use QQ-plot and "residuals versus fits plot" to verify this assumption and conclude the kernel ridge regression is a good fit. The QQ plots plot the sample quantiles against the theoretical quantile, while in the right "residuals versus fits plot", residuals are plot against fitted values. We can see in Figure 4, 5 that only the residuals of state 2 when using MLP kernel deviate from the QQ-line which indicates the regression the distribution of residual is not an ideal normal distribution. By contrast, Figures 6, 7 show that using RBF kernels gives a better fit as the points in the QQ plot form a straight line and in the residual vs interpolation plot, residuals distribute evenly around the zero across interpolations. Overall, We conclude the assumptions for kernel ridge regression that error term follows a normal distribution is not strongly violated and the RBF kernels are more suitable in this case.

After the regression method manages to interpolate the time series of state 1 and state 2 in the Lotka–Volterra model, the time derivatives of the interpolants are calculated and compared with the gradients predicted from the ODEs of the model. The estimated parameters $\hat{\theta}$ are searched in parameter space to minimize the objective function in equation 21.

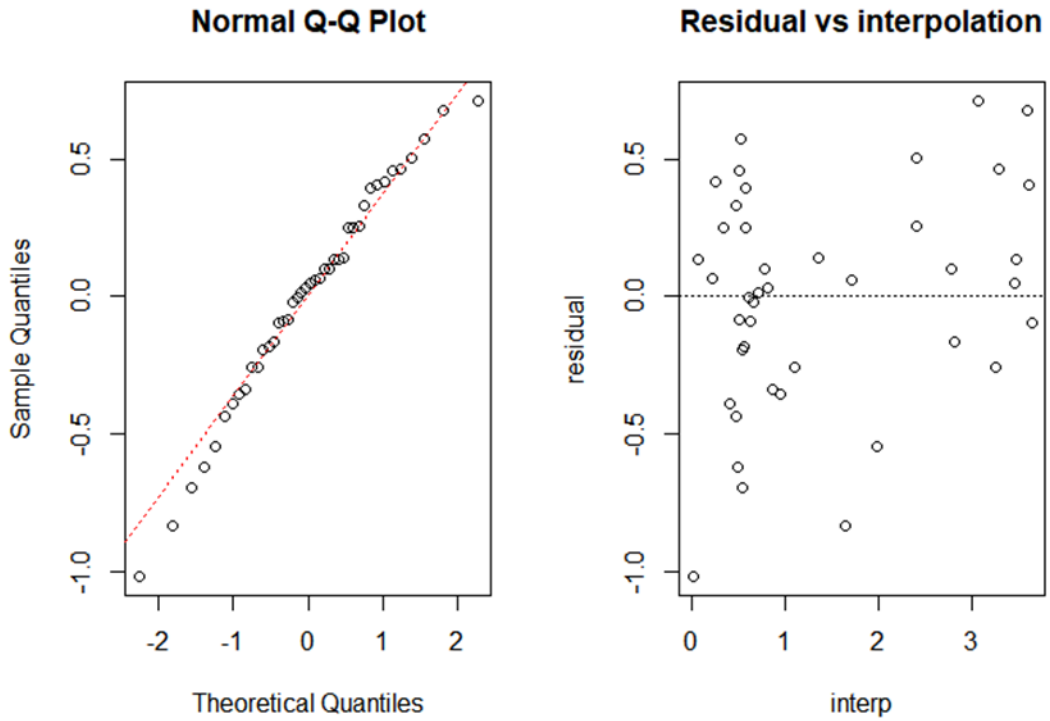


Figure 4-Diagnostic analysis on state 1 for the model using MLP kernel.

