1 Problem and Project goals

Although mathematical modeling for biological systems is not a new topic, with the advent of microarray experiments that enabled us to take measurements of the expression levels of a big number of genes at different time points, there has been a big grow in the attempts to mathematically capture the behaviour of such systems. These measurements led to the discovery of common patterns and networks in cell behaviour. Increasingly complex models have been employed to describe the behaviour of such networks. The use of models in this case is beneficial as they provide a great intuition and increased understanding into the dynamic nature of such systems and can even uncover and predict new behaviour which can lead to experimental confirmation. These models are often a system of Ordinary Differential Equations (ODEs) which are the natural way to capture dynamic and changing behaviour. As the complexity of these models grew the problem of estimating the parameters of these models, in this case primarily rate constants and kinetic parameters, became a major task for modelers as those parameters are often infeasible to experimentally measure. This fact along with the noisy and often scarce data made people resort to manual search of parameter space and/or mathematical techniques to assist in the task.

The last few years and with the rapid technological advancements, the computational cost associated with such techniques has fallen and there have been some attempts to automate this process with the production of computational tools that tackle the parameter estimation problem using mathematical methods borrowed from other fields. Unlike other fields however the requirements of models of biological systems are greater than just a simple reproduction of experimental data. These systems have qualitative features that are often very important and their behaviour changes as they evolve or respond to stimuli. If the proposed model does not take into account such behaviour then the model does not capture the precise behaviour and although it can sometimes be useful, it is incomplete. The main goal of this project is the production of a computational tool that systematically attacks the problem of parameter estimation by automating the estimation process using mathematical methods. This tool will take into account the dynamic behaviour of biological systems and use these criteria in the parameter selection process. This new aspect will make the tool more powerful and the models produced more complete and closer to the true behaviour as exhibited in nature. I believe that mathematical models that capture all aspects of these systems will increase the confidence in modeling and the use of mathematical tools in biological research just like they are being used successfully in other scientific disciplines such as Physics.

2 Background

Recent advances in recombinant DNA and microarray technology have led to greater understanding of the networks that govern different cell processes. In these networks a number of different genes and proteins interact to carry out a biochemical process. Examples of well studied networks are metabolic pathways and circadian oscillators[1]. The traditional way to describe these networks is by means of diagrams that show the signals and interaction between entities (genes, proteins). As these systems grow in complexity the diagrammatic way of describing the system is increasingly proved to be a poor way to do it as it does not give any intuition into the inner workings of the system nor does it capture the dynamic nature that such systems exhibit. Moreover as the complexity of the network grows it become increasingly difficult to predict the future behaviour of the system especially quantitatively. The advantages and the need for using quantitative mathematical models to describe such systems has been recognised and the perception of their importance is growing[10]. Also the number of attempts to model systems that are thought to be well understood like circadian oscillators is ever increasing[2, 15]. Chemical kinetic laws such as the Law of mass-action or the Michaelis-Menten kinetics can be easily translated to simple systems of ODEs to construct simple components or commonly found patterns in biochemical networks such as switches, buzzers, blinkers[22] the inspiration for the decomposition into electronic-like components coming from the resemblance that these networks show at the systems level with electronic circuits. By combining these simple motifs more complex networks can be built. Although the underlying behaviour at the molecular level is highly stochastic the models are deterministic systems of ODEs because the underlying stochasticity gives rise to deterministic behaviour at the systems level.

A typical systems can be described with a set of ODEs like so: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t, \theta)$. Vector $\mathbf{x}$ is usually functions describing the change of the concentration of a substance (like mRNA, protein etc.) over time with θ being the parameter vector. And although the construction of such models can be done by using simple laws (law of mass-action, Michaelis-Menten kinetics) or from just knowledge of the behaviour of the system in question, finding the parameters θ that will make these systems behave according to observations is an equally if not more difficult task since experimentally measuring these parameters like rate constants can be next to impossible. The task therefore becomes from a set of observations over time $\mathbf{Y} = \{\mathbf{y}_{t_1}, \mathbf{y}_{t_1}, \ldots, \mathbf{y}_{t_n}\}$ to find the parameter vector θ^* such that the distance between the simulated dataset from $f(\mathbf{x}, t, \theta^*)$ and the actual dataset $\mathbf{Y}$ is minimum. This is a common formulation of the problem of the inverse problem which reduces the problem to an optimisation problem which is a well studied area[6]. Common local and global optimisation techniques have been used in this area as well[16].

