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

Chaolin Han

July 17, 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

Acknowledgements

This template is a slightly modified version of the one developed by Prof. Charles Duncan for MSc students in the Dept. of Meteorology. His acknowledgement follows:

This template has been produced with help from many former students who have shown different ways of doing things. Please make suggestions for further improvements.

Chapter 1

Introduction

This should contain a description of your project and the problem you are trying to solve. Where appropriate you should also include references to work which has already been done on your topic and anything else which lets you set your work in context.

One of the things you will need to do is to ensure that you have a suitable list of references. To do this you should see [1] or some other suitable reference. Note the format of the citation used here is the style favoured in this department. Here is another reference [2] for good measure.

You will also want to make sure you have no spelling or grammatical mistakes. To help idwentify spelling mistukes you caan use the commands *ispell* or *spell*. See the appropriate manual pages. Remember that spelling mistakes are not the only errors which can occur. Spelling checkers will not find errors which are, in fact, valid words such as *there* for *their*, nor will they find repeated words which sometimes occur if your concentration is broken when typing. **There is no substitute for thorough proof reading!**

[background and motivation]

[what is done]

Chapter 2

Background

2.1 Zebrafish spinal cord regeneration

[describe the process and how cells and cytokines are involved]

2.2 Mathematical modelling

[how this can be modelled as dynamic systems]

[general topics of mathematical models: how to build model according to interactions; how to calibrate/ parameterise the model; how to solve the model; dynamics; how to evaluate the model]

2.3 Bayesian inference

[how to parameterise the model]

[how the infer the parameter given data]

[likelihood-free inference and information about ABC SMC]

[model comparison, ABC SMC for model comparison]

2.4 Software tools

[existing tools and software for this task]

[workflow and developing]

Chapter 3

Mathematical modelling

Mathematical models can describe different kinds of dynamic system, and can be used as a guide to prediction and analysis. An ideal model in this case, can represent reasonable interactions/effects between cells and cytokines and recover the observed trend against time.

Models for the regeneration process is in the form of ordinary differential equations, specifically the time differential form. Terms in the ODE are mostly explainable and corresponds interaction paths.

3.1 Observed data

Our models is built regarding the existing experiments data form Tsarouchas et al.[3]. The measurement includes 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 [3], 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 (il-1β) and α (tnf-α). N and Φ is of the unit 'number of cells', β and α is of the unit 'relative concentration'.

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, assuming that measured data is Gaussian-like distributed. The distribution of the measured data points is plotted and examined to see if the measurement mean could represent the distribution. The result is that at most time points the measurement values are Gaussian-distributed, 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 measures and thus is still used as the target summarised observed data. A plot of the mean of the four variables is shown in Figure 3.1

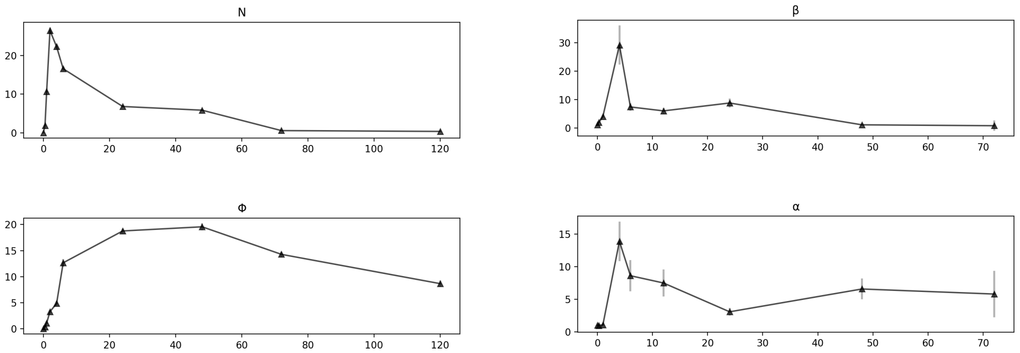


Figure 3.1: Mean of the observed data for neutrophil N, macrophage Φ, il-1β and tnf-α, from experiment results of [3]. Error bars indicate standard error of mean

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 be more close to the true process, recover more features of the observed data or test our hypothesis.