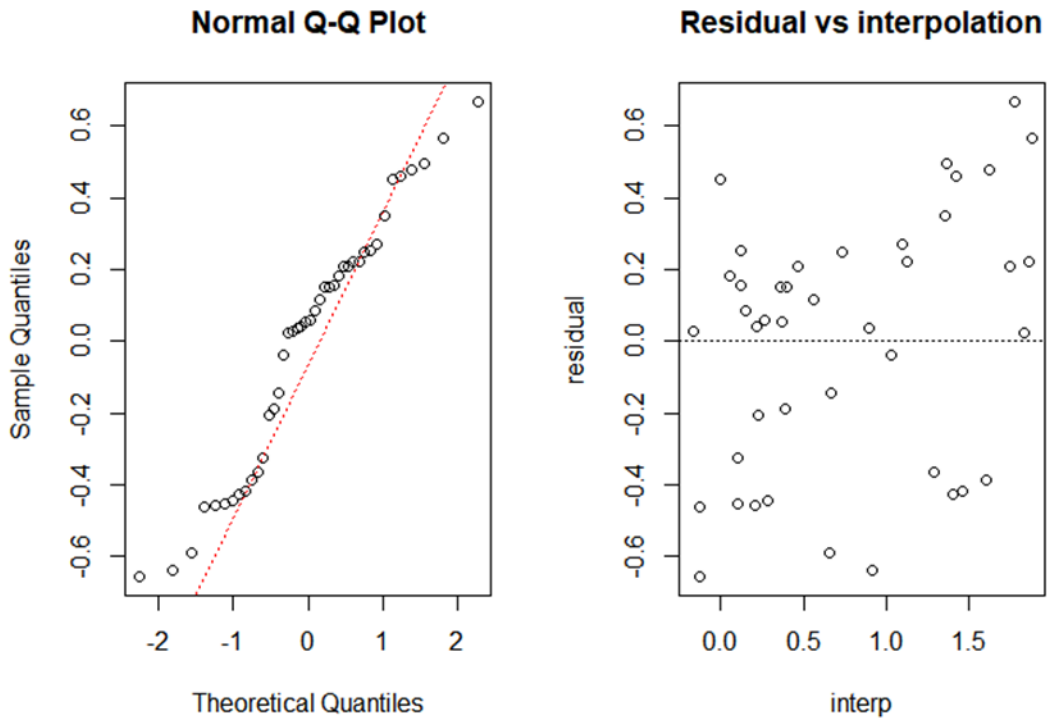


Figure 5- Diagnostic analysis on state 2 for the model using MLP kernel

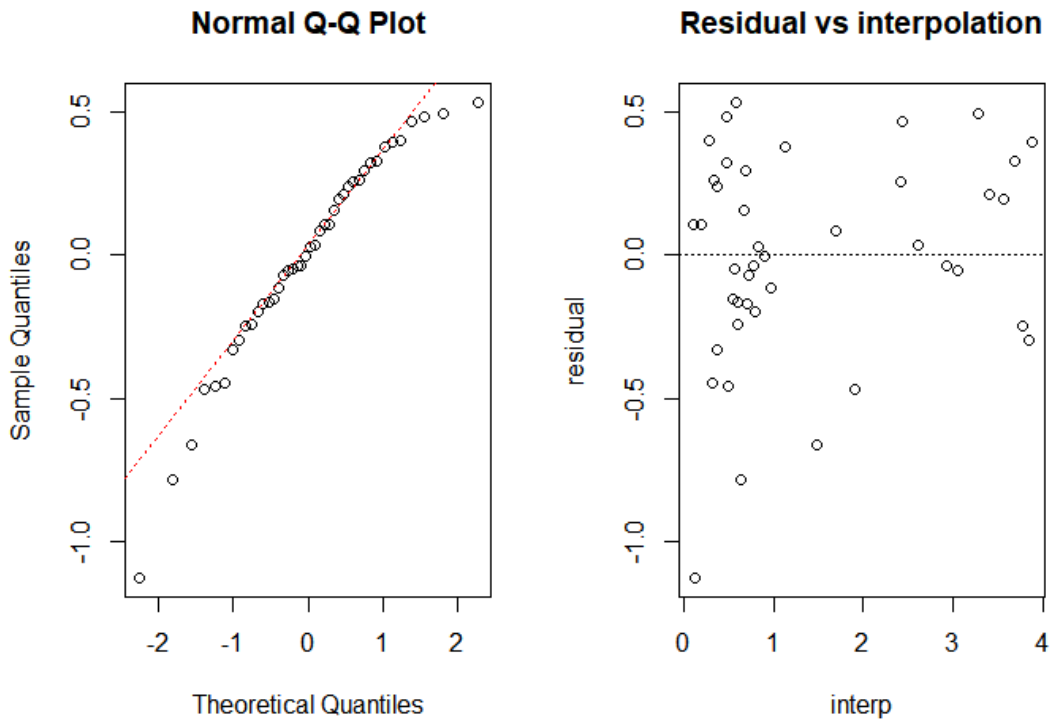


Figure 6- Diagnostic analysis on state 1 for the model using RBF kernel

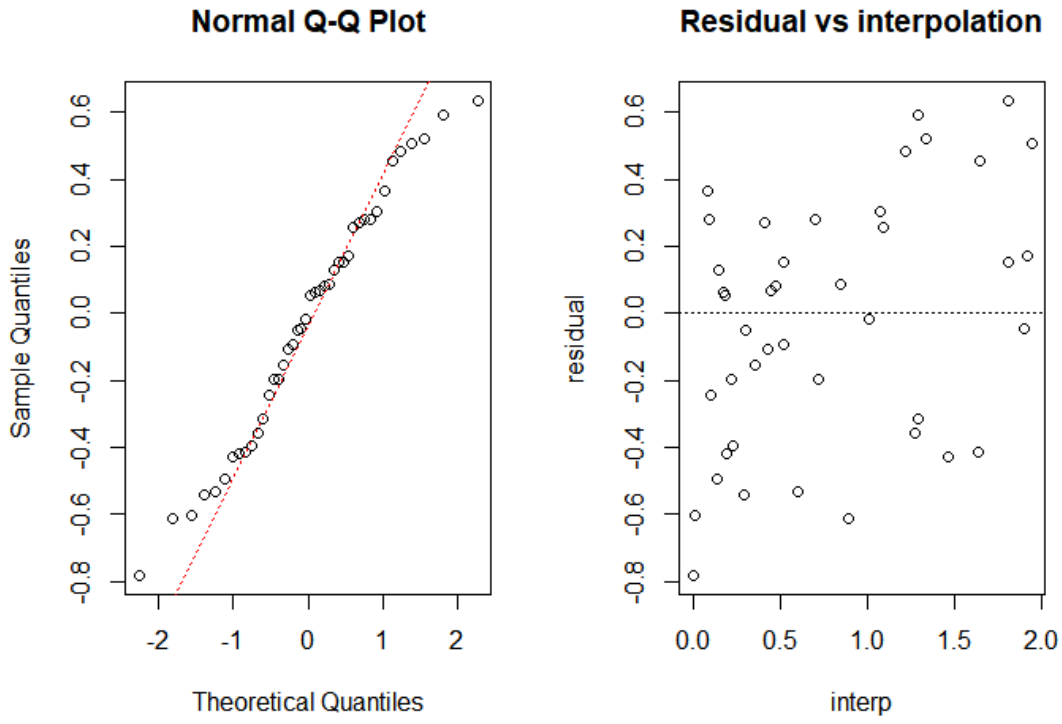


Figure 7- Diagnostic analysis on state 2 for the model using RBF kernel

3.1.2 Quantification of the uncertainty in parameter inference

It is worth noting that the parameter inference using kernel ridge regression can be prone to noise in the observed data and leads to an unsatisfying result. To quantify the effect of uncertainty caused by noise, the simulation is run 100 times which yields 100

sets of estimates of parameters. For each set of simulated data, the kernel ridge regression is fitted using both MLP and RBF kernels which gives 100 parameter estimations respectively. To assess the distributions of these estimations, they are pooled together in a dot plot, and the red point is the true parameters we set in the Lotka–Volterra model.

It is can be seen in both the MLP's and RBF's panel that the bulk of estimates for the four parameters cover the true parameter which indicates the gradient matching based on RKHS interpolation is effective. Moreover, the widths of the piled dots are proportional to the probability density of the estimated parameters at a certain value, so the results that the red points (true parameter values) lie in the high-density area are more favorable.

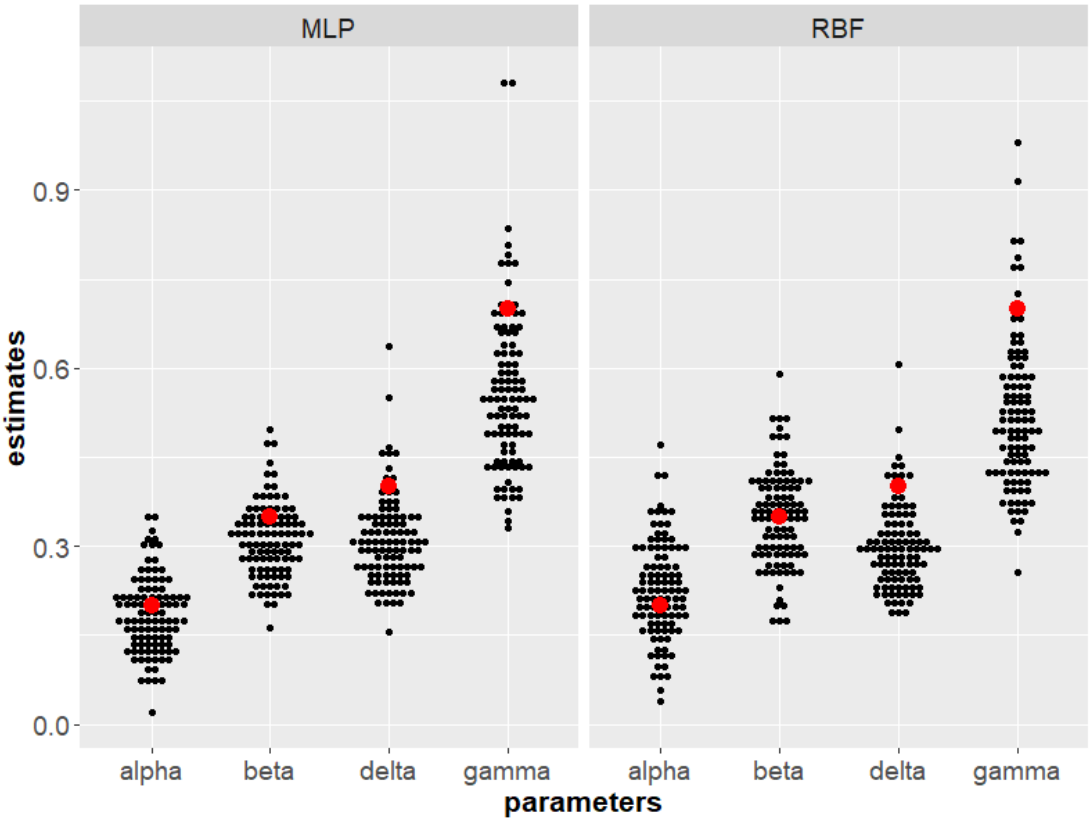
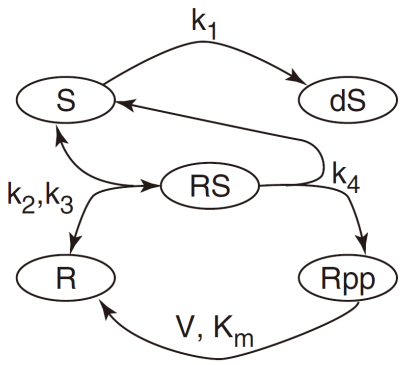


