|epcc|

Bayesian Inference Using Sequential Monte-Carlo Algorithm for Dynamic System Models

Chaolin Han

August 10, 2020

MSc in High Performance Computing

The University of Edinburgh

Year of Presentation: 2020

Abstract

This is the bit where you summarise what is in your thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

While we are interested in the tissue regeneration among different species, mathematical models and techniques can help us to resolve complex interactions and effects of cells and chemicals in a biological system. The objective in this dissertation is an example of complex biology system found in tissue regeneration: zebrafish spinal cord regeneration. Zebrafish has the ability to regenerate its spinal cord after injury; at the lesion site various immune cells are presented and complex interactions are found while they are regulating inflammation and repair.

Our interest lies in the modeling of the regeneration process that happens around the lesion site, especially in the model parameter and comparison of different models. According to available researches [1], a mathematical model was firstly proposed but the addressed interactions remains to be precisely defined, and quantitatively examined in experiments. However, we would like to explore the the 'best fit' of the model which comes with a set of parameter estimates, given the observed data. Also, we wish to compare different it with other alternative models and possibly find the best model to describe the observed data. In these steps, models are expected to retain mathematical and biological significance, where differential equations are suitable to describe this system.

Fo the parameter estimation tasks, Bayesian inference techniques can help in the problems of over-fitting and local optima [2], compared to optimisation-based methods. Moreover, approximate inference methods (Approximate Bayesian Computation, ABC) make likelihood-free inference possible, which is suitable for our task as writing down likelihood as a function expression for the dynamic system can be hard.

Such methods introduce computation-intensive sampling-and-test patterns, where massive number of samples are drawn and followed by complex computations. As the sampling is independent from each other in a certain period, the parallel implementation of the algorithm becomes feasible and is expected to accelerate the parameter inference in a large scale when using HPC resources.

This dissertation includes the implementations and experiments of parameter estima-

tion using ABC SMC (sequential Monte-Carlo) method, and discussions on the results. ABC SMC was also applied in model comparison. Additionally, performance experiments illustrated the strong-scaling performance of the algorithm.

Chapter 2

Background

2.1 Zebrafish spinal cord regeneration

Compared to mammals, zebrafish can regenerate spinal cord after injury, and possibly recover swimming function. They are widely used in regenerative ability researches due to this ability. Experiments found that although being cut off, the axonal of nerve cells can regenerate and bridge across the lesion site, which critically determined whether zebrafish can regain the swimming function [3]. Tsarouchas et al. [1] studied the the complex interactions happen at the lesion site. The inflammation affects the regeneration, as if it is promoted, it accelerates the axonal regeneration; if it is inhibited, it reduces the regeneration. Further experiments showed that two kinds of cytokines, il-1β and tnf-α, are related the inflammation, and they are dynamically controlled by immunes cells, i.e. cytokines works as media to control the axonal regeneration. Cytokines are small proteins, work as messenger to pass signal across different cells, which are similar to hormones and neurotransmitter.

An interaction map can be drawn, according to the evidences found by Tsarouchas et al. [1]. After injury, a massive immune response was observed, with the increasing numbers of several kinds of cells: macrophage, neutrophil. This dynamical system consists of complex interactions between cells and cytokines, For example, il-1β promotes macrophages' increase, and as a major source, more macrophages leads to more expression of tnf-α mRNA. Consequently as media, tnf-α and il-1β plays determinative roles in the axonal regeneration, where generally il-1β promotes the regeneration at early stages, but later has an inhibition effect; tnf-α promotes the regeneration all the time. The presence of the cytokines also affects the number of immune cells, while as sources, more cells lead to more expression of cytokines. Interactions between different kinds of cells are also considered in the system.

This dynamic system is suitable for mathematical modelling, where a set of equations can be written to represent the dynamics of different objects e.g. cells and cytokines. Our interest lies in the modelling of the regeneration period of the dynamic system, and we wish to find that if we can quantitatively study the interactions iniside the system,

based on the proposed hypothesis [1], and furthermore based our own hypothesis which may include different interactions.

The biological background of works in this dissertation is mostly related the the experimental evidences, quantitative data and proposed hypothesis in [1]. The published data on the quantitative measurement of cells and cytokines make the modelling and further analysis possible. Some other interactions proposed in our modelling stages are enlightened from Anderson et al. [4].

2.2 Mathematical modelling

Mathematical models can express real-life problems in mathematical forms, and provide qualitative or quantitative inside-look of the question through various mathematical tools and techniques. Mathematical models can be used for prediction, classification, testing our understanding or hypothesis and many other tasks, thus they are widely used to solve practical problems. However, for a model that describes the zebrafish spinal cord regeneration process, we were less interested to the model's predictive functionality; the main focus is on the explore of uncertainty in parameter estimates and comparison with alternative models. In this case, we wish to use modelling techniques to implement inference methods, test our hypothesis and hopefully find and compare well-fitted models.

According to our understanding of the biological mechanism in the regeneration, hypothesis should firstly proposed in a way that can represent the real mechanism as much as possible. Usually a real-life problem, e.g. the spinal cord regeneration, consists of complex dynamics and interactions with many other environmental factors to be considered; an elaborate description of the real cases using a overly complex mathematical model sometimes can be impossible; assumptions and hypothesis are often proposed to make necessary simplification and abstract for a final reasonably detailed model. Some qualitative models are simple but illustrative for the real-life observed patterns, e.g. Lotka-Volterra (L-V) equation for predator and prey models; they are usually analytically solvable and some properties are used to make predictions (e.g. steady states in ecosystem). If a more precise model is desired, usually more effects or interactions are considered and added as terms of equations to a re-constructed model, e.g. considering environmental factors, a changing environmental capacity and predation from other species in L-V model.

Models like L-V equation are mostly analytically solvable when analysing properties of the model, however, analytical solution for many other general models are not feasible, because of more complex dynamics, or the model itself is not a deterministic model. In the areas where solutions are not analytically feasible, computational techniques are usually applied for approximate solutions. Solvers for different models had been proposed to simulate the system or find roots for some spacial points, such as Explicit Runge-Kutta [5] and LSODA.

For the regeneration process that we wanted to model, differential equations are our preference. Non-linear ordinary differential equations are used in this dissertation, where the only independent variable is time t, denoting the post-lesion time (hours). Further models can also consider other models in other forms, e.g. stochastic models and partial differential equations. For ordinary differential equations (ODE) system, there are several available computational solvers using different algorithms, implemented in MATLAB[1], Python[2] and many other languages. These solvers help to simulate the trajectories of the dependent variables, given initial values and a set of fixed parameters. The goodness of 'fit' then can be measured by compared the simulated data and observed real data.

Often further researches on the modelled problems require the model to be deterministic on its parameters. Parameters in a dynamic system models usually have their practical significance, e.g. describing reaction speed, environmental capacity or the birth rate. There are many approaches to derive these parameters from data or experience. For a popular topic and a widely used models, the parameters' values can be read from previous literatures; however in many others cases, we need to find parameter's values specifically for our model, and numerical values for parameters are mostly preferred, as they suitable for practical analysis and comparison.

To find the best parameters' values given observed data, either optimisation-based approaches (e.g. least-square (LS) fit) or inference approaches can be used. Optimisation-based approaches may face two major drawbacks: over-fitting and local optima [2]. Bayesian inference approaches gain popularity in recent years, which can relief the concerns in over-fitting and provide a measurement for the uncertainty in the inferred values. Local optima can still be a concern, and many efforts have be proposed for a more through explore of the parameter space to avoid this.

2.3 Bayesian inference

Bayesian inference approaches for parameter estimation put Bayes Rule in the centre:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \tag{2.1}$$

where p is the probability density function, θ denotes the parameter set, D is the given observed data.

Alternatively, as $p(D)$ is fixed once the observed data is given, the rule can be rewritten as

[1] `https://uk.mathworks.com/help/matlab/math/choose-an-ode-solver.html`

[2] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html`

$$p(\theta|D) \propto l(D|\theta)p(\theta) \tag{2.2}$$

where l represents the likelihood function. In other words, Eqn 2.2 can be described as 'posterior is proportional to the product of likelihood and prior', and here posterior of parameters is our target in parameter estimation task. Prior $p(\theta)$ should express our initial believes or estimates on the parameter values before observing the data and conducting experiments; likelihood function $l(D|\theta)$ is used to describe how well the parameter set θ explains the data D, in probability values. Once the data is given, θ is the only parameter in the likelihood function. Some optimisation-based methods try to find estimates of parameter set θ s.t. maximise the likelihood, however, such methods will struggle when facing a problem where it is hard to specify an analytical expression of likelihood function, let alone the derivative of it.

Based on Eqn 2.2, to find the posterior of parameter set, prior can be set according to the problem context and our understanding and cover a reasonable ranges; computational methods can be used to approximate likelihood function, e.g. Markov Chain Monte Carlo (MCMC) [6]. As an alterative solution, Approximate Bayesian computation (ABC) approaches do not focus on the evaluation of likelihood function, but focused on simulating the data from the given model and observed data, then used the simulated data to approximate the desired posterior. ABC-rejection [7] is a simple example of the ABC process, and further algorithms e.g. sequential Monte-Carlo (ABC SMC)[8] were proposed to improve the efficiency.

General steps for ABC algorithm includes the following (ABC-rejection):

1. Draw a sample θ^* from parameters' prior distribution. θ^* is a vector that consists values for each parameter
2. Plug θ^* into the model, and use the model to generate simulated data D^*, where D^* is of the same form as the observed data D, e.g. includes same summary statistics
3. The discrepancy between the simulated data and observed data is measured by a distance function $d(D^*, D)$. For a pre-fixed threshold value ϵ, if $d(D^*, D) < \epsilon$ then the sample θ^* is accepted
4. Repeat 1, 2 and 3 until enough numbers of samples are accepted. The posterior distribution can be approximated by these samples

In our study of the regeneration model, full trajectories of the dependent variables are used as the data to be compared, as our goal is to find a model with a parameter set that can intuitively recover the real process as much as possible. The mean values at each time point for each dependent variable is calculated to form D, and for each sample, the ODE is accumulatively solved at the same time points and generate the simulated trajectories D^*. Using other statistics to summary the raw data are also possible but

care must be taken as the choice of summary statistics can largely affect the resultant posterior [9, 10].

Instead of fixing the threshold value, the algorithm can be more adaptive by using a sequence of deceasing threshold values ϵ_t, and derive new populations (called generation) of samples from the accepted samples (called particles) in previous population. An example is ABC SMC [8], which can significantly reduce the required samples compared to ABC-rejection [11]:

1. Set a threshold schedule $\{\epsilon_1, \epsilon_2, ..., \epsilon_T\}$ where T is the total number of populations and $\{\epsilon_1, \epsilon_2, ..., \epsilon_T\}$ is a descending sequence; set the number of particles in each population as N, set the population index $t = 1$
2. if $t = 1$, draw samples from prior distribution until N particles are accepted under threshold ϵ_1, i.e. each of N particles s.t. $d(D, D^*) < \epsilon_1$

   Else ($t > 1$), draw samples from previous population $\{\theta^*_{t-1}\}$ with weights $\{w_{t-1}\}$, perturb the particles using a perturbation kernel $K(\theta^*|\theta)$ and obtain $\theta^{**} \sim K(\theta^*|\theta)$
3. Repeat step 2 until N particles are accept under threshold ϵ_t
4. Calculate the weights of the accepted particles. For particle i

$$w_t^i = \begin{cases} 1 & \text{if t=1} \\ \frac{p(\theta_t^i)}{\sum_{j=1}^N w_{t-1}^j K(\theta_t^j|\theta_{t-1}^j)} & \text{if t} \neq 1 \end{cases} \tag{2.3}$$

5. Normalise $\{w_t^i\}$, increase population index t by 1
6. Repeat step 2, 3 and 4 until $t > N$
7. Output the accepted particles $\{\theta_t^*\}$ and their weights $\{w_t^i\}$ in each generation t

An illustrative graph of the ABC SMC procedures can be found in Figure 2.1. In implementations, many factors can affect the efficiency and goodness of fit, some of them are discussed in the Chapter 4. Rather than a pre-fixed threshold schedule, median or quantile based epsilon schedule can save the total sampling size and thus improve acceptance rates of particles [12]. Some other improvements include using adaptive population size for each generation [13], using adaptive distance function [14] and switching kernels [15], etc. In our implementations these options were tried and compared when applicable and suitable.

