UNIVERSITY OF AMSTERDAM

STOCHASTIC SIMULATION ASSIGNMENT 2

Preying on Parameters: A Study on Optimization Methods for Parameter Estimation of the Lotka-Volterra Model

December 24, 2021

Students:
Steven Oud (steven.oud2@student.uva.nl)
13688650

Malou Bastiaanse
(malou.bastiaanse2@student.uva.nl)
11426934

Lecturer:
Gábor Závodszky

Course:
Stochastic Simulation

Course code:
5284STSI6Y

Abstract

The most commonly known predator-prey model is the Lotka-Volterra-model. This model, and variants thereof, have been extensively studied and used to analyse population dynamics of real-life predator-prey systems. When doing the latter, a complication is to accurately tune the parameter values of the predator-prey model, to both accurately fit the time-series population data and be biologically meaningful. A new approach is to use a global optimization method to approximate the parameters, in particular with the use of a SA algorithm. This study investigated the use of SA for parameter estimation of the Lotka-Volterra predator-prey model based on time-series data of a predator and prey population. Firstly, we compared to different objective functions, namely MSE and MAE, and observed that the MSE leads to better mean and median performance in terms of R^2 for our instance of the SA algorithm. Secondly, we compare our global optimization algorithm of SA with the local optimization algorithm of Nelder-Mead. The SA is able to find a good approximate global optimum frequently independent of the initial guess, where as the Nelder-Mead gets stuck in local optima for most initial guesses. However, if the initial guess is close enough to the global optimum, the Nelder-Mead algorithm finds a better solution. Lastly, data removal of the predator time-series data has more of an impact than data removal of the prey time-series data. Yet, even with these reduced data sets, an approximate global optimum was still found, but less frequently.

Contents

1 Introduction

Why does war favour predatory fishes? This question was pondered by Italian biologist dr. U D'Ancona, after the population of predatory fishes had increased, whilst its prey decreased, during the first World War. During this period, the Adriatic Sea was the battleground between the navies of Italy and the Austro-Hungarian empire, causing fisheries to cease during that period (Sigmund, 1993). In search of an explanation, he asked the help of his father-in-law, the Italian mathematician Vito Volterra. To answer this question, Volterra build upon the much simpler population model proposed by Alfred Lotka (Lotka, 1910), from which he derived the now-famous Lotka-Volterra predator-prey model (Sigmund, 1993).

The Lotka-Volterra model consists of two first-order nonlinear differential equations, that describe the population growth of two species, one predator and one its prey. These two species are negatively coupled, meaning the growth rate of predators increases as the number of prey increases, yet, the proportional rate of increase of the prey decreases as the amount of predators increases (de Roos, 2011). This results in the periodic fluctuations of the species' populations number, which is so characteristic of predator-prey systems. In case of an outside influence that impacts both populations' growth rate such as fisheries, the number of predators will decrease, yet, somewhat counter-intuitively, the number of prey will increase. Hence in the absence of fisheries, which was the case in the Adriatic Sea during the first world war, the predator population can increase again (Sigmund, 1993).

Although the Lotka-Volterra Model is relatively simple, it became the foundation for much further work on predator-prey interactions (de Roos, 2011), leading to more complex predator-prey models, which are still widely studied within the field of theoretical biology. Such predator-prey models can be fitted to observed data, to better understand and predict the dynamics of real-life species interactions. Parameter estimation for such models have been traditionally done in two ways (Jost and Arditi, 2001): (1) qualitatively, with an analysis of the system behaviour such as the cyclic dynamics, period length, and amplitude, or, (2) quantitatively, based on findings within the field and academic literature, from which the best fit can be manually tuned (David et al., 2006). Yet, due to the increase in computational power over the years, recent studies have made use of global optimization algorithms for fitting models to time series data (Jost and Arditi, 2001).

An example of such a global optimization method is simulated annealing (SA). It is inspired by the annealing process used in metallurgy and material science for creating near-perfect crystals. In annealing, the material is brought to its solid-state by raising it to a high temperature after which it is cooled slowly and carefully to reach a solid-state with minimum energy. If cooled too quickly or slowly however, the material may reach a solid-state with non-minimum energy, comparable to getting stuck in a local optimum in optimization ("Natural Solvers", n.d.). Simulated annealing looks to simulate this problem in silico to solve optimization problems, in which they aim to either minimize or maximize an objective function. As most traditional descent methods, in search of an optimum, whether it steeply or randomly descents, only allow downhill moves, whereas SA allows occasionally for an uphill move, hence has the potential to escape local optima (Tsuzuki and Martins, 2014).

Although SA is not a new optimization method, it has only been sparsely used for fitting of predator-prey models to time-series data (David et al., 2006; Jost and Arditi, 2000). Hence, this study will be evaluating the use of SA for parameter estimation of the Lotka-Volterra predator-prey model based on time-series data of a predator and prey population. The first two research objectives of this report will be on the performance of the SA algorithm for fitting of the predator-prey model, which are: (1) compare the performance of two different objective functions of the SA algorithm, and, (2) assess if and when the global optimization algorithm SA has better performance than a local optimization algorithm. Lastly, we are interested in the interaction between the performance of the SA algorithm and the data is trained in, more specifically, the impact of data removal during the optimization process. Hence our last research objective is: (3) assess the impact of reduced data-sets during the optimization process on the performance of the SA algorithm.

2 Methods

2.1 Predator-Prey Model

As mentioned within the introduction, the Lotka-Volterra predator-prey model consists of two first-order nonlinear differential equations, that represent the growth rate of the predator and prey population. The Lotka-Volterra model can be described as followed:

$$\frac{dx}{dt} = \alpha x - \beta x y \tag{1}$$

$$\frac{dy}{dt} = \delta x y - \gamma y \tag{2}$$

where x and y are the abundance of prey and predators respectively. The parameter μ represents the predator's mortality rate in the absence of prey, whereas r represents the exponential growth rate of the prey without predators (de Roos, 2011). Commonly within the field of theoretical biology, the rate at which two species encounter, or when two sexual partners meet, is proportional to the product of their abundance (de Roos, 2011). Hence the rate at which the predator and prey encounter each other is the product of x and y. The parameter β represents the attack rate of the predators, whereas δ encompasses the rate at which the predator population grows as a consequence of the abundance of prey. The values of these parameters are not fixed and can differ between what type of predator-prey interaction is needed to be modelled. However, the only constraint is that the parameter values need to be positive and nonzero for them to have any meaning. This only bound that we have imposed on the problem and thus integrated within local and global optimization algorithms. Whichever parameters are chosen, they determine the dynamics of the systems, which we will briefly outline in the next section.