Figure 8 – Dot plot of parameter estimates for the Lotka–Volterra model for 100 different random initializations. The true values for the parameters are alpha=0.2, beta=0.35, gamma=0.7, delta=0.4, marked by red dots. The black dots are stacked, with each dot representing one observation.

3.2.1 Parameter inference for Bio-pathway model

In the above example, MLP and RBF kernels perform similarly in regression and later parameter estimation. However, in the following example, where the two kernels see a bigger difference. The biological model to be discussed has previously been studied by Vyshemirsky and Girolami (2008) as the Model 1 in the literature, which describes a signal transduction cascade including 5 protein- S, dS, R, RS, Rpp. As the biological

essence of this model is beyond the remit of this work, a detailed description of the model is quoted from preceding literature (Vyshemirsky and Girolami (2008)):



"Model 1: This model defines a common motif of signalling pathways that is a stage in a signal transduction cascade, for example this motif is repeated several times in Schoeberl et al. (2002). The input signal is represented by the concentration of protein S depicted in the top left of the diagram (Fig. 1a). This protein activates the next stage of the cascade by binding to protein R forming complex RS, and activating R into its phosphorylated form Rpp. Protein Rpp can then be deactivated. Model 1 also defines input signal degradation by converting protein S into its degraded form dS.
All the proteins used in this model will be represented as dependent variables in our ODE model. As we are interested in modelling and analysis of temporal behaviour, the independent variable is time. The dephosphorylation reaction Rpp⟶ R is defined using the Michaelis–Menten kinetic law, while the rest of the reactions (arrows in Fig. 1a) are defined using the Mass Action kinetic law with parameters depicted as textual remarks beside the arrows in the model diagram (e.g. k1, k4). The system of ODEs which defines this model can be found in the Supplementary Material. This model has six kinetic parameters: k1.. . k4, V, Km."

The model can be summarised by a series of ODEs:

$$\begin{aligned}
[\dot{S}] &= -k_1 \cdot [S] - k_2 \cdot [S] \cdot [R] + k_3 \cdot [RS] \\
[\dot{dS}] &= k_1 \cdot [S] \\
[\dot{R}] &= -k_2 \cdot [S] \cdot [R] + k_3 \cdot [RS] + \frac{k_5 \cdot [Rpp]}{k_6 + [Rpp]} \\
[\dot{RS}] &= k_2 \cdot [S] \cdot [R] - k_3 \cdot [RS] - k_4 \cdot [RS] \\
[\dot{Rpp}] &= k_4 \cdot [RS] - \frac{k_5 \cdot [Rpp]}{k_6 + [Rpp]}
\end{aligned} \tag{25}$$

The square brackets, [·], denote concentrations of 5 protein isoforms (the stat es in the dynamical system), and k_1, k_2, k_3, k_4 represent the 4 Mass Action ki netic parameters to be inferred, and k_5, k_6 correspond with *V, Km,* the Micha elis-Menten kinetic parameters. It turns out that k5 and k6 are only weakly i dentifiable, and we have thus assessed the accuracy of inference based

on the ratio $\frac{k_5}{k_6}$.

Data are simulated by adding independent identically distributed normal noise to the numerical integration of the ODEs on the interval $[0,100]$, where the authentic parameters to be detected are pre-defined: $k_1 = 0.07, k_2 = 0.6, k_3 = 0.05, k_4 = 0.3, k_5 = 0.017, k_6 = 0.3$. The noises are set to have variances 0.09 times as big as the sample variance of each state and mean zero (noise could be directly proportional to the scale of state values). Figure 9 and 10 show the results of kernel ridge regression using RBF kernel and MLP kernel respectively, where the colored dashed lines are the interpolants of the 5 states in Bio-pathway model and red lines are the true function of them. We can see that the RBF's interpolants are quite wiggly. The reason behind this is the observations gather in the interval [0,20] where the state values change rapidly, and lead to a hyperparameter that only captures a local pattern instead of the whole pattern.

By contrast, MLP kernel can capture the global pattern, thus in the further assessment, we might expect methods with MLP may give better parameter estimation because the quality of parameter estimation of gradient matching highly relies on the qualities of interpolants.