Another way to look at the problem is from a statistical/probabilistic point of view. The goal in this case is to increase the likelihood $L(\theta) = f(\mathbf{Y}|\theta)$ where $\mathbf{Y}$ are observations and θ is the parameter values[5]. Then the problem becomes to find the values of parameters θ that maximise the likelihood function $L(\theta)$. Maximum likelihood estimation is again a well studied area and techniques used in other areas have been employed in this particular case as well.Other statistical methods have also been employed to tackle this problem, like Bayesian inference techniques. The traditional Bayesian inference in this case becomes $\pi(\theta|Y) = f(\mathbf{Y}|\theta)\pi(\theta)$. [21]. The likelihood given here is not computable so that gave rise to new set of techniques that simulated the Bayesian inference by repeatedly sampling from some distribution accepting only samples θ^* such that the simulated dataset is close to the original dataset Y. That way the output is sample from the posterior $\pi(\theta|\mathbf{Y})$. Generating a sample to simulate an unknown or difficult distribution is not a new technique as it is the main theme in traditional Monte Carlo techniques such as Metropolis-Hastings[23]. Because computer simulation is more feasible nowadays these techniques commonly referred as Approximate Bayesian Computation techniques (approximate because they do not really do the Bayesian inference computation) have gained popularity. A range of different algorithms have been employed, from simple rejection samplers[20] to Markov Chain Monte Carlo methods in the spirit of Metropolis-Hastings[14] and Sequential Monte Carlo methods which use a series of distributions that at every stage are closer to the true posterior[21].

Tools that assist in the modeling process have been created in recent years(XPP-Auto, GRIND). These tools are widely used because they allow modelers to get an intuition and greater understanding of the model in question usually by graphical means such as plotting bifurcation diagrams and allowing them to see the changes in the dynamics of the system as parameters change. And although they help in parameter estimation they do not address the problem directly. One other attempt that has been made that addresses the problem directly is the ABC-Sysbio[11]. This tool does automatic parameter estimation using Sequential Monte Carlo method.

All these methods and these tools have as their main aim to find the set of parameters that reconstructs the experimental data. They do not take into account other features of the system in question that the model must capture to correctly capture its dynamics. However systems have a dynamic nature that has to be taken into account. They respond to signal in a certain way, they evolve. For example circadian oscillators that exist in many tissues get synchronised with the main clock in the SCM of the brain by getting an entrainment signal from it. When perturbed in that way their signal changes in a specific way as described by their Phase Response curves[19]. Other studies have also been made in using Fourier Analysis which is a widely used technique in engineering applications to model selection by comparing the Fourier Transform as produced by different models with the Fourier Transform of the experimental data instead of actually comparing the data points of simulated versus real dataset[9]. That way the model selection process captures other qualitative features of the model that are not directly observable from the data. Techniques from the rich topic of non-linear dynamical systems have been employed to look at the changes of dynamics of biological systems like how the inference process gets affected by the presence of Hopf bifurcations[8].

Another aspect of dynamic behaviour of such systems is captured by the changes in them when they go through the process of evolution. In particular gene duplication gene duplication is believed to play a major role in molecular evolution. The new duplicated gene does not have selective pressure making its genetic mutations highly likely event , since it does not have effect on the function of the organism, which could lead to the development of new function(neofunctionalisation). For genes with multiple functions the duplication can also allow both genes to freely mutate as long as the lost function of one of the duplicated genes is picked by the others(subfunctionalisation). The rate at which this evolutionary event happens it too high to be ignored[12]. This is the basis of some recent work trying to capture this evolutionary behaviour in mathematical models of simple biological systems[3]. We believe that because of the importance and frequency of these events, they should play a role and affect the process of parameter choice because the resulting model will incorporate and account for evolutionary events (gene duplication) making it more powerful.

3 Work

The language of choice for the implementation of the tool is Python. It is a highly extensible language leading to the development of a hige number of third party libraries for many tasks. There exist many scientific packages in Python like scipy[7], numpy with a large and lively community of users. It is also a free language therefore easily available making it a good choice for the development of tools that can be accessible to large user-base. Its use in systems biology is also increasing[17, 18].