All these models assumes the 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 parameter estimation. Interactions or effects from other cells or cytokines is not considered as there might not be corresponding data.

3.2.1 Basic model

A preliminary model is proposed according to [3] and used to build and test the code. A interaction map illustrate the model is shown in Figure 3.2. This model is a simplification a the process described in [3]. 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 $\aleph$ i.e. relative mRNA expression as 'unit' for simplicity.

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

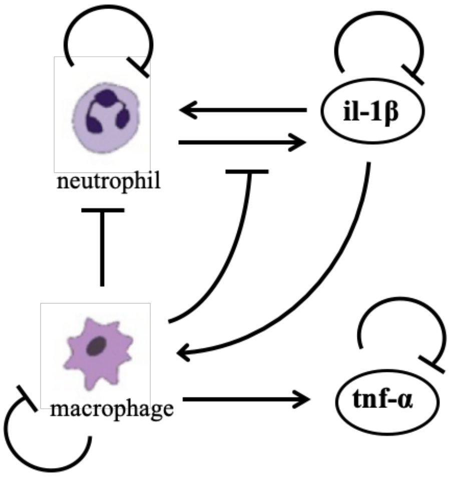


Figure 3.2: Interactions modelled in the basic model (model 1) based on Tsarouchas et al.[3]. Lines ended with arrow represent promoting effect, lines ended with T-connectors represent inhibition

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

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)

3.2.2 Alternative models

Model 2 and model 3 As the observed data indicates, 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 term λ cannot be constant, thus a exponentially decay λ term is introduced and model 2 is proposed as Eqn. 3.2. The inhibition of il-1β being produced by neutrophil, i.e. $i_{\beta\Phi}$ is considered to be ignored, 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 has one extra parameter a which is a coefficient in the exponentially decay 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 regarded as model 2 with parameter $i_{\beta\Phi} = 0$. For these three models, model 1 is a naive one that is proposed at very first time and used as a ‘template’ to build and test parameter inference framework. As the implementation is successful, model 2 and 3 is proposed. 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 are close to steady states for neutrophil and macrophage. model 3 is a small simplification of model 2 and theoretically less general than model 2.

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

According to that, attempts are tried to introduce more interactions within the dynamic system considering the both biological and mathematical context. Extra promoting effect to the expression of tnf-α is considered, by either add a phenomenological term (which means the same effect as directly promoting but the underlying mechanism is unclean) or add 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

[feasible or not; goodness of fit; features of these models]

[bayes factor for model selection]

3.2.4 Limitations

[hypothesis]

[available data]

[model misspecification]

Chapter 4

Implementations and Experiments

This chapter focuses on the implementation of the ABC SMC on the proposed models. Estimation of parameters in the proposed models is the main task of this section. ABC SMC-based models selection is another task that can be easily implemented with the parameter estimation framework.

The build and test is under `Python` environment with `pyABC`[4]. Some other code e.g. shell script and `R` code are also used to perform the experiments and analyse the results. Tested software and system environment can be found at Appendix A. The build and test is performed on local computers and HPC clusters available within EPCC.

4.1 Parameter estimation

[implementation details and options/ settings that are affecting the ABC]