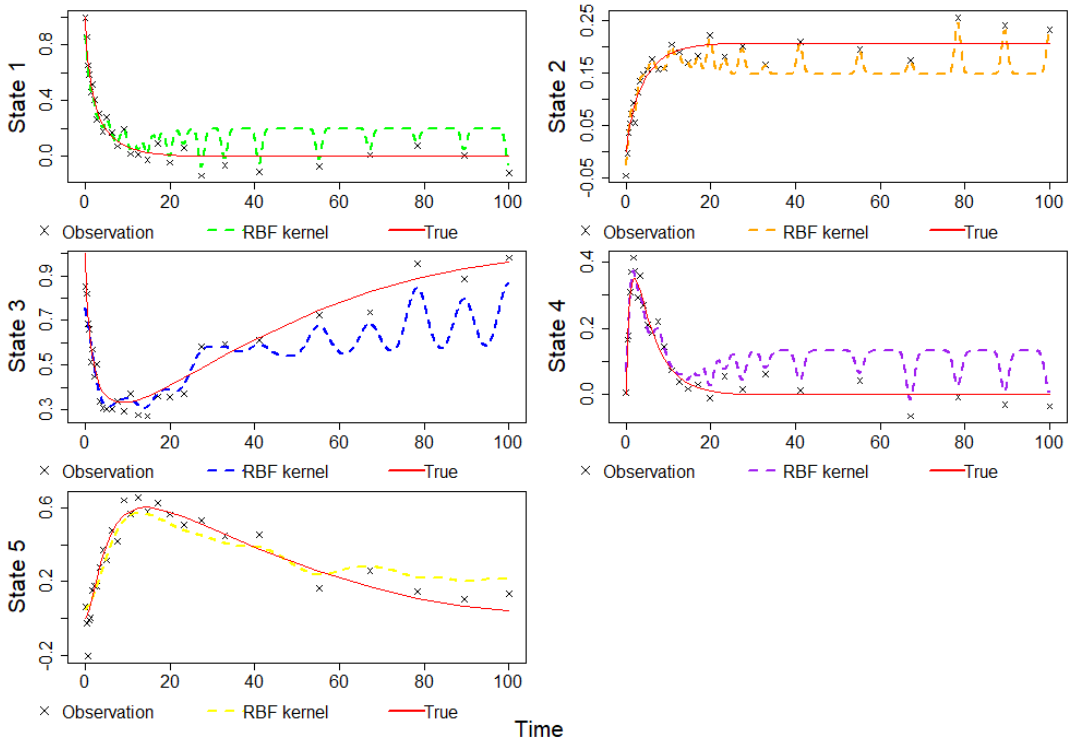


Figure 9-Interpolations for 5 states using RBF kernel

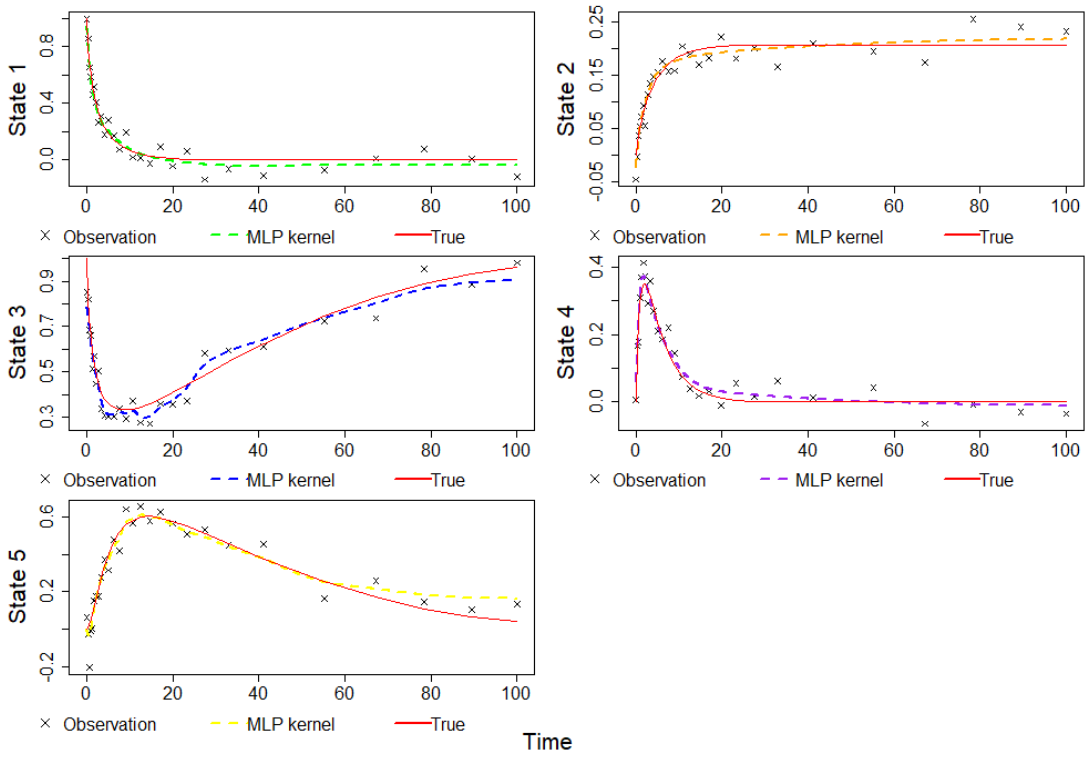


Figure 10 - Interpolations of 5 states using MLP kernel

3.2.2 Quantification of the uncertainty in parameter inference

As stated in section 3.1.2, the same plots are used to compare the result of MLP and RBF. In the 100 estimations, each simulation is performed with the same setting (noise follows identical independent normal distributions with mean zero, variances 0.09 times as big as the sample variance of each state).

In Figure 11, the red points denote the true parameter values while each dot represents the result of one estimation. Because parameters k5 and k6 are weakly identifiable in this model, we assess the uncertainty by the ratio of these two parameters. The dots in the RBF panel gather below the true parameter points while the dots in the MLP panel center around the true parameter points, so the RBF kernel is more likely to give inaccurate estimations.

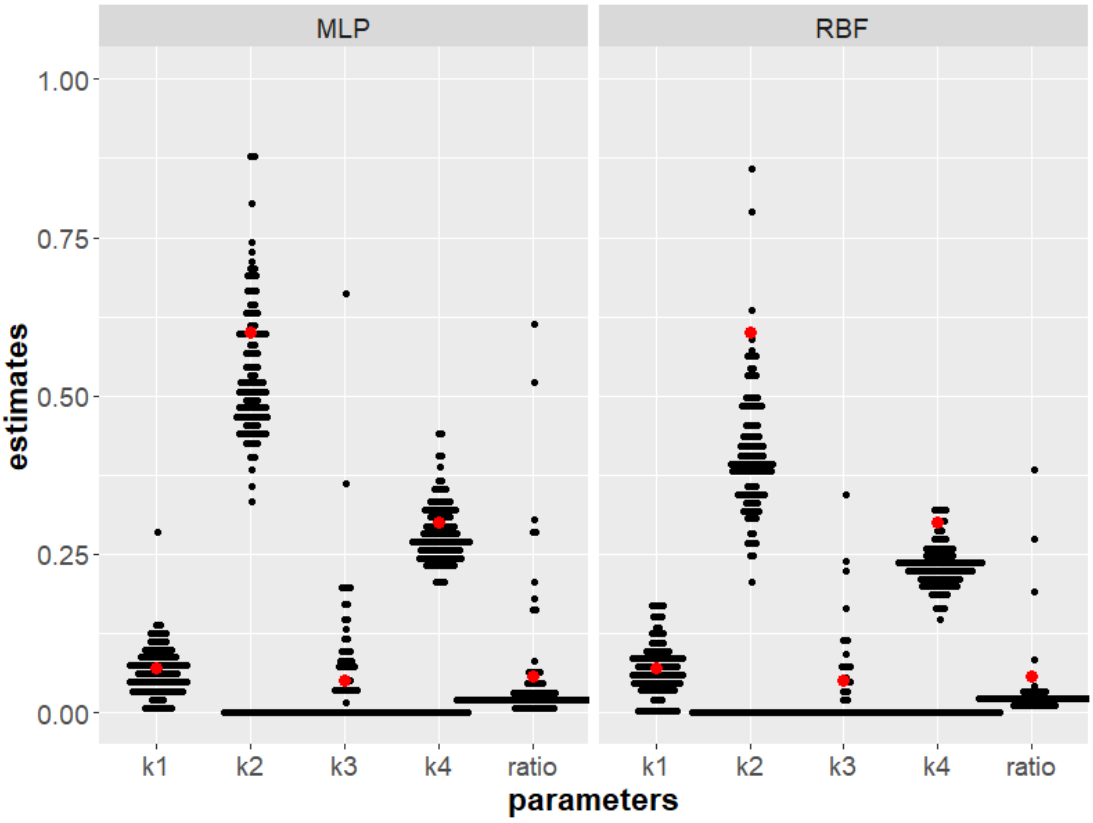


Figure 11- Dot plot of parameter estimates for the Bio-pathway model for 100 different random initializations. The true values for the parameters are $k_1 = 0.07, k_2 = 0.6, k_3 = 0.05, k_4 = 0.3, k_5 = 0.017, k_6 = 0.3, ratio = k_5/k_6 = 0.057$, marked by red dots. The black dots are stacked, with each dot representing one observation.