In addition, ABC SMC is also capable of model selection tasks [16]. Instead of draw samples from prior of a single model, we can draw samples from multiple models m_i and approximate $p(m_i, \theta|D)$, which is the joint probability distribution of model m_i and its accepted particles θ. In each generation the marginal distribution $p(m_i|D)$ can be calculated as the accepted θ is known, thus models can be ranked according to their probabilities.

The feature that ABC algorithm can perform likelihood-free inference makes it widely used in many dynamic modeling problems, especially in the area of biology [2, 11, 17].

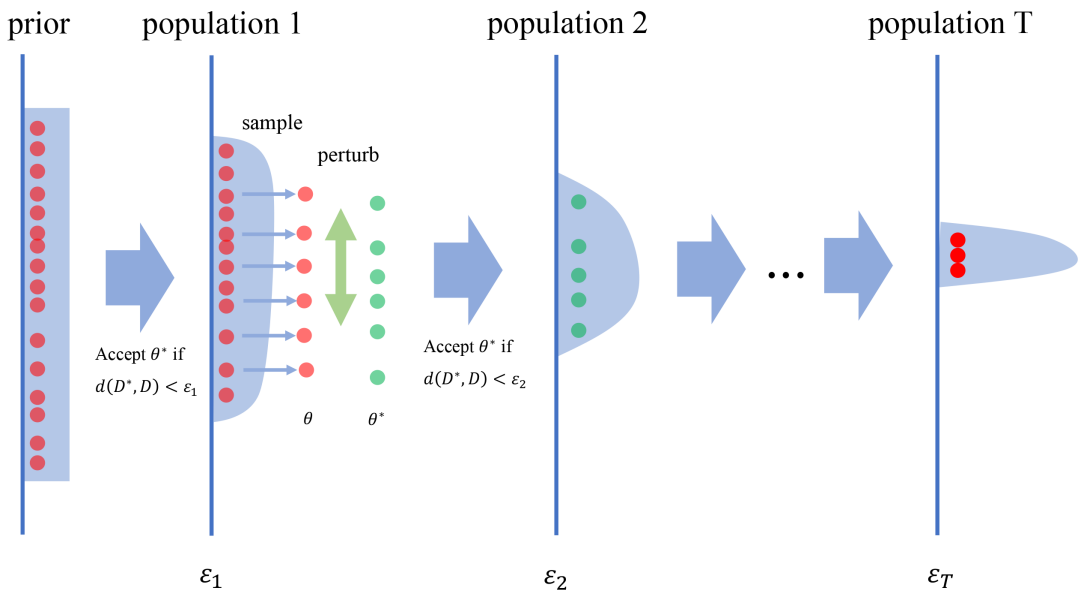

Figure 2.1: ABC SMC sampling process. θ^* denotes a sample drawn from previous population, which is used to generate simulated data D^*. d is a distance measurement, $d(D^*, D)$ represents the discrepancy between observed data and simulated data. for each population t, ϵ_t is a threshold criterion to determine whether to accept that drawn sample

But ABC is not limited to these areas, as it can be generally used in inference tasks to give parameter estimates or model rankings. E.g., it has been successfully applied in cosmology studies [18].

2.4 Software tools

As we decided to apply ABC SMC on our regeneration models for its high-dimensional parameter estimation and comparison of different models, a code development and experiments workflow was expected to be built. The ABC SMC algorithm needs a whole set of mathematical environments (e.g. class of distributions, models and particles) and closely linked sampling and fitting algorithms (e.g. KDE fitting and Gaussian sampling), thus suitable platforms with related software packages are preferred. Regarding this, several packages in `R` and `Python` were studied.

A software packages that integrally implement ABC SMC is preferred, and more options in the settings of the algorithm, more customisable features make it better. Moreover, as the ABC SMC itself is a computationally intensive task that contains massive parallel potentials, an ideal software package is expected to support as least one kind of parallel options e.g. multi-core, GPU accelerating and clusters. Otherwise, a modifica-

tion or rewrite of the software package are expected to produce a reliable input-output framework for our parameter estimation and model selection experiments, where much more work would be introduced.

Preliminarily `pyABC` packages [19] in `Python` was chosen for implementations. `pyABC` supports multiple types of parallelisation, e.g., multicore processing and distributed cluster, and supports resume stored runs, which made our experiments easier. It is well documented and flexible in customisations of the algorithm.

Some other ABC SMC related packages were also examined and considered as backups, include ELFI [1], ABC-SysBio [2], EasyABC[2], GpABC [3], etc. These packages may have their advantages (e.g. ABC-SysBio supports CUDA acceleration) and considered as alternative implementation methods, or can be used to compare performance difference.

Additionally, some other softwares were also need for solving ODE, results analysis and visualisations. These software and hardware environments are listed in Appendix A.

[1] `https://elfi.readthedocs.io/en/latest/`
[2] `https://cran.r-project.org/web/packages/EasyABC/index.html`
[3] `https://github.com/tanhevg/GpABC.jl`

Chapter 3

Mathematical modelling

Mathematical models can describe different kinds of dynamic systems, and thus can be used as a guide for prediction and analysis. An ideal model in our case, can represent reasonable interactions/effects between cells and cytokines and recover the observed data features in different time point.

Proposed models for the regeneration process are in the form of ordinary differential equations (ODE), specifically the time differential form. Terms in the ODE are mostly explainable and corresponds interaction paths.

3.1 Observed data

Our models is built on the basis of the existing experiments data form Tsarouchas et al.[1]. The measurement data include the number of three kinds of cells (neutrophil N, macrophage Φ and microglia) and the relative concentration of four cytokines (il-1β, tnf-α, tgf-β1a and tgf-β3). As proposed in [1], neutrophil and macrophage play important roles in the promotion of spinal cord regeneration with il-1β and tnf-α being the mediation. According to this, our current models focus on the changes of four variables N, Φ, β (for il-1β) and α (for tnf-α). N and Φ is of the unit 'number of cells', β and α is of the unit 'relative concentration' (compared to the value at first time point).

It is noted that the variance of the measured data is relatively high. The summary statistic used for the parameter estimations is mean of measurement data at each time point, assuming that measured data is Gaussian-like distributed. To validate this, the distribution of the measured data points is plotted and examined to see if the mean value can represent the distribution. The result is that at most time points the measurement values are Gaussian-distributed, although some distributions are skewed. One abnormal distribution is observed at time point 120 h post-lesion (hpl) for macrophage where there are two concentrations. Mean value can summarise most data and thus is still used as the target observed data. A plot of the mean of the four variables is shown in Figure 3.1.

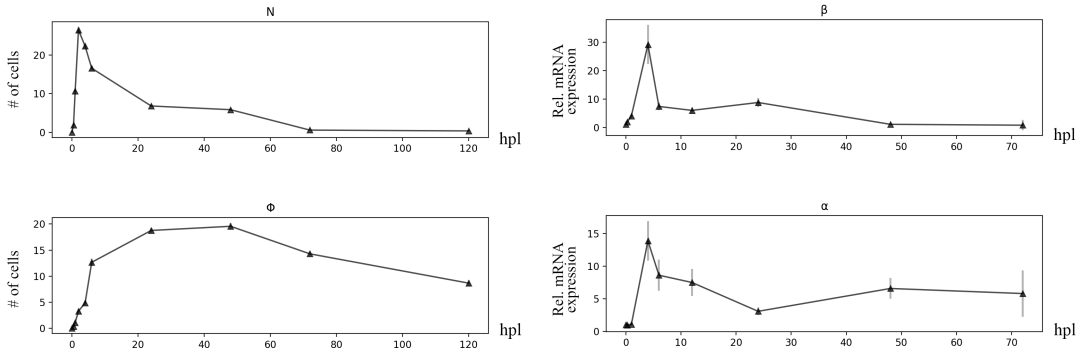

Figure 3.1: Mean of the observed data for neutrophil (N), macrophage (Φ), il-1β and tnf-α, from experiment results of [1]. Error bars indicate standard error of mean

3.2 Hypothesis and Models

5 models in total are proposed according to different hypothesis. At first our tests and implementations of ABC SMC for parameter estimations use only the basic model for developing propose. After the parameter estimation framework is built and tested, more models are proposed, in order to calibrate and adjust the basic model such that it can represent the observed regeneration process better or can be used to test our hypothesis.

All these models assume the involved interactions is within two kinds of cells (neutrophil and macrophage) and two kinds of cytokines (il-1β and tnf-α) and use the data presented in Figure 3.1 for the inference task. Interactions or effects from other cells or cytokines is not considered as there might not be corresponding data available.

3.2.1 Basic model

A preliminary model is proposed according to [1] and used to build and test the inference framework code. An interaction map of the model is shown in Figure 3.2. This model is a simplification a the process described in [1] with some minor interactions ignored. To describe the parameters' units, we denote the unit of N and Φ i.e. number of cells as 'cell', and denote the unit of β and α i.e. relative mRNA expression as 'unit' for simplicity.

It is assumed that there are negative feedbacks for all the variables. MORE DESCRIBE.