2.1.1 Dynamical System Analysis

For the stability analysis of the predator-prey model, we will be following the approach as described in de Roos (2011). Firstly, it is of interest to identify the steady states of the system. For this we need to identify the nullclines, meaning where the rates of change are zero of Equations 1 and 2. Rather easily it can be found that dx/dt is zero when:

$$x = 0 \quad \text{and} \quad y = \frac{\alpha}{\beta},$$

and that dy/dt is zero when:

$$y = 0 \quad \text{and} \quad x = \frac{\gamma}{\delta}.$$

The two steady states are thus $(x^*, y^*) = (0, 0)$, referred to as the trivial steady state, and $(x^*, y^*) = (\frac{\gamma}{\delta}, \frac{\alpha}{\beta})$, which is referred to as the internal steady state. To determine the stability of these two steady states, we will first derive the Jacobian of the system which is:

$$\mathbf{J} = \begin{bmatrix} \alpha - \beta y & -\beta x \\ \delta y & \delta x - \gamma \end{bmatrix}.$$

We thus find for the trivial steady state that:

$$\mathbf{J} = \begin{bmatrix} \alpha & 0 \\ 0 & -\gamma \end{bmatrix}.$$

As this is a diagonal matrix, the eigenvalues are on the diagonal with $\lambda_1 = \alpha$ and $\lambda_2 = -\gamma$. With both a positive and a negative eigenvalue, this steady state can be classified as a saddle point. The corresponding eigenvalues of the system are $v_1 = [1 \ \ 0]^T$ and $v_2 = [0 \ \ 1]^T$. These eigenvectors imply that the steady state is stable against perturbations within the predator density, but unstable against perturbations within the prey density (de Roos, 2011).

Next, we will analyse the stability of the internal steady state. Here the Jacobian of the system is reduced to:

$$\mathbf{J} = \begin{bmatrix} 0 & -\beta\frac{\gamma}{\delta} \\ \delta\frac{\alpha}{\beta} & 0 \end{bmatrix}.$$

The characteristics equation of the systems can be found by setting $\det(\mathbf{J} - \lambda I) = 0$, from which we find:

$$\alpha^2 + \beta\frac{\gamma}{\delta}\delta\frac{\alpha}{\beta} = \alpha^2 + \alpha\gamma = 0,$$

from which we find the eigenvalues $\lambda_{1,2} = \pm\sqrt{\alpha\gamma}i$. As there are two imaginary eigenvalues, the internal steady state is found to be a neutral center (de Roos, 2011). This results in the famous characteristic of the predator-prey model, with sustained cycles in both the predator and prey population. Even with the parameters kept at a fixed value, the size of the elliptic curves is dependent on the initial state of the population densities. To illustrate these elliptic curves, a phase plane plot is provided within Figure 1, for various initial conditions of x and y. The nullclines are plotted additionally (dashed line).

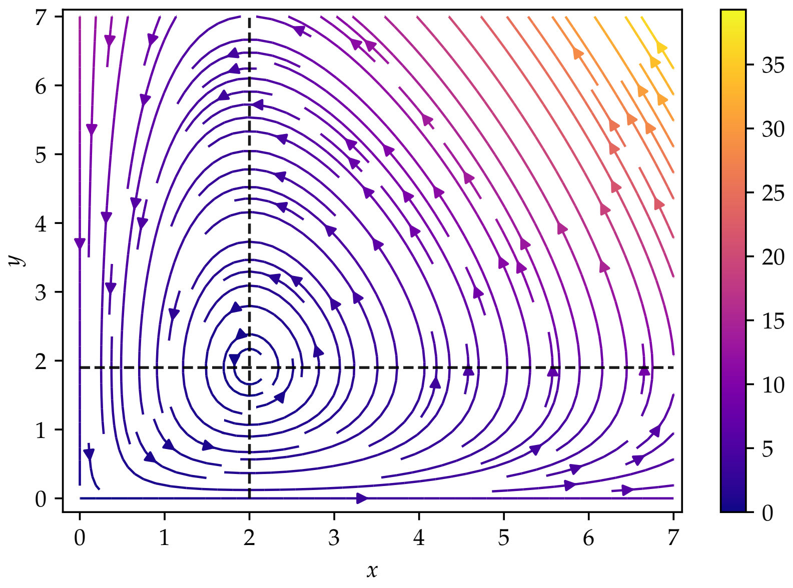


Figure 1: *Phase plot of the Lotka-Volterra predator-prey model. Color map indicates the magnitude* $\sqrt{(dx/dt)^2 + (dy/dt)^2}$ *at that point and horizontal and vertical dashed lines are drawn to highlight the states at which* dx/dt *and* dy/dt *are zero respectively (the nullclines). The parameter values are* $\alpha = 0.95$, $\beta = 0.5$, $\delta = 1$, *and* $\gamma = 2$.

2.2 Parameter Estimation Predator-Prey Model

2.2.1 Data Set

We are given a labeled data set $\mathcal{D}$ of $M = 100$ samples from a noisy Lotka-Volterra model with unknown parameters (Figure 2):

$$\mathcal{D} = \left\{ \left(t_1, \boldsymbol{p}_1\right), \left(t_2, \boldsymbol{p}_2\right), \ldots, \left(t_M, \boldsymbol{p}_M\right) \right\}, \tag{3}$$

where $\boldsymbol{p}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}$ is the population state at time t_i with x_i predators and y_i preys. The data samples are evenly spaced within the time range $(0, 20)$. From this data set, we seek to find a model such that the predicted population $\hat{\boldsymbol{p}}(t_i)$ is close to the population from the data set $\boldsymbol{p}_i \in \mathcal{D}$ for every $i \in \{1, \ldots, M\}$. It is known that the data comes from a Lotka-Volterra model, which then turns this problem into the problem of finding the corresponding parameters to the Lotka-Volterra model. As the data set is noisy, we will never find a perfect fit. Instead, we aim to find a "good enough" fit by minimizing a predefined objective function that measures how well our model predicts the given data.