4 ODE regularization

In addition to the standard two-stage gradient matching method described above, there is an alternative way to do interpolation and gradient matching (Niu et al (2016)). The proposed method is effectively based on the standard kernel ridge regression wherein the interpolation stage, the hyperparameters, and coefficients of the kernels are determined for each state independently, and in the gradient matching stage the estimated parameters $\hat{\theta}$ is yielded by minimizing the difference between the derivatives of interpolants and predicted derivatives from the ODES. However, the standard kernel ridge regression is conducted purely based on observed data without considering the dynamics of the ODE system. To improve the quality of the interpolants, the proposed method uses the ODEs themselves as a regulariser which consists of the following two steps:

Step -1 Initialization of regression coefficients and ODE parameters using standard gradient matching.

In step 1, the interpolants $g_s(t)$ for each state is obtained and the ODE parameters $\widehat{\boldsymbol{\theta}}_{\mathbf{0}}$ are estimated for initialization. The vertical n-element vector $\boldsymbol{b_s}$ contains n

kernel regression coefficients of each state $\boldsymbol{s}$ (s=1,2,…r), and forms a $n \times r$ matrix $\boldsymbol{B} = (\boldsymbol{b_1}, \ldots, \boldsymbol{b_s})$.

Step -2 Minimization of the combined objective function.
The objective function is designed to be

$$h(\theta, B) = \sum_{s=1}^{r}\sum_{i=1}^{n}(g_s(t_i; \boldsymbol{B}) - y_s(t_i))^2$$
$$+\lambda \sum_{s=1}^{r}\sum_{i=1}^{n}\left(\dot{g}_s(t_i; \boldsymbol{B}) - f_s(g(t_i, \boldsymbol{B}), \boldsymbol{\theta})\right)^2 \quad (26)$$

KGode package in R provides two approaches for optimizing $h(\boldsymbol{\theta}, \boldsymbol{B})$. The first approach optimizes the objective function by searching θ, B simultaneously using a standard optimization routine, such as conjugate gradients or quasi-Newton. This approach can be computationally expensive because of the large number of parameters. The second approach (Niu et al (2016)) was proposed which updates $\boldsymbol{\theta}, \boldsymbol{B}$ separately and iteratively. When fixing both $\boldsymbol{\theta}$ and $\boldsymbol{B}$ in the argument of f_s, the objective function $h(\boldsymbol{\theta}, \boldsymbol{B})$ becomes convex in the remaining parameters $\boldsymbol{B}$ because of the linear structure in the representer theorem. When fixing $\boldsymbol{B}$ in the whole equation, the estimation of $\boldsymbol{\theta}$ becomes also easier. The modified optimization algorithm modifies the objective function $h(\theta, B)$ by treating the coefficient matrix $\boldsymbol{B}$ in f_s as constant in optimization:

$$\tilde{h}(\boldsymbol{\theta}, \boldsymbol{B_1}, \boldsymbol{B_2}) = \sum_{s=1}^{r}\sum_{i=1}^{n}\left(g_s(t_i; \boldsymbol{B_1}) - y_s(t_i)\right)^2$$
$$+\lambda \sum_{s=1}^{r}\sum_{i=1}^{n}\left(\dot{g}_s(t_i; \boldsymbol{B_1}) - f_s(g(t_i, \boldsymbol{B_2}), \boldsymbol{\theta})\right)^2 \quad (27)$$

We can now iterate the following steps to optimize $\boldsymbol{B}$ and $\boldsymbol{\theta}$.

1. Given $\boldsymbol{\theta} = \widehat{\boldsymbol{\theta}}_{\boldsymbol{k}}$, $\boldsymbol{B_2} = \widehat{\boldsymbol{B}}_{\boldsymbol{k}}$ (k>=0), minimize $\tilde{h}(\boldsymbol{\theta}, \boldsymbol{B_1}, \boldsymbol{B_2})$ with respect to $\boldsymbol{B_1}$, i.e. find $\widehat{\boldsymbol{B}}_{\boldsymbol{k+1}} = argmin_{\boldsymbol{B1}}\tilde{h}(\boldsymbol{\theta}, \boldsymbol{B_1}, \boldsymbol{B_2})$. (The initial values $\widehat{\boldsymbol{\theta}}_{\boldsymbol{0}}$, $\widehat{\boldsymbol{B}}_{\boldsymbol{0}}$ are inherited from Step 1)
2. Set $\boldsymbol{B_1} = \widehat{\boldsymbol{B}}_{\boldsymbol{k+1}}, \boldsymbol{B_2} = \widehat{\boldsymbol{B}}_{\boldsymbol{k+1}}$ and minimize $\tilde{h}(\boldsymbol{\theta}, \boldsymbol{B_1}, \boldsymbol{B_2})$ with respect to $\boldsymbol{\theta}$ i.e. find $\widehat{\boldsymbol{\theta}}_{\boldsymbol{k+1}} = argmin_{\boldsymbol{\theta}}\tilde{h}(\boldsymbol{\theta}, \boldsymbol{B_1}, \boldsymbol{B_2})$

The ODE regularization (RKG3) method is implemented in KGode in the third(.) function. It will be applied to both the Lotka-Volterra model and the Bio-pathway model to test its performance compared to the standard gradient matching using kernel ridge regression.

4.1 Lotka-Volterra model

In the figure below, the result of the ODE regularization method is added to the previous 1plot whose interpolants are presented in dotted purple lines. In figure 12,

the dashed lines represent those of standard gradient matching based on RBF kernels while the purple dotted lines are the ODE regularized result based on the standard gradient matching. It can be seen that the ODE regularized interpolants are closer to the true function and likely to give better parameter estimation.

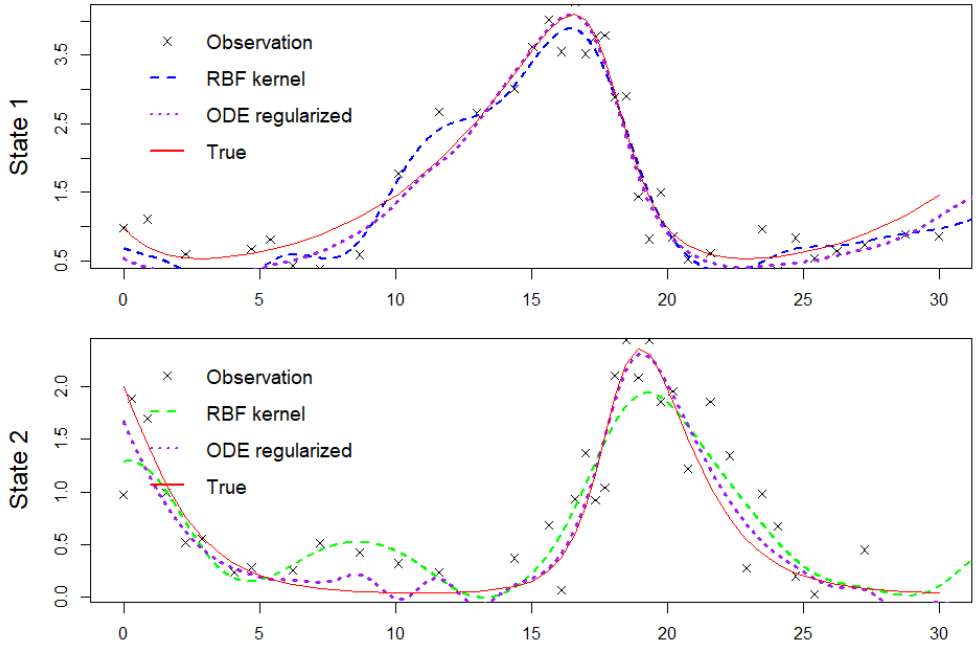


Figure 12 –The interpolation of state 1,2 in the Lotka-Volterra model. Adding the result of the ODE regularized method to figure 2.