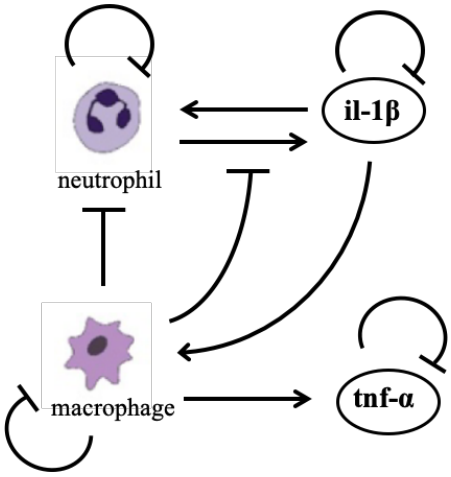

Figure 3.2: Interactions modelled in the basic model (model 1) based on Tsarouchas et al.[1]. Lines ended with arrow represent promoting effect, lines ended with T-connectors represent inhibition

$$
\begin{aligned}
\frac{\mathrm{d}N}{\mathrm{d}t} &= \lambda_N + \kappa_{N\beta}\beta - \mu_N N - \nu_{N\Phi} N\Phi \\
\frac{\mathrm{d}\Phi}{\mathrm{d}t} &= \lambda_\Phi + \kappa_{\Phi\beta}\beta - \mu_\Phi \Phi \\
\frac{\mathrm{d}\beta}{\mathrm{d}t} &= \frac{s_{\beta N} N}{1 + i_{\beta\Phi}\Phi} - \mu_\beta \beta \\
\frac{\mathrm{d}\alpha}{\mathrm{d}t} &= s_{\alpha\Phi}\Phi - \mu_\alpha \alpha
\end{aligned}
\tag{3.1}
$$

Parameter	Definition	Units
λ_N	Self-increase rate of neutrophil	$cell/h$
$\kappa_{N\beta}$	Promoting effect coefficient by il-1β	$cell/(unit\cdot h)$
μ_N	Coefficient of negative feedback of N	h^{-1}
$\nu_{N\Phi}$	Coefficient of inhibition of both N and Φ	$cell^{-1}\cdot h^{-1}$
λ_Φ	Self-increase rate of macrophage	$cell/h$
$\kappa_{\Phi\beta}$	Promoting effect coefficient by il-1β	$cell/(unit\cdot h)$
μ_Φ	Coefficient of negative feedback of Φ	h^{-1}
$s_{\beta N}$	Production rate from N	$unit/(cell\cdot h)$
$i_{\beta\Phi}$	Coefficient of inhibition to the production	$cell^{-1}$
μ_β	Coefficient of negative feedback of β	h^{-1}
$s_{\alpha\Phi}$	Production rate from Φ	$unit/(cell\cdot h)$
μ_α	Coefficient of negative feedback of α	h^{-1}

Table 3.1: Parameters introduced in the basic model (model 1) [REMOVE unit]

3.2.2 Alternative models

Model 2 and model 3 As the observed data indicate, this dynamic system has a steady state where the inflammation is resolved and immune cells should not be present at the injury site. Regarding this, the self-increase parameter λ cannot be constant, thus a exponentially decaying λ term is introduced and model 2 is proposed. (Eqn. 3.2). Also, the inhibition of il-1β production, i.e. term $i_{\beta\Phi}\Phi$ is considered to be ignored for a more simple model, in which case the relative expression of il-1β is only affected by the number of neutrophil and the negative feedback from itself. This case corresponds to model 3, written as Eqn. 3.3.

Model 2 and model 3 introduce one extra parameter a, which is a coefficient in the exponentially decaying λ, determining the decay speed, with the unit h^{-1}.

$$\begin{aligned}
\frac{\mathrm{d}N}{\mathrm{d}t} &= \lambda_N e^{-at} + \kappa_{N\beta}\beta - \mu_N N - \nu_{N\Phi} N\Phi \\
\frac{\mathrm{d}\Phi}{\mathrm{d}t} &= \kappa_{\Phi\beta}\beta - \mu_\Phi \Phi \\
\frac{\mathrm{d}\beta}{\mathrm{d}t} &= \frac{s_{\beta N} N}{1 + i_{\beta\Phi}\Phi} - \mu_\beta \beta \\
\frac{\mathrm{d}\alpha}{\mathrm{d}t} &= s_{\alpha\Phi}\Phi - \mu_\alpha \alpha
\end{aligned} \tag{3.2}$$

$$\begin{aligned}
\frac{\mathrm{d}N}{\mathrm{d}t} &= \lambda_N e^{-at} + \kappa_{N\beta}\beta - \mu_N N - \nu_{N\Phi} N\Phi \\
\frac{\mathrm{d}\Phi}{\mathrm{d}t} &= \kappa_{\Phi\beta}\beta - \mu_\Phi \Phi \\
\frac{\mathrm{d}\beta}{\mathrm{d}t} &= s_{\beta N} N - \mu_\beta \beta \\
\frac{\mathrm{d}\alpha}{\mathrm{d}t} &= s_{\alpha\Phi}\Phi - \mu_\alpha \alpha
\end{aligned} \tag{3.3}$$

Model 3 can be regarded as a simplification of model 2, as it can be treated as model 2 with parameter $i_{\beta\Phi} = 0$. Among the proposed three models, model 1 is a naive one that was proposed at very first time and used as an ODE 'template' to build and test parameter inference framework. As the implementation was successful, model 2 and 3 was proposed as we were trying to calibrate some terms and find a better model. After fitting, model 2 is supposed to be better than model 1 as it corrects the problem that appears at the final time points (which relate to the steady state). Model 3 makes a small simplification on model 2 and is theoretically less 'general' than model 2.

Model 4 and model 5 After the first model selection experiment, it was found that some significant features presented in the observed data were not recovered by any

of the existing model. To resolve this, attempts were tried to introduce more interactions within the dynamic system considering the biological and mathematical context. Extra promoting effect to the expression of tnf-α was considered, by either adding a phenomenological term (which means the same effect as directly promoting but the underlying mechanism is unclean) or adding a term that represents a promoting effect to the production process of tnf-α, namely model 4 (Eqn. 3.4) and model 5 (Eqn. 3.5).

$$\begin{aligned}
\frac{\mathrm{d}N}{\mathrm{d}t} &= \lambda_N e^{-at} + \kappa_{N\beta}\beta - \mu_N N - \nu_{N\Phi} N\Phi \\
\frac{\mathrm{d}\Phi}{\mathrm{d}t} &= \kappa_{\Phi\beta}\beta - \mu_\Phi \Phi \\
\frac{\mathrm{d}\beta}{\mathrm{d}t} &= s_{\beta N} N - \mu_\beta \beta \\
\frac{\mathrm{d}\alpha}{\mathrm{d}t} &= s_{\alpha\Phi}\Phi - \mu_\alpha \alpha + d_{\beta\alpha}\beta
\end{aligned} \tag{3.4}$$

$$\begin{aligned}
\frac{\mathrm{d}N}{\mathrm{d}t} &= \lambda_N e^{-at} + \kappa_{N\beta}\beta - \mu_N N - \nu_{N\Phi} N\Phi \\
\frac{\mathrm{d}\Phi}{\mathrm{d}t} &= \kappa_{\Phi\beta}\beta - \mu_\Phi \Phi \\
\frac{\mathrm{d}\beta}{\mathrm{d}t} &= s_{\beta N} N - \mu_\beta \beta \\
\frac{\mathrm{d}\alpha}{\mathrm{d}t} &= (s_{\alpha\Phi} + f_{\beta\alpha}\beta)\Phi - \mu_\alpha \alpha
\end{aligned} \tag{3.5}$$

Parameter	Definition	Units
$d_{\beta\alpha}$	Coefficient of promoting effect from β	h^{-1}
$f_{\beta\alpha}$	Coefficient of promoting the production of α by β	$cell^{-1} \cdot h^{-1}$

Table 3.2: Parameters introduced in model 4 and 5

3.2.3 Model evaluation and comparison

[general topics on evaluating a model]

[bayes factor for model selection]

3.2.4 Limitations

[hypothesis]

[available data]

[model misspecification]

Chapter 4

Parameter estimation and model comparison

Estimation of parameters in the proposed models is the main task of this section, which involves the implementation of ABC SMC and several experiments. ABC SMC-based model selection is another task that can be easily conducted with the developed parameter inference framework.

The build and test of code were under `Python` environment with `pyABC`[19]. Some other code e.g. shell script and `R` code are also used to perform the experiments and analyse the results. Software and system environment used in our implementation can be found at Appendix A. The build and test were performed on local computers and HPC facilities available within EPCC.

4.1 Implementations and code

[ABC implementations details, e.g. distance, population, ODE related functions]

[how the code is developed and built]

According to our preliminary studies, ABC SMC framework implemented using `pyABC` in `Python` was adopted in this project. `pyABC` is a popular SMC software packages [19] that has been used form many SMC studies and applications [13]. `pyABC` provides an open-source framework for likelihood-free inference, which is an implementation of ABC SMC algorithm and a tool-box for our own inference and analysis tasks. Besides, it supports multi-core execution implemented using `multiprocessing` package for scaling-up, which is of our interests and suitable for the followed performance experiments.

The inference framework code for the project was developed and tested in local environment (macOS 10.15.6, see Table A.1) at first, then the functional version was deployed on compute node of ARCHER and Cirrus for parameter inference experiments

and model selection task. The scaling-up experiment was performed on Cirrus, and the profiling and its analysis were performed on local machine. Some development and debug were on the remote machines to ensure experiments run correctly on compute node.

The ODE model-related code and utilities were firstly developed to enable reusable functions, variables and data calls. It includes the code for the ODE models, ODE solver, data structures, data format transform functions etc. By doing this in a separate code file, we can change options and activate ABC SMC easily in the implementation code file, without creating definition and duplicated code for the models and parameter priors.

Some modifications were made to the source code of `pyABC`. For example, the laboratorial measured data were not considered 'complete' for modeling and analysis purpose: the available time points for cells and cytokines are different, e.g. cell number was not measured at 0.25 hpl and cytokines' expression was not measured at time point 120 hpl. These missing values were treated as 'ignored' when calculating the distance between observed data and simulated data. To cope with this built-in distance function code is modified.

Analysis code was also wrapped up into functions which can be easily called after an ABC SMC run is finished, which includes multiple visualisations and summary statistic of the results. For different tasks there are also separated analysis code file. Other analysis tools used in the project include built-in database visualisation tool, R code and Microsoft Excel.

The project code was managed via `git` version-control and available as a GitHub repository[1]; it also made the cross-hardware synchronisation of code versions convenient.

After several development iterations, code files for ABC SMC implementation now mainly had two types: code files for infer-back experiments (udsing synthetic data), and code files for parameter inference (using experimental data). Model selection can be easily done by adding more model to the implementation code file.

4.2 ABC SMC and model hyperparameters experiments

As the key focus of this project is on the parameter estimations of dynamical system, the ability of inference on the proposed models was firstly examined before actual inference using the experimental observed data.

In the first part, synthetic data is used with known true parameter values. The algorithm implementation is evaluated by the efficiency and goodness of resultant model under different implementation options and ABC SMC hyperparameters.

[1] `https://github.com/chaolinhan/MSc_Project`

Then according to findings in the first part, several options with good performance are tried in the inference using experimental observed data (Figure 3.1). To obtain a more accurate and general results, these experiments are repeated for 3 or more times.

[including the options that are tested/ tried/ not adopted]

[including experiments results and discussions]

[how do these options correspond to biological model and how to set them accordingly for proposed model]

The ABC SMC inference can have largely different performance when using different hyperparameters and implementation options. To firstly test the ability of inference and observe the results of different options, synthetic data with known true parameter values are used as target data. The true value of parameters that are used to generate synthetic data is listed in Appendix B.1. These true values are firstly obtained by fitting a least square fit of model 1 onto the experimental data (Figure 4.1), the target data is then generated using these parameter values via ODE solver in `Python`.