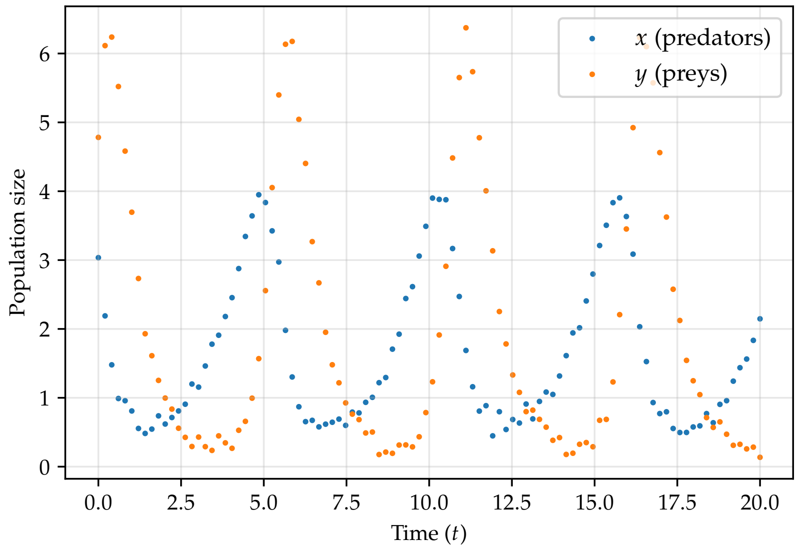


Figure 2: *Time series of a Lotka-Volterra model with added Gaussian noise.*

2.2.2 Optimization Methods

Optimization appears in many science and engineering problems, e.g. finding the shortest route between cities to minimize the time taken to deliver packages (Matai et al., 2010), finding the ground state of quantum systems to design new materials and drugs (Golub and Van der Vorst, 2000), or finding the optimal robot morphology for performing a certain task in an environment (Eiben, 2020). In an optimization problem, we seek to find the optima (minimum or maximum) of some objective function. Most interesting optimization problems are NP-hard — that is, we know of no algorithm that solves (or even verifies) the problem in polynomial time (Burke and Kendall, 2014). To tackle these hard optimization problems, we often consider smaller, more tractable problems. For example, instead of finding a global optimum, we may settle for instead finding a local optimum, or an approximate global optimum. In this report we will consider two types of optimization algorithms: a local optimization algorithm (hill-climbing), and a global optimization algorithm (simulated annealing). These are two types of optimization algorithm which have different goals and may be used for different needs. If, for example, an approximate global optimum is preferred over an exact local optimum, simulated annealing may be the better choice.

2.2.2.1 Hill-Climbing Algorithm

hill-climbing algorithms find the local optimum for a specific region of the search space. These algorithms start from some initial state and iteratively visit neighbour solutions until no better states are found. At that point, the algorithm is said to have converged to a local optimum. Any neighbour will be worse than its current state, thus no improvement is to be made. This is a key feature of hill-climbing algorithms: they get stuck in local optima, unable to further explore the search space to find a potential better optimum.

The hill-climbing algorithm we will consider for local optimization in this report is the Nelder-Mead algorithm. Proposed by Nelder and Mead (1965), the Nelder-Mead algorithm performs multidimensional optimization without derivatives by evaluating the objective function at the vertices of a simplex. We choose the Nelder-Mead algorithm because of its robustness in many applications including parameter estimation — it is widely used for optimization problems where values are subject to noise, as it the case for our problem. We use the implementation provided by `scipy` in the `scipy.optimize` package with an acceptable absolute error for convergence of 10^{-8}. The rest of the parameters are set to the default, unless noted otherwise.

2.2.2.2 Simulated Annealing

The main focus on this report will be on simulated annealing (SA). Simulated annealing is a stochastic optimization algorithm that approximates the global optimum of a function. As explained within the introduction, it is inspired by the annealing process for creating perfect crystals, thus reaching an optimum state. The general SA algorithm is described in Algorithm 1. In this report, we will consider SA with an inhomogeneous Markov chain in which the temperature is decreased after every Metropolis step ("Natural Solvers", n.d.). This choice was made to keep the experiments simple and computationally manageable, additionally, we argue that with a steady and slow temperature decrease, the global optima can still be approximated. Besides the parameters listed in the input of the algorithm, two other important parameters are the neighbourhood structure and the cooling schedule.

Neighbourhood Structure The neighbourhood structure decides how the algorithm chooses a trial neighbour state $\boldsymbol{s}_{\text{new}}$ given the current state $\boldsymbol{s}$. For the problem of parameter estimation of the Lotka-Volterra predator-prey model, we define the neighbourhood structure as follows. Given a state $\boldsymbol{s} = \begin{bmatrix} s_1 & s_2 & s_3 & s_4 \end{bmatrix}$ (corresponding to α, β, δ, and γ in the Lotka-Volterra model respectively), a random neighbour state $\boldsymbol{s}_{\text{new}}$ is obtained by sampling from a Gaussian distribution with mean s_i and $\sigma = \eta s_i$ with $\eta \in [0, 1)$ for all $s_i \in \boldsymbol{s}$:

$$\boldsymbol{s}_{\text{new}} = \begin{bmatrix} \tilde{s}_1 & \tilde{s}_2 & \tilde{s}_3 & \tilde{s}_4 \end{bmatrix}, \tag{4}$$

where $\tilde{s}_i \sim \mathcal{N}(s_i, \eta s_i)$. The standard deviations are set proportionally to the current state's values by a scaling factor η to allow the algorithm to better escape local optima.

Cooling Schedule For the cooling schedule, we will experiment with different cooling schedules proposed in the literature. As pointed out before, the cooling schedule is crucial in getting the algorithm to find a good approximate global optimum. The performance of SA depends heavily both in terms of quality and computation time on the choice of cooling schedule (Mahdi et al., 2017). Unfortunately, the ideal cooling schedule depends heavily on the specific problem, energy landscape, and neighbourhood structure. Therefore, the cooling schedule has to be chosen based on empirical experimentation. In this report, we will consider the following cooling schedules:

- Linear:
$$T_t = T_0 - nt, \tag{5}$$
where t is the time step and n the slope at which the temperature decays.
- Geometrical:
$$T_t = T_0 \alpha^t, \tag{6}$$
where $\alpha \in [0, 1)$, but usually set to a number close to 1.
- Quadratic:
$$T_t = \frac{T_0}{1 + \alpha t^2}. \tag{7}$$

For better comparison of these three cooling schedules, Figure 3 provides visualization of the decrease of T over time for these three schedules. As can be seen, keeping α fixed, the quadratic cooling schedule has the steepest decline, followed by the geometric cooling schedule. Although somewhat unsurprising, the linear cooling schedule decreases linearly. The exact tuning of the simulated annealing parameters will be discussed in Section 3.1.

Algorithm 1: Simulated Annealing

```
Input: Initial state $s_0$, objective function $E$, initial temperature $T_0$, number of iterations $k$
Output: Optimized state $s^*$
$s \leftarrow s_0$;
$T \leftarrow T_0$;
for $t \leftarrow 1$ to $k$ do
    Choose trial neighbour state $s_{\text{new}}$;
    if $E(s_{new}) < E(s)$ then
        $s \leftarrow s_{\text{new}}$;                          ▷ Always accept downhill move
    else
        $\Delta E \leftarrow E(s_{\text{new}}) - E(s)$;
        Generate uniform random number $R \sim U(0, 1)$;
        if $R \leq \exp(-\Delta E/T)$ then
            $s \leftarrow s_{\text{new}}$;              ▷ Accept uphill move probabilistically
        end
    end
    Decrease $T$ according to cooling schedule;
end
return $s$;
```