The dot plot is used to visualize the performance of the two methods. The estimates of the ODE regularized method (also using RBF kernel) denoted by dots gather around the red points more closely than those in the RBF panel (standard gradient matching). Thus we can conclude ODE regularized method does see an improvement in the standard gradient matching based on RBF kernels.

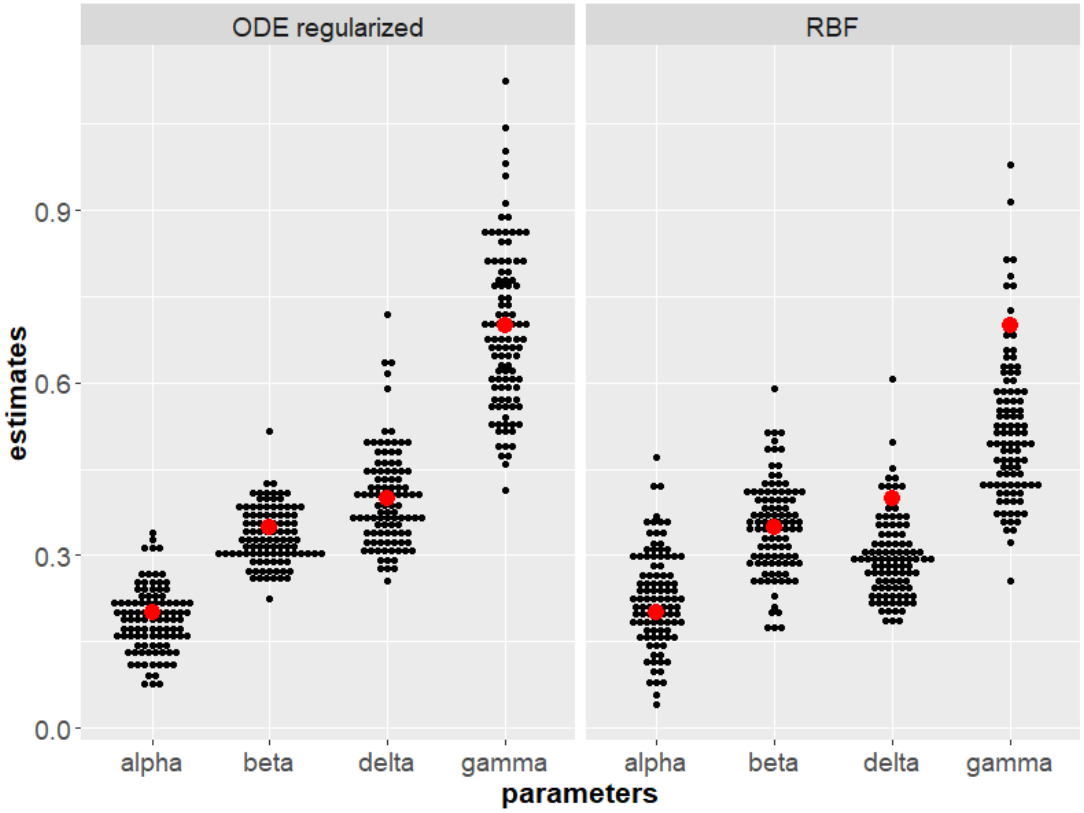


Figure 13- Dot plot of parameter estimates for the Lotka–Volterra model for 100 different random initializations. The true values for the parameters are alpha=0.2, beta=0.35, gamma=0.7, delta=0.4, marked by red dots. The black dots are stacked, with each dot representing one observation. The left panel shows the results of the ODE regularized (using the fast iterative scheme) method while the right panel shows the standard gradient matching using RBF kernels.

5 Comparison with Differential Evolution method

5.1 Procedure of Differential Evolution method

On the premise that our modeling of a dynamical system is sensible, given a set of Parameters close to the true parameters, the ODEs defining the model is supposed to predict state values very close to the observations despite some noise in the data. So if a loss function is defined that measures the difference between the predicted state values and the observed state values, then the parameter estimation of the ODEs becomes an optimization problem.

Differential Evolution (David et al (2018)) is an evolutionary algorithm to conduct a heuristic search in parameter space and is much more efficient than an exhaustive search. This algorithm is chosen also because it is not easily trapped in the local minima of a function. As it imitates a natural biological process, it has many terms borrowed from Biology:

1) Population - a set of parameter vectors that is transformed at each generation,

into another set of parameter vectors, the members of which are more likely to minimize the objective function.

2) Mutation - Mutation produce changes based on the population and thus generate new members that may replace the original member
3) Crossover - Controls the fraction of the mutant parameter values to be passed to trial vectors of parameters.
4) Bound check -If any element of a vector of parameters violates the bounds after mutation and crossover, it is reset, where here and throughout we use j to index into a parameter vector.
5) Selection - If a trial vector $v_{i,g}$ has lower or equal objective function value, them it replaces its parent vector $x_{i,g}$ to enter the next generation as a member of the new population, otherwise $x_{i,g}$ reserves its place. (index *i* denotes the *i-th* vector of parameters, *g* indicates the *g-th* generation of the population.)
6) Minimizing objective function – In the context of our work, each vector of parameters will be input into the dynamical system to solve ODEs explicitly. The objective function measures the discrepancy between the solved state values and the observed state values with the least sum of squared residuals.

Parameter estimation is made for the Lotka–Volterra model, using the same simulated data in the previous section. This method can make relatively accurate parameter estimations given enough generations. Figure 14 shows the state values solved from ODEs with estimated parameters, and figure 15 shows the process of convergence.

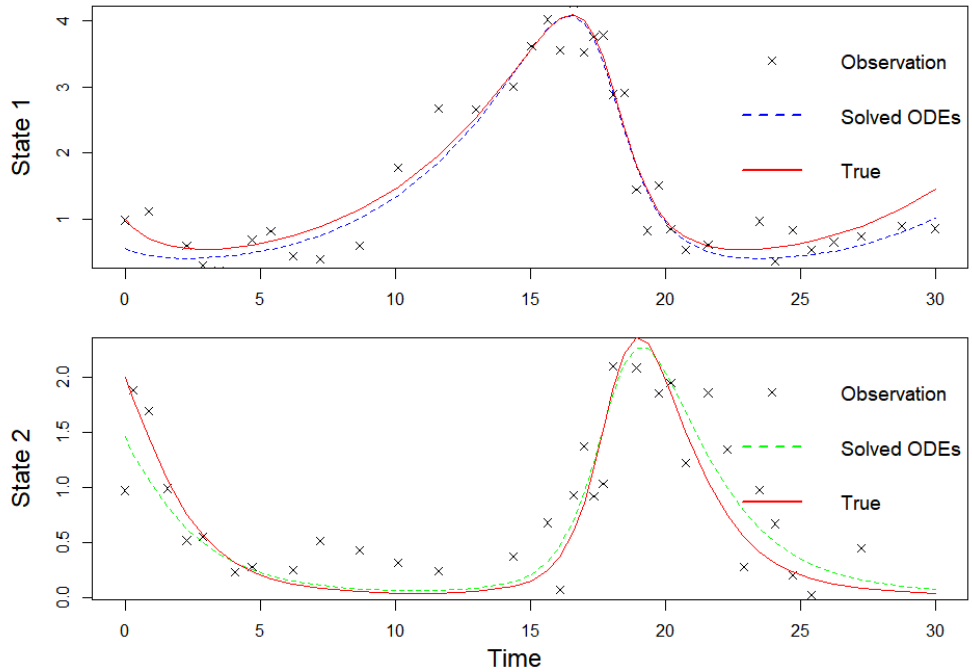


Figure 14-Solved ODEs with parameters from differential evolution, 2000 generations of parameters are made in the algorithm. The initial state values in the dynamical system are regarded as parameters to be searched, with a range between 0.5 times observed values to 1.5 times observed values