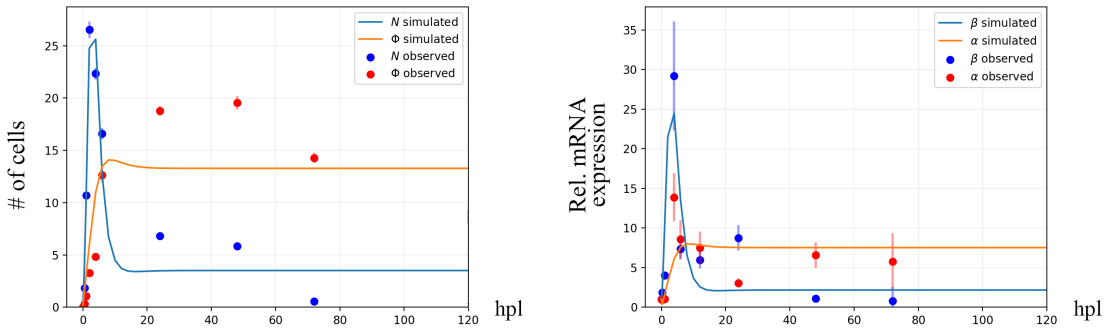

Figure 4.1: Synthetic data generated with known parameter values (line plot), compared to experimental data (scatter plot). Known parameter values are obtained from a least square fitting of the observed data

[TABLE for known parameter values]

[why use synthetic data]

Among them the following topics are studied using the synthetic data.

4.2.1 Perturbation kernels

Perturbation kernels work in the sampling process. In each generation t, samples are firstly taken from the previous population $\{\theta_{t-1}\}$ with weights $\{w_{t-1}\}$, then perturbed using the perturbation kernel $K(\theta|\theta^*)$. We keep sampling until N particles are accepted, where N is the pre-set population size. After that the new weights are calculated and normalised.

The perturbation kernel is called in the sampling of every particles and determines how the new perturbed particle is chosen, thus it is influential to both computation complexity and the resultant posterior distribution. Generally, a local perturbation kernel may face the risk of being stuck in local modes (e,g., local optimal), but it may need less computational operations, or could generate a population with a higher acceptance rate if the successive epsilon values are close; A kernel with wide variance, or spreading out in a large space could help in resolving the local optimal problem by a more thoroughly exploration of the parameter space, however it can be more computation-intensive and result in a lower acceptance rate. A desired optimal kernel should balance the trade-offs; their property and criteria is discussed in [15].

There are several common choice of perturbation kernels. Among them multivariate normal kernel and local M-nearest neighbour model is preferred to be applied on our models. A covariance matrix Σ_t of accepted particles is calculated form previous generation and used in multivariate normal kernel: $K(\theta|\theta_{t-1}) \sim \mathcal{N}(\theta_{t-1}, \Sigma_t)$. It is illustrated to be more efficient than uniform kernel and component-wise normal kernel in relecting the true posterior structure [15]. It has been proved to perform well in several dynamic system models [2, 17, 11] for the parameter estimation and model selection tasks. The $scaling \in (0, 1]$ parameter in `pyABC` will be multiplied to the covariance to produce a 'narrower distributed' perturbation result.

Local M-NN kernel provided by `pyABC` is also tried to provide a comparison. A local kernel density estimation (KDE) fit is used with M nearest neighbours considered.

The kernel experiments is designed to explore the efficiency of SMC on our dynamical systems. Given the same fixed threshold schedule, kernels that need less total samples and have higher acceptance rate would be our preference. The experiments compared multivariate normal kernel with local MNN kernels with different parameters, using synthetic data to infer back the parameters. The acceptance rate among each time point and total required samples are compared after the experiment to give suggestions on the kernel selection in the real data inference. As the threshold schedule is fixed, the final population should have similar discrepancy to the target data thus here the goodness of fit i.e. recover of target data trends/features and errors of the inferred parameters compared to true values is not discussed.

Kernel experiment results

To compare kernels, a fixed schedule schedule is used with minimal epsilon i.e. ϵ_t set to 10.0. From the total required samples graph Figure 4.2, the efficiency can be compared. Among the tested kernels, the local M-NN with M=50 has the best performance: it requires the least number of samples and has higher acceptance rates among almost all generations. Local M-NN with M=750 is the slowest kernel.

For local M-NN kernels, generally a greater M will lead to lower acceptance rate and more required particles in each generation. As shown in Figure 4.3, local M-NN with M=750 is the lowest curve. Consequently, if greater M is test, then it will have even

lower acceptance rates; the maximum M=2000 (whole population is considered) will have the lowest acceptance rates.

Multivariate normal kernel has a performance between local M-NN M=250 and M=100. It proves that rather than a trivial normal kernel, multivariate normal kernels is more efficient facing the concentrations of joint distributions among multiple parameters [15]. 'Scaling' option can narrower the distribution calculated from the last population, thus making the kernel more 'local' by sampling under a smaller variance. Similar to small M in local M-NN, small 'scaling' is also more efficient in our experiment of the model 1. Besides the fixed threshold schedule, a median threshold schedule with the same final threshold $\epsilon_{20} = 10.0$ is also tested and similar results are observed (Figure B.1 and B.2, Appendix B2).

It seems that a more 'local' kernel can give better performance by efficiently sampling around concentrations in parameters' distribution and approximate the posterior distribution. This holds in most cases under a given target final threshold value ϵ_t, however may not produce the proper result want: a more local kernel, e.g. local M-NN with small M and multivariate normal with 'scaling' is more likely to be stuck in local optimality, as the kernel is less likely to sample particles that are far from local concentrations, although these local optimal are still accepted under given threshold. In other words, using local kernels the epsilon may quickly converge to to local optimal; if a even smaller epsilon is desired, the local kernel can hardly generate enough particles to find another matched local modes. Also, the shapes of posterior distribution will affect the performance of local kernels [15]. One obvious example is presented in Figure 4.2, where local M-NN with M=50 suffered from a local optimal: the last generation takes much more samples to meet the required threshold.

Conclusion

Local kernels can be fast bur unstable in some case; for a general parameter inference of our models, multivariate kernels are preferred, as a good fit is our prior target; some local kernels are also worth trying after the multivariate normal kernel, e.g. local M-NN with M$\leq$ 250 (for population size of 2000, i.e. 12.5% of the population) and multivariate normal with scaling. Also multivariate M-NN is worth trying, although it is not built in `pyABC` and requires additional implementations.

4.2.2 Adaptive functions and factors

[how adaptive distance work]

SMC can be more adaptive by introducing adaptive distance function and adaptive population; besides, factors can be applied to manually 'normalise' the data.

The trivial distance function is set to Euclidean distance (2-norm distance)

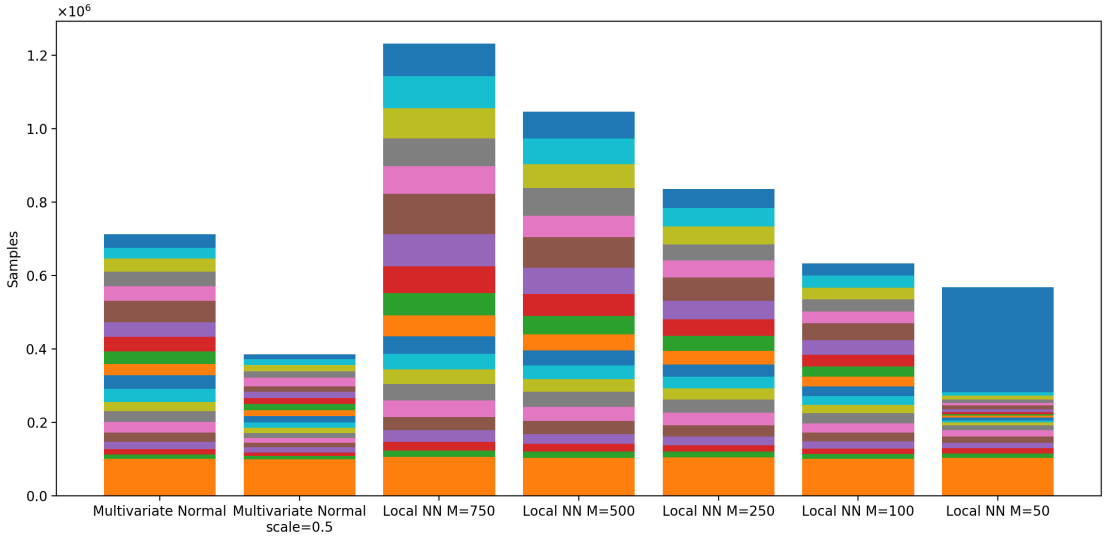

Figure 4.2: Total required samples of different kernels after 20 populations (2000 particles in each population). Different color represents different generations (bottom to top: population 1 to population 20)

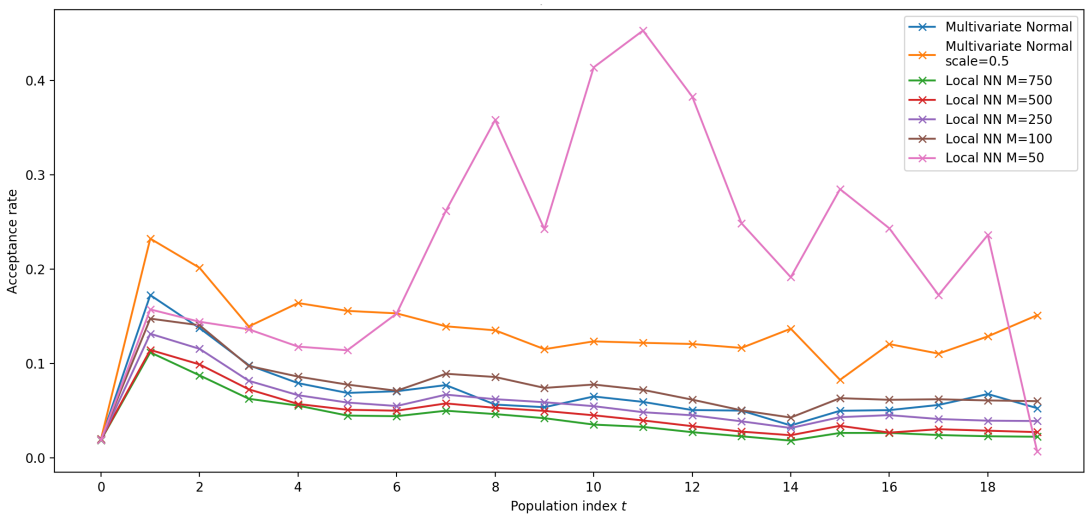

Figure 4.3: Acceptance rates of different kernels in 20 populations. Each population has 2000 particles

$$D = \sqrt{\sum_i \Delta x_i^2} \tag{4.1}$$

where i is the index of data points and Δx_i is the discrepancy between observed data and simulated data at data point i, i.e. $\Delta x_i = x_{i,simulated} - x_{i,observed}$. In this case data points are 12 time points of four variables i.e. 48 data points in total.

A weighted 2-norm distance can be written as

$$D = \sqrt{\sum_i w_i \Delta x_i^2} \tag{4.2}$$

where a weight w_i is assigned to every data point. If data point i have a higher weight, then the data value is regeared to be more informative, i.e. giving more help in inferring the true posteriors; weights can be either pre-set according to prior knowledge of the problem, or using adaptive method to be dynamically calculated according to , known as adaptive distance.

Adaptive distance is changing the weights of all data points after each generation iteration, trying to assign informative data points higher weights. Additionally, implementation of adaptive in `pyABC` also introduces additional factors f_i multiplied to weights [14]

$$D = \sqrt{\sum_i f_i w_i \Delta x_i^2} \tag{4.3}$$

Factor is helpful as an option for efficiency. When some data points are equally informative, the manually pre-defined can make the distance more focused on certain data point, or behave like a normalisation that can balance the scales of different summarised statistics (here is the 48 mean values). There are two appliance case studied: (1) factors used as an normalisation option and (2) factors used to give more focus on some features of the observed data. For (2), as we noticed that the main features (e.g. rapid increase, peaks, fluctuations) are mostly observed in the first half of the time points i.e. 0 - 30 hpl, factors are tried in the later parameter estimation of real data where a more accurate fit of the curve is desired.