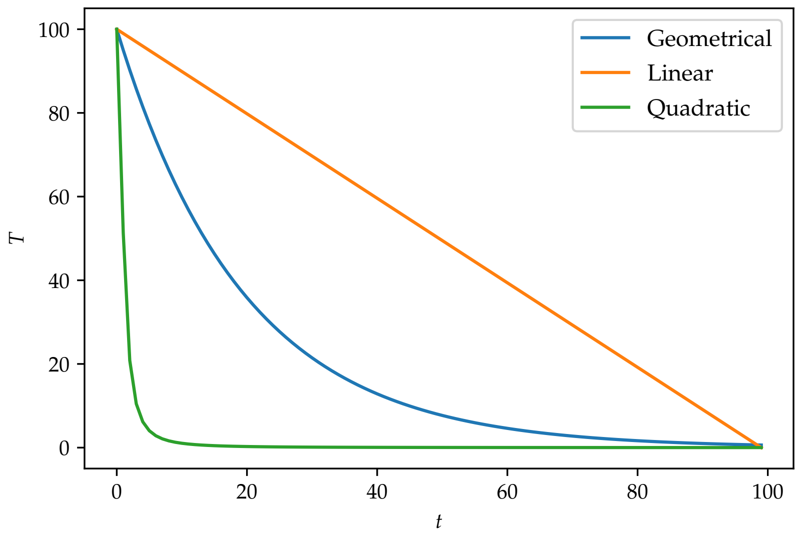


Figure 3: *Examples of the three cooling schedules used in this study: linear, geometrical, and quadratic cooling. The α parameter is set to 0.95 and T_0 is 100. This example is done for 100 iterations.*

2.2.3 Objective Functions

The objective function encodes the problem that one wishes to solve through optimization. It defines what a good and bad solution looks like, which in turn guides the optimization algorithm. Typically, the objective function is small for good solutions and large for bad solutions. Then, through optimization we wish to find parameters such that the objective function is small, or ideally, minimal. For the problem of parameter estimation of the Lotka-Volterra, our objective function will be some comparison between the given data points and our predicted data points. In order to generate the predicted data points for some trial parameters, we need to numerically integrate the Lotka-Volterra equations (Equations 1 and 2) at the time points from the provided data set. This is achieved through the `odeint` function provided by `scipy.integrate`, which uses the LSODA method from ODEPACK (Hindmarsh, 1983). Note that this means that we need to numerically integrate Equations 1 and 2 for every objective function evaluation, greatly increasing the computational cost of the optimization problem.

2.2.3.1 Mean Squared Error

The first objective function we will consider is the mean squared error (MSE):

$$\text{MSE} = \frac{1}{M} \sum_{i=1}^{M} (\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i)^2 , \tag{8}$$

where M is the number of samples, $\boldsymbol{p}_i$ is the actual data value, and $\hat{\boldsymbol{p}}_i$ is the predicted data value. It can be thought of as the average of the squared Euclidean distances between the true values from the data set and our predicted values. Because the MSE squares each term, larger errors can be more heavily weighted than smaller ones. This characteristic could either be advantageous or disadvantageous for the SA algorithm, hence for the second objective function was chosen to be mean absolute error (MAE).

2.2.3.2 Mean Absolute Error

Alternatively, we will also consider the MSE:

$$\text{MAE} = \frac{1}{M} \sum_{i=1}^{M} \left| \boldsymbol{p}_i - \hat{\boldsymbol{p}}_i \right| , \tag{9}$$

which can be thought of as the average of the absolute errors between the true values and predicted values. The MAE differs from the MSE in that it is weighs outliers less heavily.

2.3 Hyper-Parameter Choices

3 Results

3.1 Hyper-Parameter Tuning

3.1.1 Initial Temperature T_0

Choosing the initial temperature for simulated annealing is not a straightforward problem. One has to ensure that T_0 is not too high to have a reasonable computation time, but high enough to allow freedom to explore the search space well enough. A common strategy is to choose T_0 such that the uphill acceptance ratio is equal to some value χ_0 (Kirkpatrick et al., 1983). The choice of χ_0 is problem-specific and varies a lot throughout the literature. For example, Aarts et al. (2018) use $\chi_0 = 0.8$, Pao et al. (1999) use $\chi_0 = 0.7$, and Chen et al. (1998) use $\chi_0 = 0.97$. For our problem, we found $\chi_0 = 0.9$ to work well. To find the optimal T_0 that gives an uphill acceptance rate of 0.9, we run our simulated annealing algorithm with a constant cooling schedule for varying T_0 ($T_t = T_0$). We measure the ratio of uphill moves made by the algorithm for these different T_0 (Figure 4). From this data, we choose to set $T_0 = 200$ for our experiments, as it is high enough to explore the search space well, and low enough to be computationally efficient.

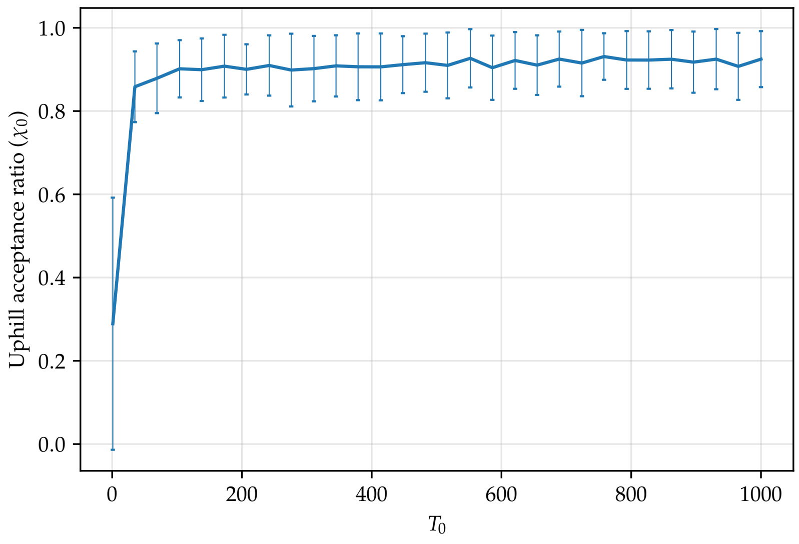


Figure 4: *Average uphill acceptance ratio for simulated annealing with constant $T = T_0$ for varying T_0. The number of iterations was set to 200 and the mean and standard deviation of 20 repeated runs for 5 different initial conditions is shown.*

3.1.2 Cooling Schedule and Iteration Number

An important parameter to consider for the SA algorithm is the cooling schedule. We will compare the three cooling schedules described in Section 2.2.2.2: linear (Equation 5), quadratic (Equation 7), and geometrical (Equation 6). For each cooling schedule, we do 20 runs for five different initial guesses for varying number of iterations and investigate the effect on the MSE. We set $\alpha = 0.95$ based on empirical experimentation. The results are shown in Figure 5. We find that the linear cooling schedule performs worst. It reaches the highest maximum MSE for every k, and the mean MSE even goes up as k is increased. The quadratic and geometric schedules perform very similar, with quadratic having a slightly lower overall mean. For our experiments, we thus choose the quadratic cooling schedule with $k = 2000$. The choice of k is a trade-off between computational cost and solution quality. It has to be high enough so that the cost landscape can be explored sufficiently, but not too high so that the computational cost is too large.