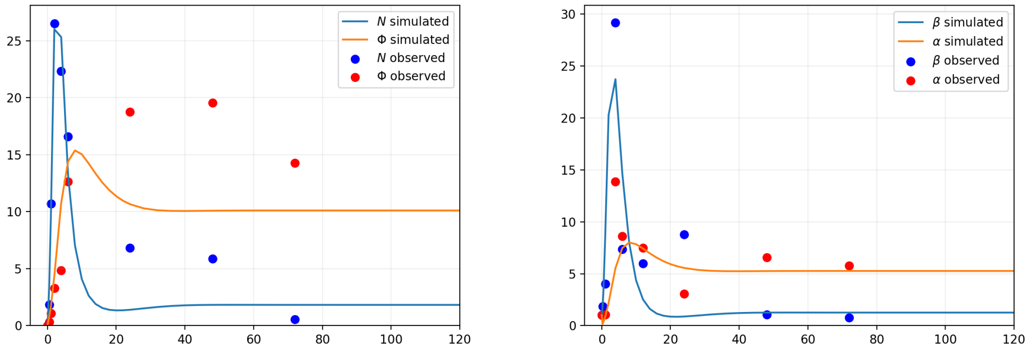
As the key focus of this project is on the parameter estimations of dynamical system, the ability of inference on the proposed model is firstly examined before actual inference using the experimental observed data.

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.

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.

4.1.1 ABC SMC and model hyperparameters experiments

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



(a)

(b)

Figure 4.1: Synthetic data generated with known parameter values (line plot), compared to experimental data (scatter plot)

[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`.

Among them the following topics are studied using synthetic data.

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

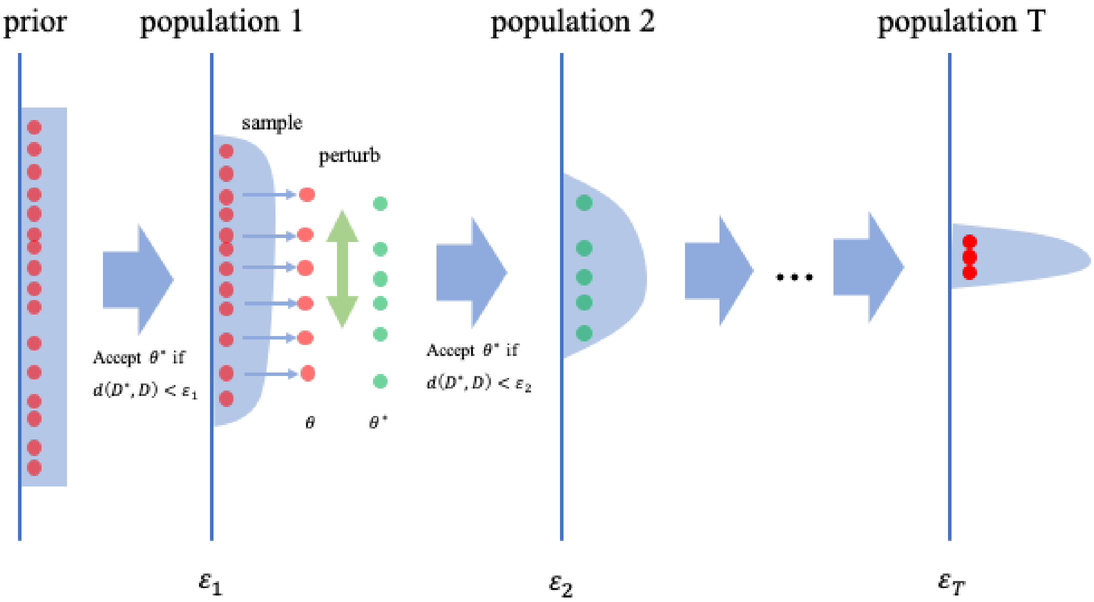


Figure 4.2: ABC SMC sampling process

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 [5].