Adaptive population [13] can adapt the population size of each generation according to the mean correlation of variation error of the previous population. In our early test the maximal population size that is allowed is set to 5,000 and the adaptive strategy always select the upper bound, as a results of wide parameter space or the wide posterior approximation shape.

[what is factor]

[what is adaptive distance]

Experiments results

[FIGURE here]

Two types of factors and adaptive distance function are tested by running the same number of generations (20 generation, 2000 particles per generation). From FIGURE, by applying factors the total required samples are less, and the adaptive distance requires much less samples; their acceptance rates are also higher than the standard one. It seems that adaptive function and factors could help to converge more quickly.

However considering the result after 20 generation (FIGURE), the resultant model gives the contrary preference. Applying factors and using adaptive distance can lead the resultant model less accurate (after the same generation iterations): they have wider inter-quantile range and may requires more generations to converge.

Conclusion

We noticed the significant efficiency improvement of these adaptive options; they can largely improve the acceptance rate in some cases. However, as the factors and weights are directly multiplied to Δx and thus the metrics for distance are changed, a direct compare of the efficiency under the same threshold schedule is unfeasible; after the same number of generations they can reach a small ϵ_t but still the approximated posterior is not as accurate as the standard implementation. AS a result, our implementation of the parameter inference will only consider standard distance function and factors are only tested for improvements (FIGURE).

4.2.3 Data size, prior distribution and population

[experiments plan]

Some other hyperparameters, e.g. feed-in data size, prior distribution, number of populations and number of particles in each population were also of our interest. Experiments were designed to explore how these options affect the goodness of fit and efficiency of the implementation.

Regarding the dynamic systems model, SMC was applied with two data size options, three prior distribution (uniform distribution, wider uniform distribution and log-uniform distribution). The population options i.e. population size and number of populations were also tried with different values. These experiments intended to give suggestions on the later SMC implementation on experimental data.

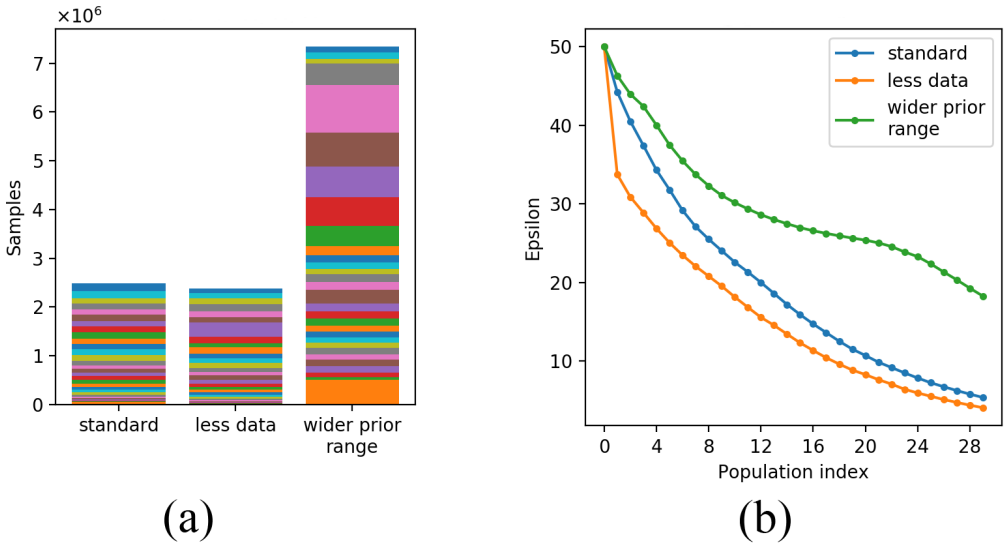

Figure 4.4: Data size and prior distribution range experiment. (a) Total required samples, (b) Epsilon values under median epsilon schedule

Experiments results

The results from data size and prior distribution range is shown in FIGURE. The standard implementation is fed with 120 data points (30 data points for each of the four variables), the 'less data' is fed with 48 data points, which is the case of the real experimental measurement. Although much less (60%) data is used, the total required sampling numbers is not much less (XX%). Although the data size does not affect the sampling process much and the inferred model are all considered acceptable when compared to the synthetic data (FIGURE), cares should be taken when using less data point, or using other more summative statistics, e.g. mean, standard deviation, peak values, starting and ending values etc. As shown in [11], less data can lead to a model with more variance, under-fitting or missing some local features e.g. local peak.

The prior range and distribution can largely affect the execution of SMC (FIGURE and FIGURE). Samples are taken from a high dimensional parameter space in each sampling process, wider prior distribution ranges will making each sampling explore more in the parameter vector space. It suggests that a narrower prior is preferred and we should make a more accurate and confident prior belief as possible for the efficiency concerns.

[discuss uniform and log uniform when result ready]

[FIGURE HERE]

Summary

This subsection aims to preliminarily explore ABC SMC settings using synthetic data with known true parameter values, thus the efficiency and the goodness of the resultant model can be compared across these options. Conclusion can be drawn to choose efficient and proper options of ABC SMC to be applied on the target models with experimental data. It also provides reference for a general inference task using SMC with high dimensional parameter space.

4.3 Parameter estimation and model comparison

4.3.1 Model 1, 2 and 3

[separated ABC run on model 1, 2 and 3]

Using the suggested options in the previous section, the target data (Figure 3.1) is prepared as input for the SMC inference framework. Regarding the result, the population size and number of generations are adjusted several times to obtain an general informative posterior which should be stable across repeated runs and avoid local optimal as much as possible.

Parameter estimation of model 1, 2 and 3 was tried with different prior distribution setting: distribution range [0, 25] and [0, 75], amd distribution type uniform and log-uniform. All these ABC SMC runs uses 2,000 particles per population and 30 generations.

Log-uniform-shaped prior seems to give a better fit of the model, and wider prior range [0, 75] does not give a more accurate result. Then distribution range [0, 25] was tried, in case some parameter have a true posterior laying between [25, 50].

The results of prior distribution log-uniform [0, 50] is shown in Figure 4.5. For model 3, the acceptance rates and epsilon path are presented in Figure 4.6. For model 3, the estimated parameters' posterior is shown in Figure 4.7. Figure 4.5 plots the estimated curve of models, using the mean value of each parameter's approximated posterior; also 1000 particles (each particle is a parameter set) were sampled and used to generate simulated trajectories of the four variable, and the mean (blue) and 25th to 75th percentile range (grey) are calculated from the 1000 simulated data.

Before applying model comparison methods, some features of the three model can still be compared. From simulated curves, It can be seen that all the inter-quartile ranges are tight around the mean value curve, which indicates the approximated posterior are concentrated to some degree; the peak-shaped posterior distributions (Figure 4.7) agrees with this. All epsilon values converges to a steady level, and model 3 has a lower final epsilon value and consequently the simulated trajectory is more close to the observed data (model 3, Figure 4.5).

Parameter	Estimated mean value (uniform)	Estimated mean value (log-uniform)
λ_N	14.1	13.3
a	9.67	0.0170
$\kappa_{N\beta}$	18.0	0.259
μ_N	13.2	0.0180
$\nu_{N\Phi}$	0.742	0.131
$\kappa_{\Phi\beta}$	0.239	0.156
μ_Φ	0.154	0.0350
$s_{\beta N}$	6.75	1.21
μ_β	6.38	1.32
$s_{\alpha\Phi}$	17.0	5.83
μ_α	11.9	2.43

Table 4.1: Estimated parameter values of model 3, using uniform prior distribution and log-uniform prior distribution respectively

The acceptance rates is fluctuating and have a gradually increase trend for all the three models, as the acceptance rate becomes higher when the true posteriors are gradually approximated. Compared to model 1, model 2 gives a better fit of the decreasing trend at the second half of the observed data, by using a exponentially decaying self-increase rate λ_N instead of a constant. The resultant simulated data from model 2 and 3 are in the similar trends and the most obvious difference is that for model 3, the simulated data of Φ and α are more 'flat'. We expect model 3 to be the best model as it reaches a threshold value and intuitively gives a better fit, although the fit could not be considered generally a good representation of the biological process that we want to model, as the simulated data are significantly biased from the observed data for tnf-α. A sharp peak at 4 hpl is observed from the measurement (Figure 3.1) but no similar features are represented by any of the three models.

[model comparison among model 1, 2, 3]

Next a model selection experiment with different prior was conducted, using the same ABC-SMC framework. As in the parameter inference, two distributions (uniform and log-uniform) are tested with three different interval ([0, 25], [0, 50] and [0, 75]). The results show that model 3 wins with nearly 100% model probability at most runs after 30 generations, except log-uniform [0, 75]. All the model probability results are of great discrepancy (e.g. 100% model 3 and 0% for model 1 and 2), so Bayesian factor is not calculated.

Figure 4.8 shows the model selection process, under log-uniform [0, 50] prior distribution. Model 3 gains total advantage over model 1 and 2 after 7th population. Together with conclusions above, model 3 is considered to be the the best model to represent the target data so far.

Compared to uniform distributed prior, we found that log-uniform distributed prior is

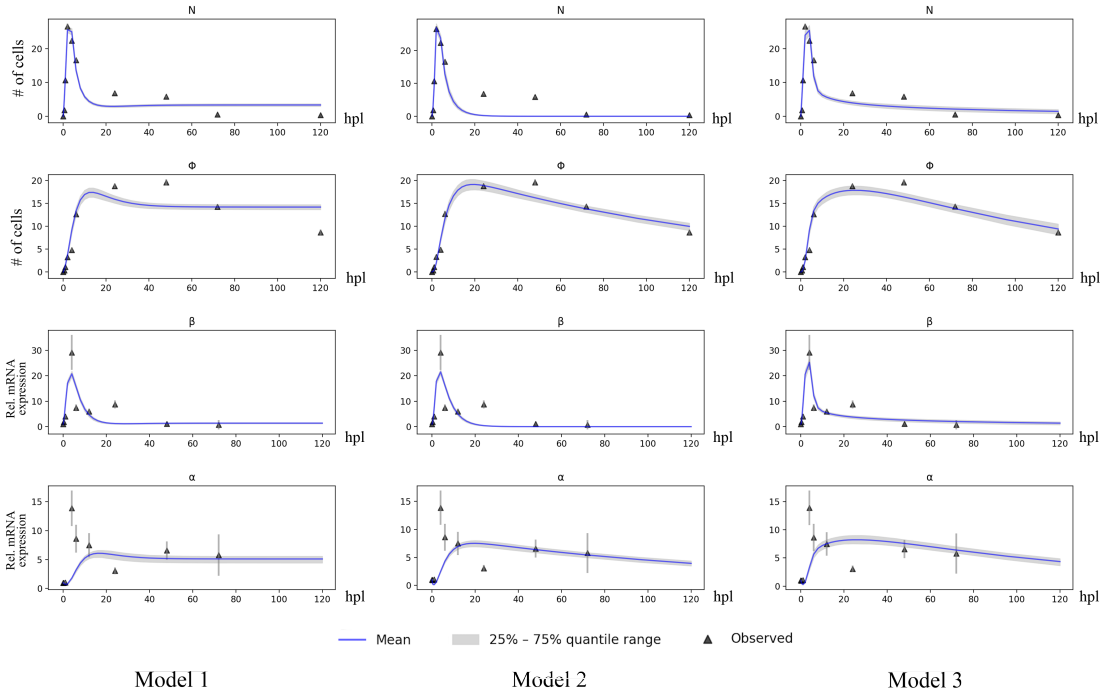

Figure 4.5: Simulated trajectory from the last population of model 1, 2 and 3. Error bar indicates SEM. 500 randomly chosen particles from last population are used to generate a sequence of trajectories where mean and inter-quartile range are calculated from