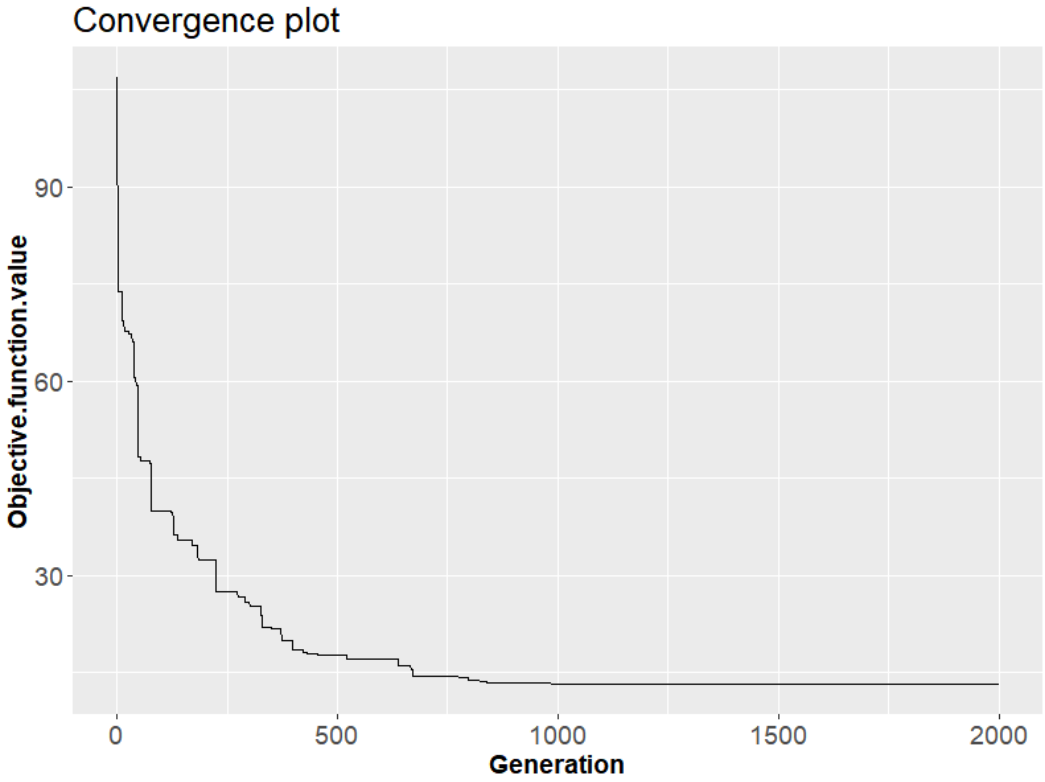


Figure 15- The convergence plot shows the process of convergence of objective function value. The convergence seems to be reached after 1000 generations of parameters and this process takes a relatively long time.

5.2 Method comparison

A parameter estimation using differential evolution can cost a much longer time compared to the gradient matching method. Typically, the former spends around 20 minutes searching in parameter space while it takes less than 3 minutes via the latter method. So uncertainty quantification we conducted by doing the estimation 100 times with different initialization of noise will be quite unfeasible for the iterative method. Alternatively, we can quantify the uncertainty of a method by repeating the parameter estimation on *K* sets of data created by residual bootstrapping (Efron and Tibshirani 1994) and summarizing the uncertainty with adjusted median absolute values. In this work, *K*=20 is used.

In more details, let $y_s = (y_s(t_1) \dots, y_s(t_n))$ denote the observed noisy data for state ***s***, and $\hat{x}_s = (\hat{x}_s(t_1) \dots, \hat{x}_s(t_n))$ denote the predicted values which in the case of differential evolution method, are the values of states solved from the ODEs, while in the case of kernel regression, is the predicted values of the regressions. Then the residuals of state ***s*** are calculated by $y_s - \hat{x}_s$, from which *K* bootstrap residual sets $\tilde{\varepsilon}_{sk}, k \in \{1, 2, \dots K\}$ are sampled with replacements. Adding a bootstrap residual set $\tilde{\varepsilon}_{sk}$ to the predicted values $\hat{x}_s$, a bootstrap training set $\tilde{y}_{sk}$ is created. On the separate bootstrap training set $\tilde{y}_{sk}$, $k \in \{1, 2, \dots K\}$, parameter estimation is

repeated *K* times as on the original data y_s, producing *K* sets of estimated parameters $\tilde{\theta}_1, \tilde{\theta}_2 \ldots \tilde{\theta}_K$.
Finally, the uncertainty of parameter estimation by a method can be quantified by median absolute deviation (MAD), which is a robust measure of the variability of a univariate sample of quantitative data. For the first ODE parameter α in Lotka–Volterra model, adjusted MAD is given by

$$1.4826 \times Median(|\tilde{\theta}_k - Median(\tilde{\theta})|)$$

Where $\tilde{\theta}_k$ is the *k*-th estimate of parameters, $k \in \{1, 2, \ldots K\}$. The MAD is adjusted by multiplying a constant scale factor of 1.4826 so that it estimates the standard deviation of normally distributed data. In R, *mad( )* calculates the above by default. Even the data is not normally distributed, it is also a good way to measure uncertainty.

The performances of three methods are compared in the case of the Lotka-Volterra model, including standard gradient matching based on RBF-kernel, the standard approach with ODE regularization, and the differential evolution method. The mean values of *K*=20 sets of estimated parameters and the adjusted MADs are summarized in the table below,

Method	Type	Alpha	Beta	Delta	Gamma
	True value	0.2	0.35	0.7	0.4
RBF kernel	Mean values	0.2842	0.4275	0.3423	0.2221
Differential Evolution		0.2364	0.3975	0.5321	0.3434
RBF kernels with ODE regularization		0.2327	0.3726	0.5655	0.3405
RBF kernel	Adjusted MADs	0.0584	0.0699	0.0850	0.0470
Differential Evolution		0.0219	0.0365	0.0626	0.0321
RBF kernels with ODE regularization		0.0536	0.0439	0.07838	0.0539

Compare the mean values and adjusted MADs of parameters of the standard RBF kernel method and the method with ODE regularization, a significant improvement can be seen that the latter method has the mean values of the estimates closer to the true parameters, also with smaller MAD.

As for the differential evolution method, it achieves a comparably good estimation

compared to the RBF kernel method with ODE regularization and has an even smaller MAD which means it is more stable. However, it comes with a higher computational cost as it solves ODEs thousands of times and also searches initial values around the observations at time 0, which is equivalent to estimate two more parameters. When it is applied to a dynamical system with more states and noisier observations, it can give poor estimation. Therefore, the result of the evolutionary algorithm should be supervised.