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 [5]. It has been proved to perform well in several dynamic system models [6, 7, 8] 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 4.3, 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 4.4, 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 [5]. '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 [5]. One obvious example is presented in Figure 4.3, 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)

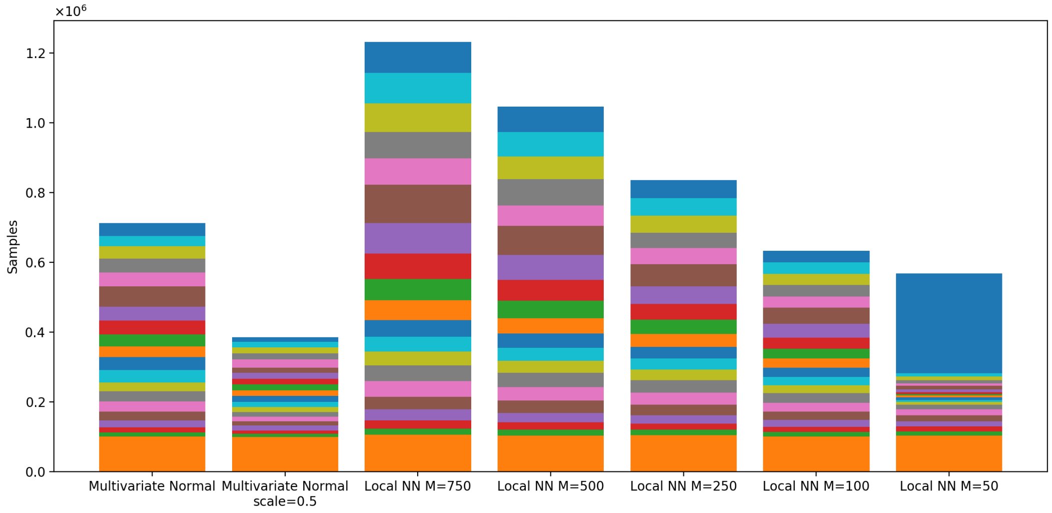


Figure 4.3: Total required samples of different kernels. Different color represents different generations (bottom to top: generation 1 to generation 20)

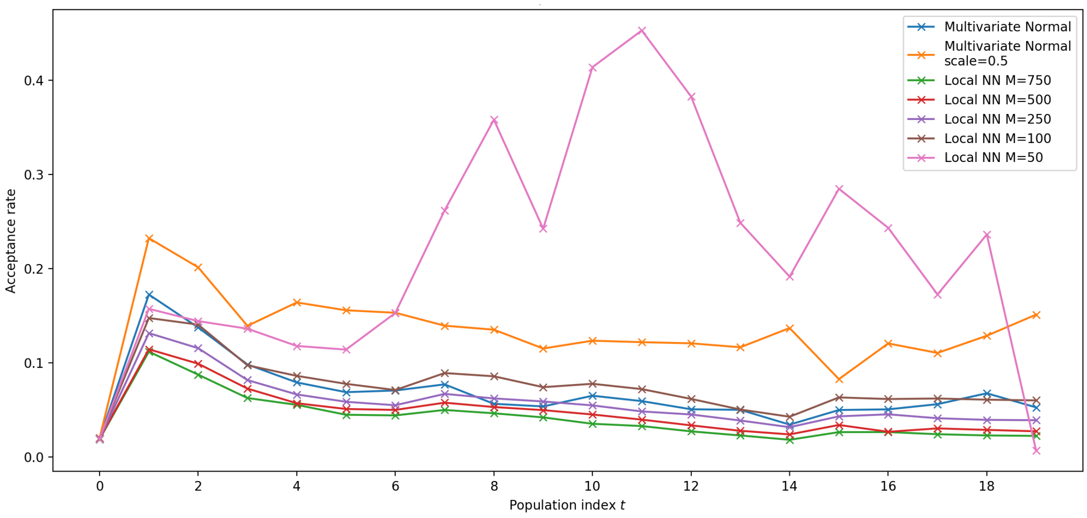


Figure 4.4: Acceptance rates of different kernels

and multivariate normal with scaling. Also multivariate M-NN is worth trying, although it is not built in `pyABC` and requires additional implementations.