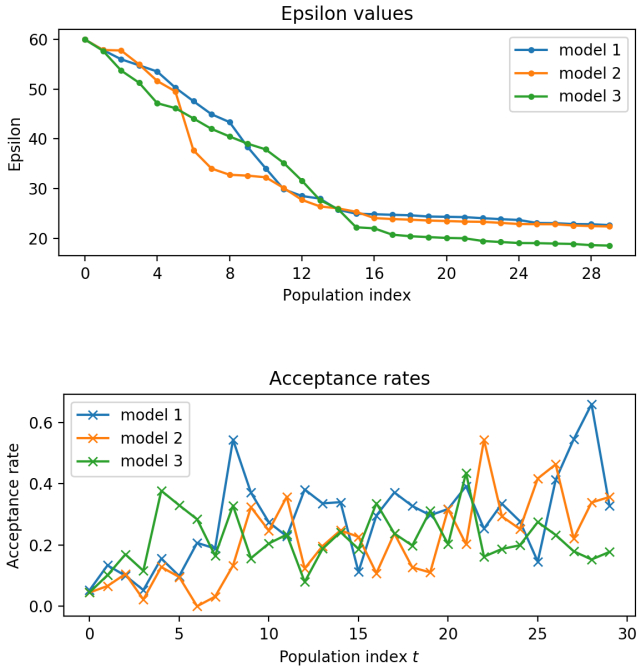

Figure 4.6: Epsilon trends and acceptance rates of model 1, 2 and 3

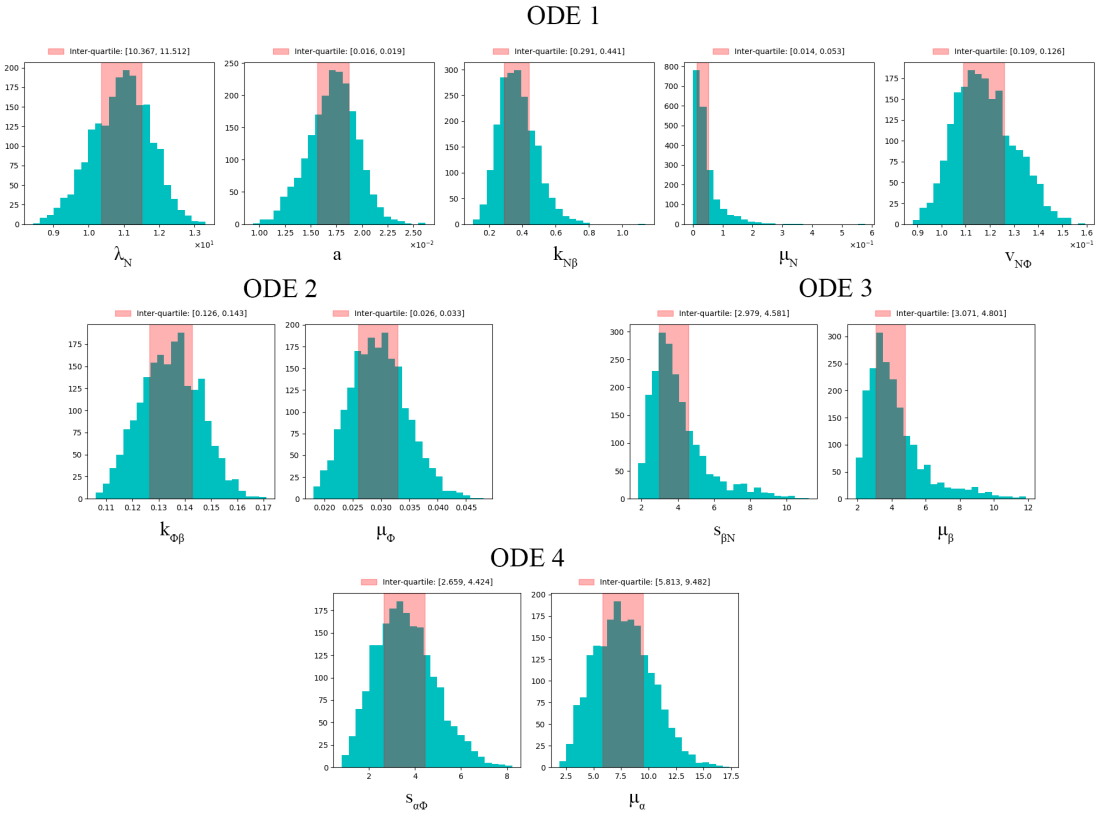

Figure 4.7: Estimated posterior distribution of parameters in model 3. Shaded range indicates 25%–75% quantile of the population

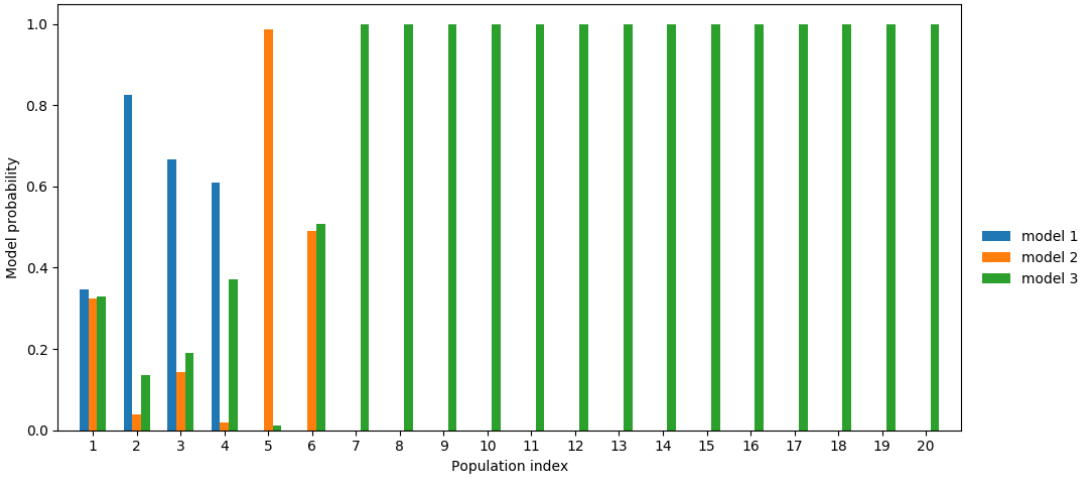

Figure 4.8: Model comparison of model 1, 2 and 3. From population 8 to population 30, model3 always wins

more suitable in the inference. Log-uniform results in an obviously better fit and narrower inter-quantile interval (Figure 4.5 and B.3), and the final epsilon value is also smaller. Log-uniform tends to assign values that are close to the left boundary (i.e. zero in our case) higher density, i.e. they are more likely to be sampled in the first population. As a result, the estimated parameter posteriors are expected to have smaller mean and/or skewed to left of the x-axis. The estimated values proved this (see 4.1; also observed in other models and prior interval experiments). It indicates that some of the true parameter tend to have small values that are close to zero, and using log-uniform distribution could results in a faster and more satisfying inference. Regarding this, further experiments all use log-uniform distribution only.

4.3.2 Model 4 and 5

[features that derive model 4 and 5]

We observed that the existing models cannot well represent the trajectory of relative expression of tnf-α: from experimental data, a rapid increase of tnf-α expression is observed in 0–4 hpl, followed by a rapid drop; all the models cannot well recover this feature and most test results see a less-rapid increase followed by a gradually decree and the observed peak value is not reached (e.g. Figure 4.5). Also in some other case e.g. Figure B.3, tnf-α trajectories is well fitted at the cost of under-fitting Φ. Hence efforts had been spent to propose more alternative models that can possibly solve this problem.

The two proposed alternative models is based on model 3 and try to add extra term in the tnf-α equation $\mathrm{d}\alpha/\mathrm{d}t$. As macrophage is the main source of tnf-α production, and a similar sharp increase-and-drop trend is observed in the il-1β trajectory, two hypothesis and corresponding terms are propose as follows.

Model 4 It is assumed that the expression of il-1β would have a promoting effect, representing by a phenomenological term $d_{\beta\alpha}\beta$ and $d_{\beta\alpha}$ is a new model parameter (positive constant). The promoting effect here is regarded to have the equivalent effect as directly promoting by il-1β, but the underlying mechanism is unclear so far. In mathematical view, this additional term can accelerate the expression of tnf-α in the first few time points and help to recover the peak-shaped trajectory. The proposed model is written as Equation 3.4.

Model 5 Alternatively, we considered promotions to the tnf-α production, i.e. $s_{\alpha\Phi}\Phi$. It is assumed that il-β could accelerate the production process. An additional term $f_{\beta\alpha}\beta$ is introduced to production rate, with $f_{\beta\alpha}$ being a new model parameter (positive constant). This model meets the biological context considering the source of tnf-α. The proposed model is written as Equation 3.5.

[further comparison of model 3, 4 and 5]

Model 4 and 5 are base on model 3, so in this phase experiments of these three models are conducted for separately parameter inference and overall model comparison. Based on our experience in conducted experiments, we used the following setting for parameter inference:

- population size: 2000 and 5000
- number of populations: 30 generations
- prior: [0, 50] log-uniform
- perturbation kernel: multivariate normal kernel

Factors (Equation 4.3) are also tried to find if we can force the inference framework to give more importance on the first few data points. In this experiment, for each trajectory first 8 data points are assigned higher factors (0.75), and the rest 4 data points are assigned with lower factors (0.25). All these runs are conducted on remote machines and the output database files are retrieved form analysis.

Figure 4.9 shows the simulated data from the inferred models, with population size 2000. It can be seen that the addressed problem in the trajectory of tnf-α is partially relieved in model 4 and 5: compared to model 3, a peak appears around 4 hpl; it can represent more features in the observed data and consequently the final reached epsilon (after 30 populations) is smaller than that of model 3, which proves that our proposed modifications in model 4 and 5 are helpful in recovering more features in observed data.

By intuition model 4 and 5 would be the best models sofar that can recover most of observed data features. Model 1, 2 and 3 con also converge to a low final epsilon value (20), but a key feature observed in target data – a peak at around 4 hpl – is not fitted. Some features, e.g. fluctuations in il-1β and tnf-$alpha$ trajectories at around 20 hpl are still not ideally fitted; all existing models give a steady or gradually decrease trends for all the four variables after 20 hpl, and fluctuations in the observed data are fitted by smooth and steady curves.

After applying factors, the results were not largely different. As expected, applying factors made the data points at first half (where exists most of the fluctuations) more 'focused', however the inferred models are close to one without factors, which could indicate that for model 4 and 5, applying factors will not largely affect the final convergency and the resultant posterior.