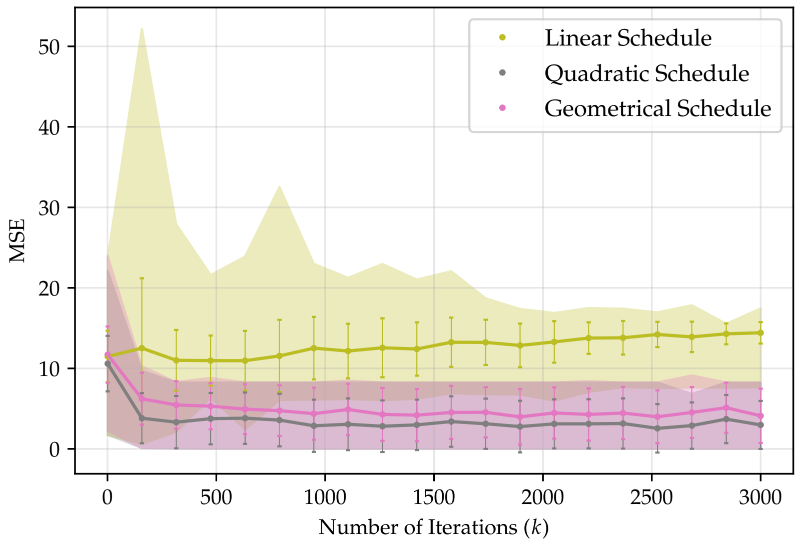


Figure 5: *Comparison of different cooling schedules for varying number of iterations. Shown is the mean MSE of 20 runs of 5 different initial guesses (S_0) with the sample standard deviation as error bars and minimum and maximum as area.*

3.1.3 Neighbour Step Size Scaling Factor η

As described in Section 2.2.2.2, for neighbour state selection we sample from a Gaussian with mean s_i and standard deviation $\sigma = \eta s_i$ for every $s_i \in \boldsymbol{s}$ where $\boldsymbol{s}$ is the current state. Choosing the right scaling factor η has a large impact on the performance of the SA algorithm. To find the right η for our problem, we experiment with various η and analyse how changing η impacts the MSE. We run 20 simulations of five different uniformly random initial conditions and plot the average MSE in Figure 6. On average, we find that $\eta \approx 0.2$ performs best. From this experiment, we choose to fix $\eta = 0.2$ for the other experiments conducted in this report.

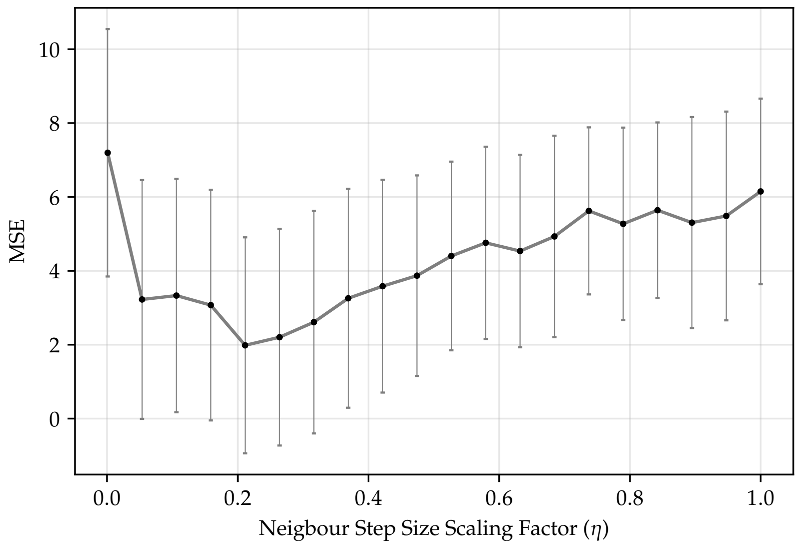


Figure 6: *Impact of choice of η on the MSE. T_0 was set to 200, number of iterations k was 2000, and the quadratic cooling was used. Shown is the mean MSE of 20 runs of 5 different initial guesses (S_0) with the sample standard deviation as error bars .*

3.2 Comparison Objective Functions

The first research objective of this report is to compare the performance of two different objective functions on the SA algorithm. The chosen objective functions were the MSE and MAE. The objective function not only determines the cost measure of the final state, meaning final parameters, it is also integral to the functioning of the algorithm, as the objective functions are evaluated within each iteration to determine if the next state should be accepted or rejected. Hence, we firstly wanted to observe if the objective functions show differences in how the algorithm convergences to their final state. We ran 500 simulations using three different parameter state, however, little to no differences could be observed in how either objective functions convergences to their final state. Both objective functions have no negative impact on the functioning in terms of convergence of the SA algorithm. Hence, these results shown in Figure 11 can be found within the Appendix.

Observable differences can be found when inspecting the quality of fit of the final state for each objective function. Similarly to before, 500 simulations were run for three different initial states, where Figure 7 shows the data distributions of the fit of the final state of parameters for both objective functions. To have an equal comparison between both objective functions, we assessed the final fit in terms of R^2. A near-perfect fit in terms of R^2 will be close to 1 and an R^2 below zero indicates that the parameter set performs worse than just taking the mean of the training data. As can be seen in Figure 7, the highest peaks for both objective functions is close to 1, meaning a good fit. The data points around this peak have most likely approximated the global optimum. Although the difference is minimal, for at least two of the initial states, the MSE objective function has more data points around this peak close to 1. Besides the global optima, a large portion of the data points are centered around local optima, all with an R^2 lower than 0, indicating a bad fit. This is the case for both objective functions. MAE has a couple of large peaks with R^2 between -0.5 and 0, whereas MAE has two large but thin peaks centered around -0.2 and -1.2. Besides these larger peaks, smaller peaks for various R^2 values can be observed, showing that the final states can end up in a large variety of local optima.