Three algorithms were implemented for the automatic parameter estimation. All of them are in the family of Approximate Bayesian Computation (ABC) algorithms. These kind of algorithms are becoming increasingly popular because of the computational efficiency of modern simulation techniques. The main theme behind these algorithms is the replacement of the computation of the likelihood function $L(\theta)$ as present in the traditional formulation of Bayesian inference problem with a comparison between observed and simulated data. The parameter estimation problem in the conventional Bayesian inference formulation becomes $\pi(Y|\theta) = L(\theta)\pi(\theta)$, where θ is the parameter vector to be estimated, $\pi(\theta|\mathbf{Y})$ is the posterior probability distribution of observed data $\mathbf{Y} = \{\mathbf{y}_{t_1}, \mathbf{y}_{t_1}, \dots, \mathbf{y}_{t_n}\}$ with parameter θ and $L(\theta) = f(\mathbf{Y}|\theta)$ the likelihood function. The common form of ABC algorithms is to avoid these calculations and instead simulate from a proposal distribution and accept parameter values that give simulated data close to the observed data:

1. Sample a parameter vector θ^* from a proposal distribution $\pi(\theta)$
2. Simulate a dataset $\mathbf{y}^*$ from the model with parameters θ^*
3. Compare the simulated dataset $\mathbf{y}^*$ and observations $\mathbf{Y}$ according to some distance function d. If $d(\mathbf{y}^*, \mathbf{Y}) < \epsilon$ where ϵ is the error tolerance of accepted solutions, then accept $\theta*$ otherwise reject.

All the accepted samples form a sample from the posterior simulating it. This is the common form of all the algorithms in this family and they differ by how they choose their proposal distribution $\pi(\theta)$ and how they adapt to reduce the number of simulation steps needed to accept a reasonable number of samples in order to simulate the goal posterior distribution reliably.

3.1 Simple Rejection Sampler

The simpler ABC algorithm is a simple rejection sampler[20] which samples from a prior distribution not taking into account correct guesses and 'blindly' continues to draw samples from the prior:

1. Sample a parameter θ^* from prior distribution $\pi(\theta)$.
2. Simulate a dataset $\mathbf{y}^*$ from model with parameters θ^*.
3. If $d(\mathbf{y}^*, \mathbf{Y}) < \epsilon$ accept θ^*, otherwise reject.

The distance function used is a simple Euclidean distance metric defined as usual by: $\sqrt{\sum_{i=0}^{n}(y_i^* - Y_i)^2}$ and where $\mathbf{y}^*$ is the simulated dataset and $\mathbf{Y}$ is the observed dataset.

Obviously the acceptance rate for this algorithm is really low since it does not take into account previous good guesses to inform the search and keeps 'blindly' jumping in parameter space. This low acceptance rate means a high number of simulation steps are needed to get a reasonable number of accepted samples and although the simulation does not have a high computational cost the number of steps needed make this algorithm a poor choice and it was just implemented as a reference point to see the improvements with the more sophisticated algorithms. For precise results see next section.

3.2 Markov Chain Monte Carlo

An improvement to the random search of the simple rejection sampler is offered by a Markov Chain Monte Carlo method in the spirit of the traditional Metropolis-Hastings algorithm but slightly modified to fit this problem[14]. The intuition behind this algorithm is that you use previous good guesses as a basis for the next sampling thereby creating a Markov Chain with stationary distribution the goal posterior:

1. Initialise $\theta_i, i = 0$.
2. Sample a candidate parameter vector θ^* from a proposal distribution $q(\theta^*|\theta_i)$.
3. Simulate a dataset $\mathbf{y}^*$ from model with parameters θ^*.
4. If $d(\mathbf{y}^*, \mathbf{Y}) < \epsilon$ jump to new value θ^* - making it the next value in the chain, $\theta_{i+1} = \theta^*$- with a probability given by:
$$a = min\left(1, \frac{\pi(\theta^*)q(\theta_i|\theta^*)}{\pi(\theta_i q(\theta^*|\theta_i)}\right)$$
5. Set i=i+1 and continue from 2.