An advantage of the evolutionary method is it can estimate a dynamical system with unobserved states. For example, in the work of C. A. K. Kwuimy et al 2020, they established a dynamical system to explain and predict the increase of COVID cases, and in the model, the numbers of exposed individuals are not available, which means the gradient matching method is not applicable but the differential evolution method is applicable because it only needs to take initial values and then minimize the discrepancy between the simulated data and observed data. In their work, they use the genetic algorithm (GA) for parameter identification which is another evolutionary algorithm, more details can be seen in their thesis.

The time cost of a single run for the three methods is also recorded because this project's main purpose is to investigate a faster method with a guarantee of the accuracy of estimation. Executed on an i7 7700HQ CPU, the RBF kernels with ODE regularization are about 15 times faster than the standard iterative method and have comparable accuracy.

Method	Time cost (seconds)
RBF kernel	26
Differential Evolution	1213
RBF kernels with ODE regularization	76

6 Summary

Motivated by the expensive computational cost of a standard iterative method for parameter inference in non-linear dynamic system modeling, which usually involves solving ODEs repeatedly and searching parameters in large parameter space, this project mainly discusses an alternative faster scheme using gradient matching based on kernel ridge regression (named as RKG for brevity). The second objective is to understand a modified scheme proposed in Niu et al (2016), named as RKG3 for brevity. The method section summarizes the theoretical basis of the kernel ridge regression,

including what kind of kernels are used (reproducing kernels), why the regression takes such a form (Representer theorem). The application section reports the performance of the RKG method in two specific systems by data simulation, based on two kernels, RBF and MLP kernel respectively.

In addition, RKG3 which regularizes the interpolants by ODEs in the dynamical systems is introduced and compared to the standard iterative method in terms of accuracy of the estimates, the time cost of each method. And we can conclude the RKG3 method can have similar accuracy and much smaller computational cost compared to the standard iterative method, though with more uncertainty as shown by MAD metrics.

7 Future work

Despite the efficacy shown in the two simulated cases, this kernel-based gradient matching scheme can give misleading results if not applied to the appropriate situation. As one may notice, we have only worked on dynamical systems with relatively short duration, but in the real world, the estimation for a dynamical system can be persistent where the parameters might vary with time (e.g. modeling a finance model of six months' duration, consisting of the gold price, dollar, staple commodities). My interest would be how to extend the kernel-based gradient matching scheme to allow for time-varying parameters.

The existent literature has already proposed three approaches to allow for non-constant parameters. First, the parameters are assumed to vary across subsets of observations in the data in a deterministic way. Second, the parameters are assumed to be stochastic and be generated by a stationary stochastic process. Third, the parameters are stochastic and are generated by a nonstationary process. In the context of this work, the first approach (Orbe et al (2005)) may be compatible with a gradient matching scheme, which can be interesting to verify in future work.

8 Reference

[1] Mu Niu, Joe Wandy, Rónán Daly, Simon Rogers, Dirk Husmeier (2020) R package for statistical inference in dynamical systems using kernel based gradient matching: KGode

[2] Mu Niu, Simon Rogers, Maurizio Filippone, Dirk Husmeier (2016) Fast Inference in Nonlinear Dynamical Systems using Gradient Matching

[3] Benn Macdonald and Dirk Husmeier (2015) Gradient matching methods for computational for system biology: A review and comparative analysis

[4] Steven H. Strogatz, Addison-Wesley, Pajevic, S. (1995) Nonlinear dynamics and chaos. J Stat Phys 78, 1635–1636

[5] Liang H, Wu H (2008) Parameter estimation for differential equation models

using a framework of measurement error in regression models. J Am Stat Assoc 103(484):1570–1583
[6] Ramsay J, Hooker G, Campbell D, Cao J (2007) Parameter estimation for differential equations: a generalized smoothing approach. J R Stat Soc [1] B 69(5):741–796
[7] Michael M (1998) An introduction to genetic algorithm. The MIT Press
[8] Holland JH (1975). Adaptation in Natural Artificial Systems. University of Michigan Press
[9] Storn R, Price K (1997). "Differential Evolution A Simple and Ecient Heuristic for Global Optimization over Continuous Spaces." Journal of Global Optimization, 11(4), 341-359.
[10] Vyshemirsky V, Girolami MA (2008) Bayesian ranking of biochemical system models. Bioinformatics 24(6):833–839
[10] C. A. K. Kwuimy . Foad Nazari . Xun Jiao . Pejman Rohani . C. Nataraj (2020) Nonlinear dynamic analysis of an epidemiological model for COVID-19 including public behavior and government action
[11] Schöelkopf, B., and Smola, A.(2002). Support Vector Machines, Regularisation, Optimization and Beyond. Cambridge: MITPress.
[12] Mercer, J.(1909).Functions of positive and negative type, and their connection with the theory of integral equations. Philos.Trans.R.Soc.Lond.A 209, 415–446. doi:10.1098/rsta.1909.0016
[13] Murphy, K.(2012). Machine Learning: A Probabilistic Perspective. Cambridge: MIT Press.
[14] A. N. Tikhonov and V. Y. Arsenin. Solution of ill– Posed Problems. Winston, Washington, DC, 1977.
[15] David Ardia, Katharine M. Mullen, Joshua Ulrich, Brian G. Peterson (2018) DEoptim: An R Package for Global Optimization by Differential Evolution
[16] Susan Orbe, Eva Ferreira, Juan Rodriguez-poo (2005) Nonparametric estimation of time-varying parameters under shape restrictions

Reflection

The peer-review session has helped with doing my project. On the one hand, I got to learn from the beauties of others' work and to understand what makes a good report by matching their works against the given criteria and giving my suggestions. On the other hand, I got feedback on my initial materials which enabled me to notice the deficiencies I overlooked.

In detail, there are some suggestions I highly appreciate. For one, reviewer 1 suggested I include some plots to illustrate the ideas of this project since the whole section of rigorous definitions and mathematical notations may be a bit discouraging for readers who want to have a fast grasp of the basic ideas. So I took this piece of advice and include a sketch map in the method section and I believe it will make my report more readable. Also, reviewer 1 kindly pointed out the lack of references which I improved on in the latter work. Besides the shortcomings mentioned by reviewer 1, reviewer 2 would like to see a detailed description of the data to be used in my project, however, by the time I submitted my initial material, I haven't decided whether I should work on simulated data or real data. Thanks to his comment, I reflected on the objectives of my project and which would suits the objectives better. Thus I managed to justify why I chose simulated data over real data as seen in the report.

I also learned a lot when I was reading their initial materials and giving feedback. During the process, I realized how demanding it was for a reader to get an idea about new topics because of complex theorems and unfamiliar algorithms. To make an academic report accessible, I need to deliver it in a clear structure so that my readers can easily follow my flow of thoughts or read only the section of his/her interest. Also, I was enlightened by the use of coin toss examples in a peer's initial material, where detailed procedures and an informative diagram were presented. A similar style is used in my report as I believed it is a good way to convey ideas.

Overall, I find the peer-review session extremely helpful. It is a process where we learn from each others' mistakes and advantages, and then reflect on our work. I believe I have benefited a lot from it.