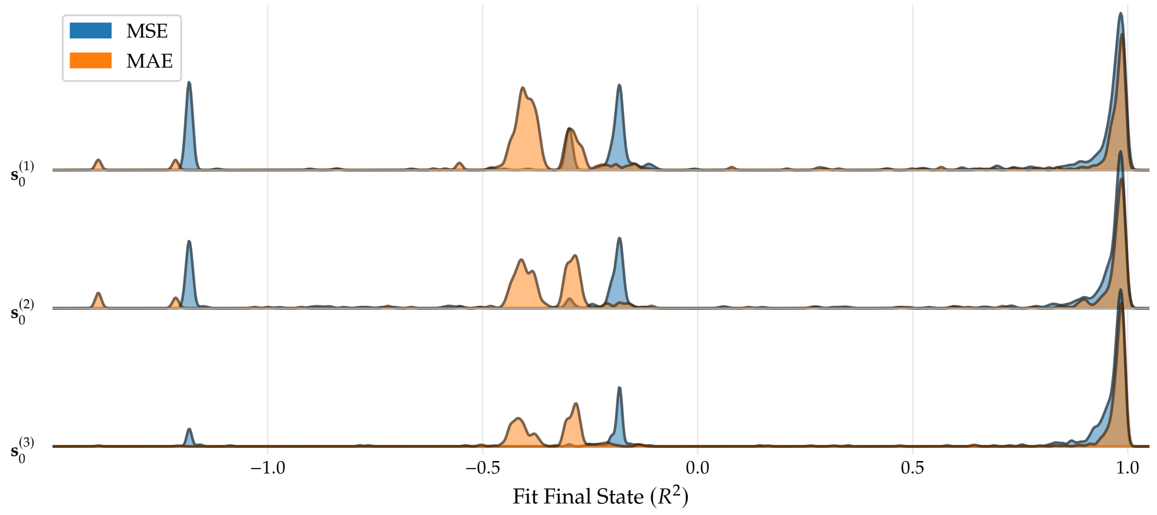


Figure 7: *The data distribution of the R^2 of the final states, meaning parameter sets, are shown of 500 simulations using three different initial states (S_0). The SA algorithm was run with $T_0 = 200$, quadratic cooling, and 2000 iterations. Results are shown for to different objective functions, meaning the MSE and the MAE objective functions.*

Statistics of the results within Figure 7 are more closely looked at in Table 1. In all cases, the MSE objective function has a higher mean and median R^2 value for their final states. Additionally, the use of MSE will also lead to lower minimum values of R^2. These minima are in some instances (once for MSE and three times for MAE) even lower than the R^2 value of the initial state, showing that the final state has a worse fit than the state it started from. The high values of the maxima for either objective function indicate that the global optimum has been approximated. Overall, these results indicate that the MSE objective function is the better choice in terms of quality of fit for the SA algorithm, hence all further experiments will be using MSE.

s_0	**MSE Objective Function (R^2)**				**MAE Objective Function (R^2)**			
	Mean	Median	Min	Max	Mean	Median	Min	Max
$s_0^{(1)}(R^2 = -1.297)$	0.293	0.888	-1.183	0.995	0.102	-0.291	-1.394	0.994
$s_0^{(2)}(R^2 = -1.239)$	0.354	0.931	-1.183	0.995	0.160	-0.278	-1.395	0.995
$s_0^{(3)}(R^2 = -0.016)$	0.667	0.966	-1.183	0.994	0.390	0.953	-1.394	0.994

Table 1: *The statistics in terms of mean, median, minimum, and maximum R^2 value of the final states are shown, based on 500 simulations using three different initial states (s_0). The R^2 value of the initial state is displayed additionally. The SA algorithm was run with $T_0 = 200$, quadratic cooling, and 2000 iterations. Results are shown for to different objective functions, meaning the MSE and the MAE objective functions.*

3.3 Comparison Optimization Methods

In this section we will compare the SA algorithm with the hill-climbing algorithm Nelder-Mead. We note that it is hard to do a fair comparison of these two optimization algorithms: they are different tools meant for different purposes. One approach to make the comparison somewhat fair would be to set a limit on the number of objective function evaluations both methods are allowed to do. However, this would favor the SA algorithm more than the local optimization. While the SA algorithm can always use more objective function evaluations to reach a better approximate optimum, the local optimization algorithm tends to get stuck in a local optimum relatively quickly, after which no further progress can be made with the extra objective function evaluations. In term, our comparison is about the full potential of both optimization methods — we do not look to say anything about their efficiency in terms of resource utilization, or which method is better, as this heavily depends on the problem and desired output.

We compare the final MSE obtained by the Nelder-Mead and SA algorithm for three different initial guesses in Table 2. As SA is a stochastic algorithm, statistics are shown for the SA results. First, we note that the Nelder-Mead algorithm gets stuck in a local optimum when starting from initial guess $s_0^{(1)}$ and $s_0^{(2)}$. It finds a good global optimum for $s_0^{(3)}$, indicating that $s_0^{(3)}$ sits on a gradient towards a global optimum. On the other hand, the SA algorithm is able to approximate a global optimum for each s_0 with a high probability. The distribution of all MSE for the SA results is shown in Figure 8. From this distribution we can see that the SA algorithm roughly favors three different kinds of optima. Most frequently, it finds an approximate global optimum around 0, but it also tends to settle into two local optima around 6 and 8. Furthermore, from Table 2 we can see that the median MSE of the SA method is at most 0.440, meaning that half of the MSE are below that, so relatively few runs are required to get a decent approximation of the global optimum. For better initial guesses this median is even lower.

s_0	**Nelder-Mead MSE**	**SA MSE**			
		Mean	Median	Min	Max
$s_0^{(1)} = \begin{bmatrix}0.666 & 2.942 & 8.129 & 4.888\end{bmatrix}$	8.728	3.373	0.440	0.034	8.245
$s_0^{(2)} = \begin{bmatrix}1.590 & 3.965 & 8.940 & 2.725\end{bmatrix}$	6.443	3.290	0.134	0.051	8.245
$s_0^{(3)} = \begin{bmatrix}2.912 & 2.706 & 2.016 & 1.473\end{bmatrix}$	0.019	1.558	0.114	0.038	8.245

Table 2: *Comparison of the Nelder-Mead and SA optimization algorithm for different s_0. Statistics for SA were taken from 500 runs per initial condition. The SA algorithm was run with $T_0 = 200$, quadratic cooling, and 2000 iterations. The Nelder-Mead algorithm was run until an absolute error of 10^{-8} was reached, or until 800 objective function evaluations were done.*

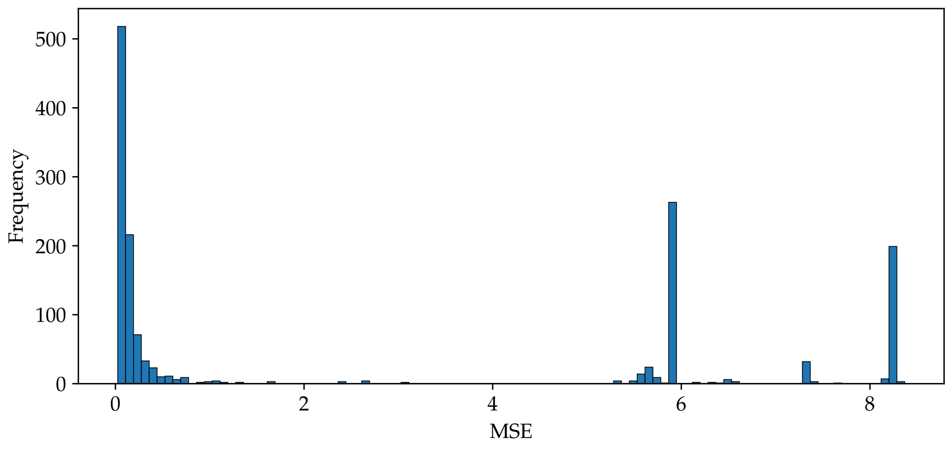
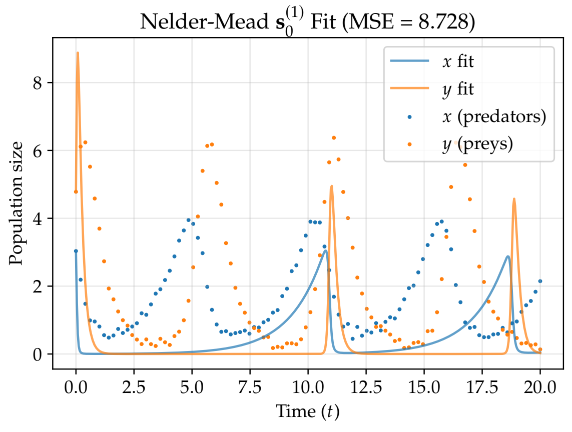