One of the decisions that was taken was to use an adaptive Gaussian distribution as a proposal distribution. The proposal Gaussian from which we sample has big variance at first so that it will make big jumps in the parameter space. Once a good sample is found the variance of the proposal distribution is decreased to stay in the area. But that has the problem of getting stuck at local optimum areas for a long period of time so a rejection count is employed. If in the close vicinity we keep rejecting then the variance of the proposal distribution is increased to move out of that area. That way areas with many accepted samples are oversampled in the final population. It should be noted that univariate Gaussians were used for the proposal distributions, with each parameter in the parameter vector θ coming from a different proposal distribution. The relations between the parameters are however captured in the fact that when a sample is accepted it is accepted as whole. Despite the attempts to avoid the problem of getting stuck at local sub-optima, this algorithm is still prone to doing exactly that. One possible improvement is to change it to the original Metropolis-Hastings and therefore accepting (with low probability) samples that do not come close to the observed data. That way sometimes jumps will be made to other areas which can result in better exploration of the parameter space. This can also give a resulting population with more than one candidate value and then other criteria can be used to evaluate the fitness of the model with both set of candidate parameters.

3.3 Sequential Monte Carlo

Another variation of the ABC form is the Sequential Monte Carlo approach[21]. In this algorithm instead of having a single ϵ value as tolerance we have an ϵ schedule of a series of decreasing values. We pass a population of parameter values sampled from a prior $\pi(\theta)$ from a series of distributions with the decreasing ϵ schedule ensuring that the intermediate distributions gradually go towards the posterior.

1. Initialise the ϵ-sequence $\epsilon_1, \ldots, \epsilon_T$
2. If $t = 0$, sample $\theta^{*}*$ from prior $\pi(\theta)$
   Else, sample θ^* from previous population θ^i_{t-1} with weights w_{t-1} and perturb it with a perturbation kernel $K_t(\theta|\theta^*)$ to get particle θ^{**}.
3. Simulate dataset $\mathbf{y}^*$ from model with parameters θ^{**}.
4. if $d(\mathbf{y}^*, \mathbf{Y}) > \epsilon_t$ go back to 2, otherwise accept particle -set $\theta^{(i)}_t = \theta^{**}$- and calculate its associated weight by:

$$w^{(i)}_t = \begin{cases} 1, & \text{if } t = 0 \\ \frac{\pi(\theta^{(i)}_t)}{\sum_{j-1}^{N} w^{(j)}_{t-1} K_t(\theta^{(j)}_{t-1}, \theta^{(i)}_t}, & \text{if } t > 0 \end{cases}$$

5. Continue for all $\epsilon \in \{\epsilon_1, \ldots, \epsilon_T\}$

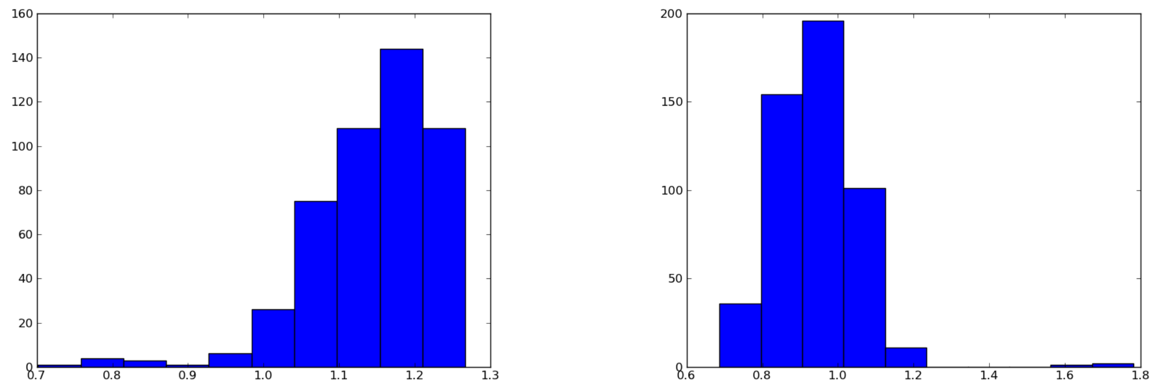
Here the success of the algorithm both in computational cost and quality of the solution depends on the choice of the ϵ schedule and the perturbation kernel K_t. If the difference between two consecutive tolerances ϵ_t and ϵ_{t+1} is very small the corresponding intermediate distributions $p_{\epsilon_t}(\theta|\mathbf{Y})$ and $p_{\epsilon_{t+1}}(\theta|\mathbf{Y})$ are very similar and although the acceptance rate in the corresponding simulations will be high, the $|E|$ needs to be big. If the difference is small the acceptance rate will be lower but the $|E|$ needs to be smaller to reach the true tolerance rate[5]. We tuned ϵ-schedule by hand but there are some proposals on an adaptive choice[4].