When we are trying more populations, an 'over-fitting' behaviour is observed at the trajectory of macrophage (FIGURE). In this case, the discrepancy of simulated data and observed data comes from the last 2 data point (72 hpl and 120 hpl).

Next, we performed a model selection on model 3, 4 and 5. FIGURE shows the model probabilities at different generations. From population 10 to population 20 model 4 gains advantage but after population 20 model 5 is preferred. Again, as DISCUSSED in the model comparison of model 1, 2 and 3, we considered the the violently changing model probability is related to the inference ability at certain epsilon threshold values

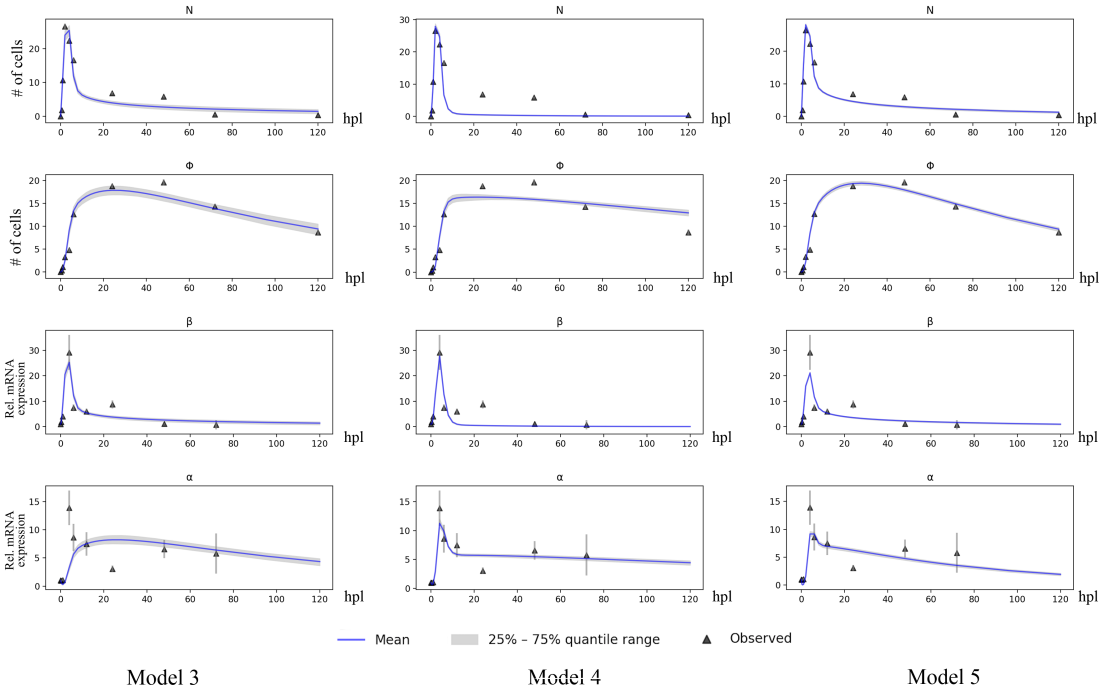

Figure 4.9: Simulated data from the last population of model 3, 4 and 5

(epsilon) and possibly the insufficient exploration of the parameter space. Given the present result, model 5 is regarded as the best model. Some further experiments tried larger population size (10,000 particles per population) and more populations (36 populations), but the resultant model did not have a significantly better fit of the observed data, although much more computation resources are used.

A inferred parameter posterior distribution of model 5 is shown in FIGURE, the estimated mean values is shown in TABLE. SOME DESCRIPTION.

4.3.3 Sensitivity of parameters

[why last PCs]

[which parameters are most sensitive]

[compared to credible intervals plot]

[what does that mean]

A quick sensitivity analysis of the dynamical systems models can be quantified using principle component analysis (PCA) [8], where the first PCs corresponds to sloppy parameters while the last PCs corresponds to stiff parameters [20]. PCA can only provide rough approximated sensitivity behaviour. As models 5 won in the model selection, a further interest is to explore the parameter-to-model sensitivity

FIGURE shows the results of PCA on the last population of inferred model5. The last PC (PC12) is mostly consists of the linear combination of a (from the equation of neutrophil) and μ_Φ (from the equation of macrophage), to which the model is most sensitive. If we conclude the last 5 PCs (FIGURE), then $v_{N\Phi}$, μ_N) (from the equation of neutrophil) and $k_{\Phi\beta}$ (from the equation of macrophage) contribute most portions together with a and μ_Φ. As a conclusion, model 5 is sensitive to changes in the above parameters, which all come from the equation of neutrophil and macrophage i.e. equations of the cells; it may suggests that given the observed data, the inferred model is more sensitive to cells' dynamic rather than cytokines' dynamics.

Also, some visualisations of the last population agreed with conclusions from PCA. Approximated posteriors of the last population plotted in FIGURE and credible ranges of parameters across populations in FIGURE shows that the 5 parameters identified by PCA is indeed 'stiff' parameters.

Chapter 5

Performance experiments

The performance experiments were designed to explore the parallel performance of ABC SMC inference framework which were used in the parameter estimation and model selection tasks. Usually ABC SMC is a time-consuming and computation intensive task and ideally executed on large clusters. The scheduling strategy, implementation details and many other factors can affect the parallel efficiency.

[MORE meanings of the performance study]

5.1 Scaling-up

First experiments are designed to illustrate the scaling-up performance. The program used here is an implementation of ABc SMC on model 5. The details of the ABC SMC settings is listed below

- Prior distribution: default to log-uniform distribution [1×10^{-6}, 50] for all the 12 parameters
- threshold schedule: median epsilon
- No factors, no adaptive distance or adaptive population applied
- Population size is 2000, with 20 generations

[HOW PYABC parallelise the sampling]

For HPC systems like Cirrus, `pyabc` uses `multiprocessing` for multi-core parallel sampling. By default if the number of cores is not specified, it will automatically read the number of available cores and use them all. Cirrus has a 36-core CPU which support hyperthreading, such that the maximal number of cores available to `multiprocessing` is 72.

The program is executed on Cirrus, using 8, 16, 24, 36, 54 and 72 cores respectively. Each run is repeated 10 times. The average execution time, required sampling numbers

are recorded. Hyperthreading is enabled when using 54 and 72 cores. The access to the node that contains computation cores is exclusive, such that the execution would not be affected by other programs of operations.

The implementation of ABC SMC in `pyabc` enables the parallelisation of sampling, which is the must time-consuming part. The rest part of the program is mostly not parallelised, e.g. database I/O and reductions operations. The sampling process involves sampling, perturbation and test of the acceptance criteria, all of which are computation-intensive.

In practice, using ABC SMC to estimate the parameters of a given model could cost up to several hundred of hours if the computational resources is limited[REF]. The performance experiment result could provide a reference that illustrate that how the efficiency changes when scaling-up or the trade-offs in computational resources' cost and their benefit.

Results

[scaling-up performance: speed-up and efficiency]

[large variation in required sampling numbers]

[possible reasons]

The recorded execution time using different number of cores is shown in FIGURE. According to that, speed-up and efficiency can be calculated. Here 8 cores is regarded as the baseline for speed-up and efficiency, as the serial version (using only one core) can take quite a long time to finish 10 repeats. When increasing number of cores, the execution time drops fast at first but the decreasing rate (dropping speed, i.e. the derivative of the execution time against time) is gradually reduced to zero; 36, 54 and 72 cores gives nearly the same average execution time and consequently gains very close speedup values (FIGURE).

Denote the number of cores used in the ABC SMC runs as variable p. The speedup curve shows a linear trend when $p \leq 36$; when $p \geq 36$ the speedup stays nearly unchanged. $p = 36$ is an inflection point, where 36 is the maximum numbers available physical cores in a compute node of Cirrus. A constant drop of efficiency is observed when increasing p; smaller p ($p < 36$) gives an efficiency higher than 65% but greater p e.g. $p = 72$ results in a low efficiency (34%).

A high variance of total required samples was observed in the scaling-up experiments: some single runs required much more samples to finish 20 populations. Moreover, we found that the execution time is not ideally proportional to the total required samples (FIGURE). A possible explain to these phenomenons can be that the median epsilon schedule results in different threshold strategies due to the randomness in the sampling process, thus results in different approximate convergency paths for different runs. Some runs that are stuck in a local optimum and require smaller target epsilon will take

huge amount of samples to move away from the local optimum. This have been observed when the distribution of posterior is plotted before and after jumping away from local modes (FIGURE and more explain). Also, time spent in each sample using sanme number of cores is not constant in our case, which might be a result of the parallel schedule; ideally a serial run using only one core will see a nearly constant time spent in every particle.

Due to the unstable sampling numbers, our interests switched to the per-particle performance, where the average sampling numbers per second (FIGURE) and average time spent in one particle (FIGURE) under different of cores (p) are plotted. The log-like average time per particle curve shows that using more cores can improve the sampling efficiency, but when p is at a high level (e.g. 36 in our case), the improvement will be less significant.

5.2 Profiling

The performance could also be analysed given a profiling report. The second experiment profiles the program to reveal the detailed time consumption for each operation and the possible bottleneck, according to which we could find the hot-spot of program and given possible suggestions on improving the performance.

In this case, profile tools `cProfile` and `yappi` is used in PyCharm IDE.

Results

[profiling results: hot-spot and possible improvements]

Chapter 6

Future works

[something not done as scheduled]

[interesting topics to look inside]

[moving to generalisations]

Chapter 7

Conclusions

[what is studied]

[to what extend]

[how good is the result]

[summary of the findings and advise]

[thanks]

Appendix A

System and software environment

A.1 ABC SMC implementation

A.1.1 Local machine

Local development machine is a Mac laptop, running on macOS 10.15.6. The environment of the development is listed in Table A.1.

Environment	Version
Operating system	macOS Catalina 10.15.6
PyCharm (IDE)	Professional 202.2
Python interpretor	3.7.0
IPython	7.12.0
Clang	6.0 (clang-600.0.57)

Table A.1: Environment on local machine

Version requirements for some critical site packages (`Python`) used in the code are listed in Table A.2.

Environment	Version
pyABC	$\simeq 0.10.3$
NumPy	$\simeq 1.18.4$
SciPy	$\simeq 1.4.1$
pandas	$\simeq 1.1.0$
matplotlib	$\simeq 3.0.1$
bokeh	$= 1.4.0$

Table A.2: Site packages on local machine

Hardware	Detail
CPU	Quad-Core Intel Core i5 8259U 2.3 GHz
Architecture	64 Bit
Hyper-Threading	Supported
Turbo Boost	Max 3.8 GHz
Memory	16 GB
Graphics card	Iris Plus Graphics 655

Table A.3: Hardwares on local machine

Local development and some runs of small size were done under the hardware listed in Table A.3. When running ABC SMC, the program can make used of 8 python process (Hyper-Threading enabled) and run under 3.8GHz (maximal) for a long time under proper cooling, proved that the program could efficiently use the computation resources for a local personal computer. It is noted that the program took no advantage of graphics card, as `multiprocessing` along can out exploit the resources inside CPU. The execution time and other performance data presented in Chapter 5 were obtained using remote machines but not local machine.