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)

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

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 [9]

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

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 ans (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 [10] 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

[picture here]

[what is the result]

Conclusion

Data size, prior range, population size and generations

[experiments plan]

Experiments results

Conclusion

4.1.2 Implementations and code

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

[how the code is developed and built]

4.2 Model comparison

[comparison of model 1, 2 and 3]

[further comparison of model 3, 4 and 5]

4.3 Performance experiments

The performance experiments are designed to explore the parallel performance of ABC SMC implementations. Usually ABC SMC is a time-consuming and computation intensive task and usually is executed on large clusters. The scheduling strategy, implementation details and many other factors can affect the parallel efficiency.

[MORE meanings of the performance study]

4.3.1 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, 12, 16, 24, 36, 54 and 72 cores respectively. Each run is repeated 5 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.

4.3.2 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.

Chapter 5

Results and Discussions

[WHAT results is outputted and WHAT topics are discussed]

5.1 ABC SMC results

[inferred parameters: joint distribution, estimated values and simulated trajectory for each models; sensitivity; features of the inferred model]

[model selections result; bayes factor]

5.2 Performance experiments

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

[profiling results: hot-spot and possible improvements]

5.3 Discussions

[goodness of fit (evaluation); preferred models and effects of modification to basic model]

[efficiency and trade-offs in scaling-up]

[POSSIBLE: compare to exact inference; compared to other implementations]

[limitations]

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.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	2.1989	$cell/h$
$\kappa_{N\beta}$	3.9627	$cell/(unit\cdot h)$
μ_N	1.7219	h^{-1}
$\nu_{N\Phi}$	0.2195	$cell^{-1}\cdot h^{-1}$
λ_Φ	1.3146	$cell/h$
$\kappa_{\Phi\beta}$	0.1235	$cell/(unit\cdot h)$
μ_Φ	0.1454	h^{-1}
$s_{\beta N}$	6.5536	$unit/(cell\cdot h)$
$i_{\beta\Phi}$	1.7062	$cell^{-1}$
μ_β	0.5212	h^{-1}
$s_{\alpha\Phi}$	10.2416	$unit/(cell\cdot h)$
μ_α	19.6642	h^{-1}