A perturbation kernel K_t with small variance will lead to a high acceptance rate between iterations if the corresponding tolerances have small difference. A kernel with higher variance will lead to lower acceptance rate and more steps to generate the required number of particles but it will explore the parameter space better[5]. In our implementation two different perturbation kernels were tried: a multivariate Gaussian with the covariance of the previous population and separate univariate Gaussian distribution for each parameter with the variance of the previous population.

3.4 Results

The three implemented algorithms were tested with Lotka-Volterra[13], a simple predator-prey model consisting of a pair of non-linear first order differential equations :

$$\begin{aligned} \dot{x}_1 &= a x_1 - x_1 x_2 \\ \dot{x}_2 &= b x_1 x_2 - x_2 \end{aligned}$$



(a) Posterior distribution obtained for parameter a(b) Posterior distribution obtained for parameter b

Figure 1: Posterior distribution inferred for parameters a and b with MCMC method(500 particles)

The model has two parameter (a, b) which were set to $(a, b) = 1$ and a dataset was created from 8 data-points chosen at random in the interval $[0, 15]$. Then Gaussian noise $\mathcal{N}(0, (0.5)^2)$ was added to the dataset. The distance between the original set and noisy dataset averaged over some runs is $\approx$4 so the tolerance rate ϵ could not be < 4 so a choice of $\epsilon = 4.5$ was deemed appropriate.

First, we apply the simple rejection sampler algorithm with $\epsilon = 4.5$. The prior distributions for parameters a and b are taken to be uniform $a, b = \mathcal{U}(-10, 10)$. As expected the acceptance rate is really low. In order to accept 200 particles, 247000 data generation steps are required which gives an acceptance rate of 8×10^{-4}.

Then the Markov Chain Monte Carlo method with the adaptive Gaussian as described in the previous section is applied with the same tolerance rate $\epsilon = 4.5$. In order to obtain 500 particles 20000 to 40000 data generation steps were needed which is a big improvement from the simple rejection algorithm. The inferred posterior distributions for model parameters a and b are shown in Figure 1.

Next, we apply the SMC algorithm with a manually tuned schedule $\epsilon = (30.0, 16.0, 6.0, 5.0, 4.3)$. The prior distributions for model parameter a and b are taken to be uniform $a, b = \mathcal{U}(-10, 10)$. The perturbation kernels K_t are taken to be Gaussian centered around the previous population and with standard deviation equal to that of the previous population. The results are shown in Figure 2. The evolution of the intermediate distributions for both parameters to the inferred distributions at $t = 5$ is shown in Figure 1. It can be seen that the parameters are well inferred a: [mean:1.021, median:1.016] b: [mean:1.08, median:1.20]. The number of data generation steps needed for every intermediate population generated are shown in Table 2. The numbers verify the claims in the previous section: the big difference between ϵ_1 and ϵ_2 is reflected in the big number of data generation steps needed to go from $t = 1$ to $t = 2$. On the other hand very few data generation steps are needed between $t = 4$ and $t = 5$ due to the small difference between ϵ_4 and ϵ_5.