A.1.2 Remote machine

Environment	Version
Placeholder	2

Table A.4: Environment on remote machine

A.2 Data analysis

Appendix B

Data and settings

B.1 Infer-back experiments

B.1.1 Parameter values used to generate synthetic data

Parameter	Value	Unit
λ_N	2.20	$cell/h$
$\kappa_{N\beta}$	3.96	$cell/(unit \cdot h)$
μ_N	1.72	h^{-1}
$\nu_{N\Phi}$	0.220	$cell^{-1} \cdot h^{-1}$
λ_Φ	1.31	$cell/h$
$\kappa_{\Phi\beta}$	0.124	$cell/(unit \cdot h)$
μ_Φ	0.145	h^{-1}
$s_{\beta N}$	6.554	$unit/(cell \cdot h)$
$i_{\beta\Phi}$	1.71	$cell^{-1}$
μ_β	0.521	h^{-1}
$s_{\alpha\Phi}$	10.2	$unit/(cell \cdot h)$
μ_α	19.1	h^{-1}

Table B.1: Parameter values used for model 1

B.1.2 Kernel experiment: median epsilon schedule

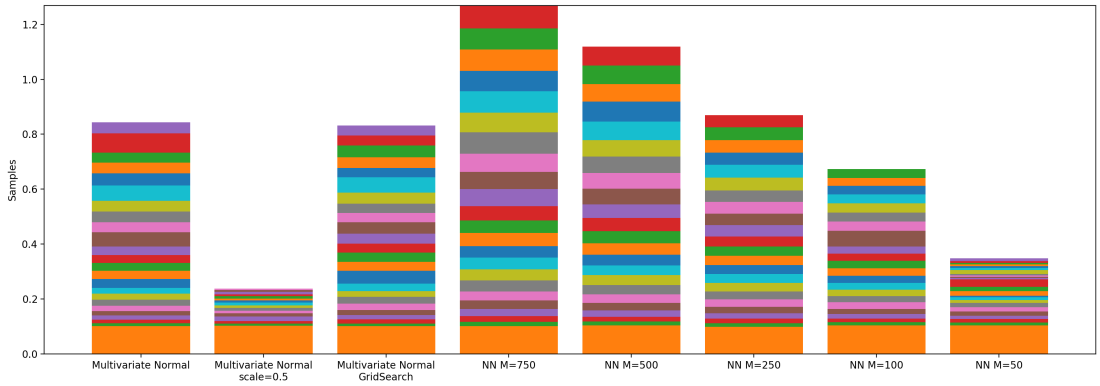

Figure B.1: Total sampling size of different kernels, using median epsilon strategy. Different color represents different generations (bottom to top: population 1 to population 20)

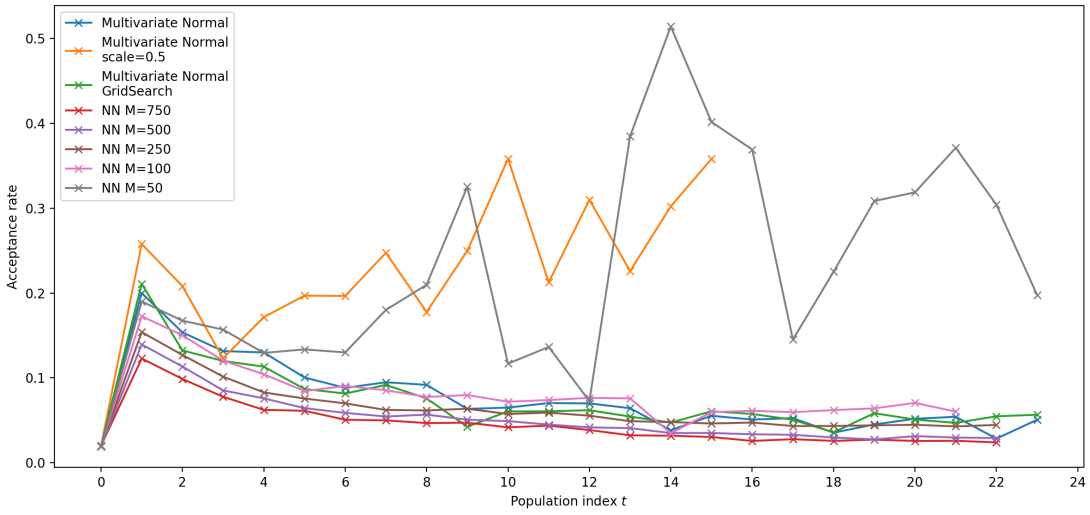

Figure B.2: Acceptance rates of different kernels, using median epsilon strategy. Each population has 2000 particles

B.2 Parameter inference and model comparison results

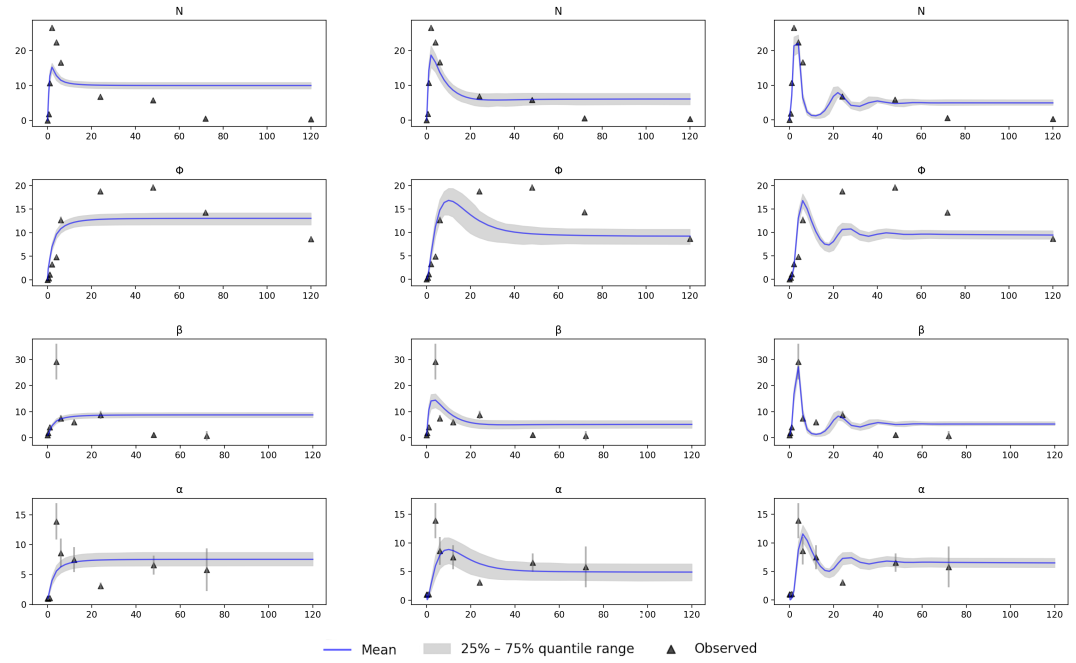

Figure B.3: Total sampling size

Bibliography

[1] T. M. Tsarouchas, D. Wehner, L. Cavone, T. Munir, M. Keatinge, M. Lambertus, A. Underhill, T. Barrett, E. Kassapis, N. Ogryzko, *et al.*, "Dynamic control of proinflammatory cytokines il-1β and tnf-α by macrophages in zebrafish spinal cord regeneration," *Nature communications*, vol. 9, no. 1, pp. 1–17, 2018.

[2] J. Liepe, P. Kirk, S. Filippi, T. Toni, C. P. Barnes, and M. P. Stumpf, "A framework for parameter estimation and model selection from experimental data in systems biology using approximate bayesian computation," *Nature protocols*, vol. 9, no. 2, pp. 439–456, 2014.

[3] T. Becker, M. F. Wullimann, C. G. Becker, R. R. Bernhardt, and M. Schachner, "Axonal regrowth after spinal cord transection in adult zebrafish," *Journal of Comparative Neurology*, vol. 377, no. 4, pp. 577–595, 1997.

[4] W. D. Anderson, H. K. Makadia, A. D. Greenhalgh, J. S. Schwaber, S. David, and R. Vadigepalli, "Computational modeling of cytokine signaling in microglia," *Molecular BioSystems*, vol. 11, no. 12, pp. 3332–3346, 2015.

[5] J. R. Dormand and P. J. Prince, "A family of embedded runge-kutta formulae," *Journal of computational and applied mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[6] W. Gilks, S. Richardson, and D. Spiegelhalter, "Markov chain monte carlo in practice chapman & hall: London," *Markov Chain Monte Carlo in practice. Chapman and Hall, London.*, 1996.

[7] S. Tavaré, D. J. Balding, R. C. Griffiths, and P. Donnelly, "Inferring coalescence times from dna sequence data," *Genetics*, vol. 145, no. 2, pp. 505–518, 1997.

[8] T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. Stumpf, "Approximate bayesian computation scheme for parameter inference and model selection in dynamical systems," *Journal of the Royal Society Interface*, vol. 6, no. 31, pp. 187–202, 2009.

[9] P. Joyce and P. Marjoram, "Approximately sufficient statistics and bayesian computation," *Statistical applications in genetics and molecular biology*, vol. 7, no. 1, 2008.

[10] M. A. Nunes and D. J. Balding, "On optimal selection of summary statistics for approximate bayesian computation," *Statistical applications in genetics and molecular biology*, vol. 9, no. 1, 2010.

[11] A. Minter and R. Retkute, "Approximate bayesian computation for infectious disease modelling," *Epidemics*, vol. 29, p. 100368, 2019.

[12] D. Silk, S. Filippi, and M. P. Stumpf, "Optimizing threshold-schedules for approximate bayesian computation sequential monte carlo samplers: applications to molecular systems," *arXiv preprint arXiv:1210.3296*, 2012.

[13] E. Klinger and J. Hasenauer, "A scheme for adaptive selection of population sizes in approximate bayesian computation-sequential monte carlo," in *International Conference on Computational Methods in Systems Biology*, pp. 128–144, Springer, 2017.

[14] D. Prangle *et al.*, "Adapting the abc distance function," *Bayesian Analysis*, vol. 12, no. 1, pp. 289–309, 2017.

[15] S. Filippi, C. P. Barnes, J. Cornebise, and M. P. Stumpf, "On optimality of kernels for approximate bayesian computation using sequential monte carlo," *Statistical applications in genetics and molecular biology*, vol. 12, no. 1, pp. 87–107, 2013.

[16] T. Toni and M. P. Stumpf, "Simulation-based model selection for dynamical systems in systems and population biology," *Bioinformatics*, vol. 26, no. 1, pp. 104–110, 2010.

[17] A. C. Daly, J. Cooper, D. J. Gavaghan, and C. Holmes, "Comparing two sequential monte carlo samplers for exact and approximate bayesian inference on biological models," *Journal of The Royal Society Interface*, vol. 14, no. 134, p. 20170340, 2017.

[18] A. Weyant, C. Schafer, and W. M. Wood-Vasey, "Likelihood-free cosmological inference with type ia supernovae: approximate bayesian computation for a complete treatment of uncertainty," *The Astrophysical Journal*, vol. 764, no. 2, p. 116, 2013.

[19] E. Klinger, D. Rickert, and J. Hasenauer, "pyabc: distributed, likelihood-free inference," *Bioinformatics*, vol. 34, no. 20, pp. 3591–3593, 2018.

[20] R. N. Gutenkunst, J. J. Waterfall, F. P. Casey, K. S. Brown, C. R. Myers, and J. P. Sethna, "Universally sloppy parameter sensitivities in systems biology models," *PLoS Comput Biol*, vol. 3, no. 10, p. e189, 2007.