Figure 8: *Distribution of MSE from SA experiment. Shown is the MSE of all 500 runs of all three different initial conditions.*

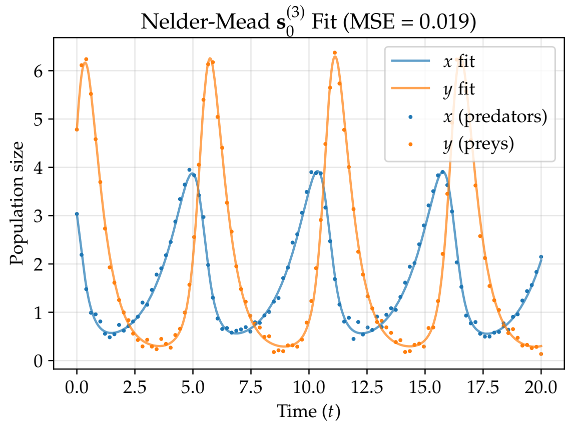
To get a better understanding of the difference in quality of fit between different MSE, we visualize the fits found by Nelder-Mead and SA in Figure 9. The median MSE fit is shown for the SA algorithm. From this we can see that the difference between the median fit of SA (MSE = 0.445) and the fit of Nelder-Mead (MSE = 8.728) for $s_0^{(1)}$ (left column) is large: the parameters found by Nelder-Mead fail to represent the patterns in the data, while the SA solution managed to find the general period of the system and approximate the amplitude. The difference between the two algorithms for $s_0^{(3)}$ (right column) is less visible: while the fit found by Nelder-Mead (MSE = 0.019) certainly fits the data better, the median fit found by SA is very close, and while not going through all data points directly, it certainly predicts the dynamics of the system well. It is hard to even say which fit is actually better for $s_0^{(3)}$, as there is noise in the original data set. For example, the Nelder-Mead algorithm might actually be overfitting to the noisy data, while the SA algorithm potentially prevents overfitting by approximating the solution.

From our experiment we can conclude a few things about the hill-climbing local optimization method Nelder-Mead and the global approximation optimization method SA. First, the hill-climbing algorithm is more sensitive to the choice of initial guess, as is expected from a local optimization algorithm. Second, the cost difference between the two methods is big: Nelder-Mead used on average ~ 550 objective function evaluations to find a solution, while the SA algorithm used 2000 objective function evaluations. In addition, the SA algorithm has to be repeated multiple times due to its stochastic nature and chance of finding a bad local optimum. However, as discussed in the previous section, the median SA MSE is already better than the MSE obtained by Nelder-Mead for two of the three initial guesses (for the third initial guess $\mathbf{s}_0^{(3)}$ the median is higher, but not by much). Thus, it would take on average two SA runs to get a solution with a MSE lower than the median.

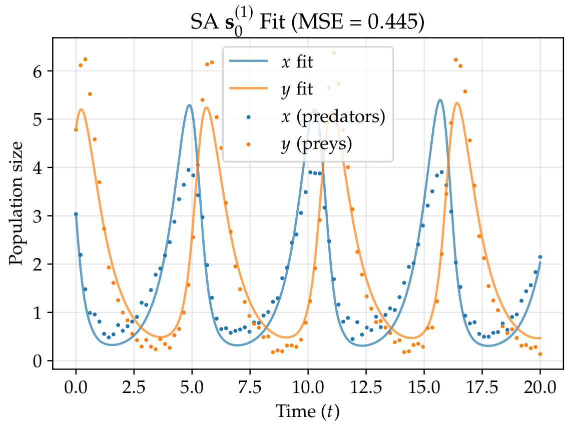
Choosing what kind of optimization algorithm to use then depends on the desired result and resource limitations. If an exact local optimum is preferred over an approximate global optimum, or if the cost of evaluating the objective function is extremely expensive, a hill-climbing algorithm may be preferable. On the other hand, if an approximate global optimum is preferred over a local optimum, SA will be a better choice. Even if the cost of evaluating the objective function is large, by tweaking the hyper-parameters of the SA algorithm some balance between solution quality and computational cost can be achieved.



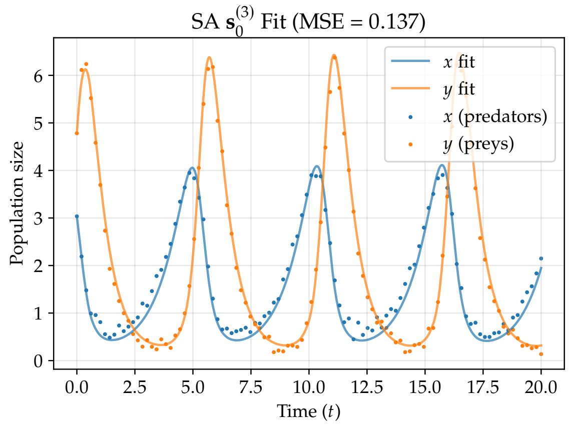
(a) $s^* = \begin{bmatrix}0.689 & 2.788 & 7.979 & 5.203\end{bmatrix}$



(b) $s^* = \begin{bmatrix}0.857 & 0.438 & 1.161 & 2.007\end{bmatrix}$



(c) $s^* = \begin{bmatrix}1.530 & 0.764 & 0.637 & 1.135\end{bmatrix}$



(d) $s^* = \begin{bmatrix}1.024 & 0.503 & 1.059 & 1.717\end{bmatrix}$

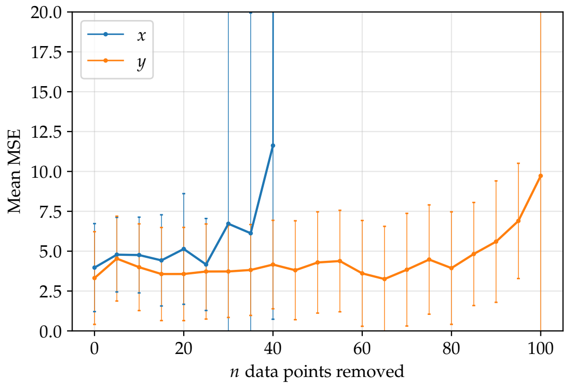
Figure 9: *Visualization of the fit found by Nelder-Mead and SA for* $\mathbf{s}_0^{(1)}$ *and* $\mathbf{s}_0^{(3)}$. *Top: Nelder-Mead fits found, bottom: median SA fits found. Optimized parameters found are shown in the subcaption of each figure.*