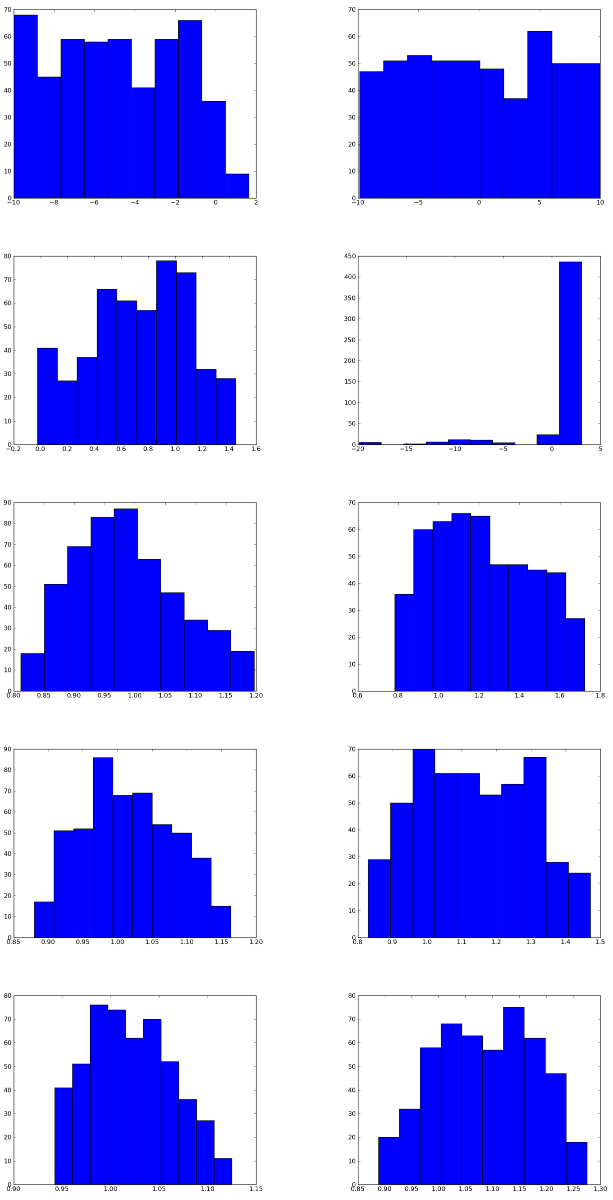


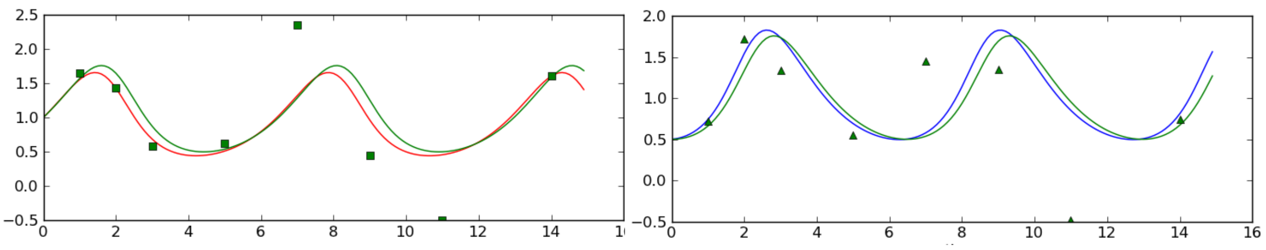
Table 1: Evolution of intermediate distribution for SMC method and $t = 5$.

4 Future Work

As stated in a previous section the main goals of the project were to create a tool that does automatic parameter estimation in dynamic biological systems and augment that by taking into account in the process other properties and characteristics of the systems such as evolution to make the inference process more powerful and the resulting models more complete. So far only the first part of task has been addressed, namely the automatic parameter estimation problem. The implemented algorithms could be refined by making better choices for the settings like using an adaptive ϵ-schedule for the SMC algorithm

population	1	2	3	4	5
data generation steps	926	13 563	50 135	51 214	52 411

Table 2: Cumulative data generation steps needed for each population.



(a) x_1 (b) x_2

Figure 2: Results for SMC method(500 particles). The green graphs are the idealised functions. The green triangles and squares are the noisy dataset used and the blue and red graphs are the inferred functions.

and switch to Multivariate Gaussian distribution to account for correlations between parameters. The main bulk of the future work will however fall on the second task, incorporating the dynamic nature of these systems in the parameter inference process. One area of focus is accounting for evolutionary events like gene duplication. This could be a post-processing step which will act as model selection step after getting parameter candidates from the ABC procedures as described above. In order to get parameter candidates the algorithms will have to be tweaked because in their current state they give a population centered around a single value. One way to do it could be to employ some form of genetic algorithm of heuristic process to explore other areas of parameter space to get more parameters sets with high likelihood. Then the model selection among models with the different candidate parameter sets will be done according to how well the competing models behave under evolutionary events such as gene duplication.