Table B.1: Parameter values

B.2 Kernel experiment: median epsilon schedule

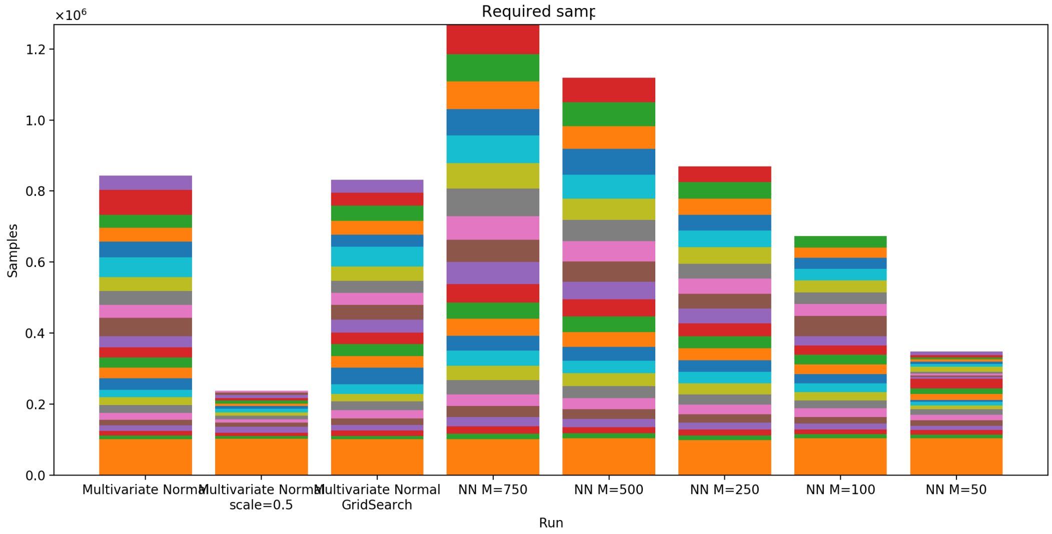


Figure B.1: Total sampling size

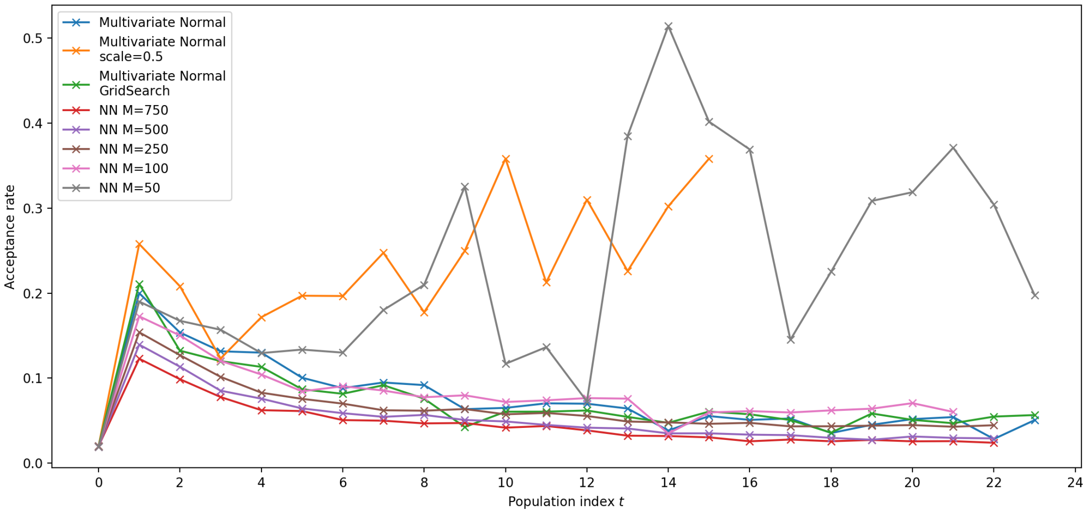


Figure B.2: Acceptance rates

Bibliography

[1] L.Lamport. *1986 Latex User's Guide and Reference Manual.* Addison Wesley. pp242.

[2] F.Bloggs. *1993 Latex Users do it in Environments* Int. Journal of Silly Findings. pp 23-29.

[3] Tsarouchas, T.M., Wehner, D., Cavone, L., Munir, T., Keatinge, M., Lambertus, M., Underhill, A., Barrett, T., Kassapis, E., Ogryzko, N. and Feng, Y., 2018. Dynamic control of proinflammatory cytokines Il-1β and Tnf-α by macrophages in zebrafish spinal cord regeneration. Nature communications, 9(1), pp.1-17.

[4] Klinger, E., Rickert, D. and Hasenauer, J., 2018. pyABC: distributed, likelihood-free inference. Bioinformatics, 34(20), pp.3591-3593.

[5] Filippi, S., Barnes, C. P., Cornebise, J., and Stumpf, M..H. (2013). On optimality of kernels for approximate Bayesian computation using sequential Monte Carlo. Statistical Applications in Genetics and Molecular Biology 12, 1, 87-107.

[6] Liepe, J., Kirk, P., Filippi, S., Toni, T., Barnes, C.P., and Stumpf, M.P.H., 2014. A framework for parameter estimation and model selection from experimental data in systems biology using approximate Bayesian computation. Nature Protocols. 9 (2) pp. 439–456.

[7] Daly AC, Cooper J, GavaghanDJ, Holmes C. 2017. Comparing two sequential-Monte Carlo samplers for exact andapproximate Bayesian inference on biologicalmodels.J. R. Soc. Interface14: 20170340.

[8] Minter, A. and Retkute, R., 2019. Approximate Bayesian Computation for infectious disease modelling. Epidemics, 29, p.100368.

[9] Prangle, D., 2017. Adapting the ABC distance function. Bayesian Analysis, 12(1), pp.289-309.

[10] Klinger, E. and Hasenauer, J., 2017, September. 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, Cham.