3.4 Data Point Removal

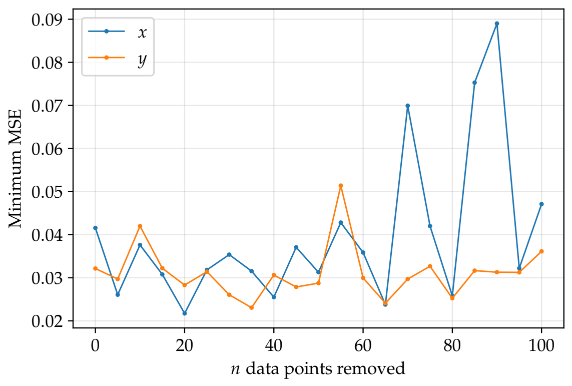
Finally, we investigate the effect of data point removal on the ability of the SA algorithm to estimate the parameters to the model relatively well. We will conduct a number of experiments with various types of reduced data sets. The SA algorithm will be given these reduced data sets to optimize the MSE objective function. Then, the optimized parameters found will be used to predict the data points of the complete data set, for which we will calculate the MSE.

3.4.1 Reverse Chronological Data Removal

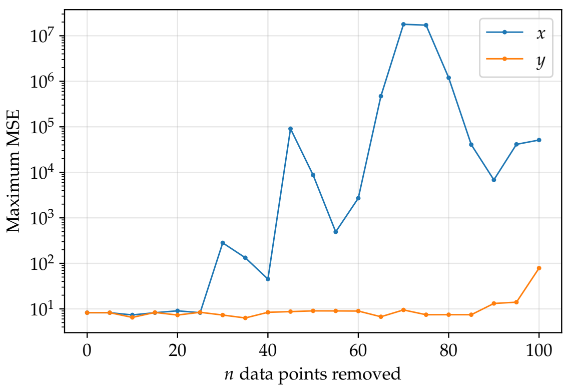
To see the effect that removing data points has on the ability to estimate the parameters well, we fix one dimension of the data set while removing points from the other dimension. This is done for both dimensions x (predators) and y (preys): one gets fixed while data points are removed from the other. Data points are removed in a reverse chronological matter from the back to the front, meaning that points at the end are removed first. The results are shown in Figure 10. It is clear to see that removing data points from the x dimension has a bigger impact on the performance than removing data points from the y dimension. From Figure 10(a) and Figure 10(c) we can identify the critical number of data points for every dimension. For the x dimension, removing more than 25 data points leads to the MSE blowing up. The y dimension seems less important, but removing more than 80 data points shows a steady increase in the mean and maximum MSE, however not nearly as extreme as for the x dimension. We furthermore find that for data point removal in both the x and y dimension, the SA algorithm is still able to find a good approximate global optimum with MSE ≤ 0.09, even if one entire dimension is removed (Figure 10(b)).



***(a)** Mean MSE with sample standard deviation.*



***(b)** Minimum MSE.*



***(c)** Maximum MSE.*

***Figure 10:** MSE for each number of data points removed per dimension. One dimension is fixed while data points are removed from the dimension shown in the plots. Shown data is from 500 runs for one initial guess. Parameters used for SA were $T_0 = 200$, quadratic cooling, and 2000 iterations.*

Following this experiment, we are curious to see if we can still infer the parameters well if we combine two reduced data sets. That is, we remove 25 points from the x dimension and 80 points from the y dimension and see if the parameters can still be inferred. We find that the SA algorithm can still find a good approximate global optimum: the minimum MSE found is 0.033. However, it has great difficulty in estimating the parameters: the mean MSE is 4.460, the median is 7.549, and the maximum is 7.888. With the median so close to the maximum, we can conclude that while possible to find a good approximate optimum with this combined reduced data set, it might take a lot of repeated runs.

3.4.2 Extrema Data Point Removal

Besides removing the data points in reverse chronological order, we additionally wanted to show what happens when we remove specifically chosen data points. The points we chose to remove were the extrema, meaning the highest and lowest values for each time series. Contrary to what was done in the previous section, data points are removed from both populations' time series at once. Two experiments were performed to obtain a valid comparison, removing both the data points furthest from the average (the extrema), and closest to the average, so we end up comparing data sets with an equal amount of data points. An example of this data point removal is visualized in Figure 12 within the Appendix.

The results of these two experiments can be seen within Table 3, shown with 10, 40, and 70 data points removed. These results are from 500 simulations for one initial state ($\mathbf{s}_0 = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}$). There seems to be very little pattern between the quality of fit of the removal of extrema data points and non-extrema data points. In some cases (e.g. 10 and 70 data points removed) the removal of extrema leads to worse results, yet in other instances (e.g. 40 data points removed) the removal of extrema leads to better results. Within this experiment there is also a large number of extreme values for the maximum MSE in both experiments, which ultimately impact the mean and the median. Due to the multi-modal and non-normal data distribution of these experiments, there were no appropriate statistical tests to identify, and thus remove, these extreme values as outliers. Hence no conclusive statements can be drawn from this table.

Number of	**Close to average (MSE)**				**Furthest from average (MSE)**			
data points removed	Mean	Median	Min	Max	Mean	Median	Min	Max
10	5.657	3.447	0.022	0.023	6.246	3.846	0.023	8.523
40	6.010	22.36	0.022	9134	7.477	5.487	0.019	146.6
70	0.328	3.970	0.020	13.80	5.991	4.013	0.022	123.1

Table 3: *Statistics in terms of MSE for two reduced data sets, one with the data points close to the average removed and one with the data points furthest from the average removed. Shown data is from 500 runs for one initial guess. Parameters used for SA were T_0 = 200, quadratic cooling, and 2000 iterations.*

4 Discussion

This report investigates the use of local and global optimization for estimating the parameters of the Lotka-Volterra predator-prey model. We first describe the hyper-parameter tuning process for the simulated annealing (SA) algorithm. Second, we construct and compare two objective functions to use in the optimization process. Third, we compare a local hill-climbing optimization algorithm (Nelder-Mead) with a global approximation optimization algorithm (SA). Finally, we investigate the impact that data removal has on the performance of the SA algorithm.

We first consider the problem of hyper-parameter tuning to assure the SA algorithm performs well. The hyper-parameters of the SA algorithm are very problem-specific and are thus chosen based on empirical experimentation. Different experiments were conducted to choose the best initial temperature, cooling schedule, number of iterations, and neighbour step size scaling factor. All hyper-parameter choices were made with a balance of quality and computational cost in mind. For our problem at hand, we decided upon the parameters $T_0 = 200$, $\eta = 0.2$, quadratic cooling, and $k = 2000$.

Second, we compare the performance of SA with two different objective functions: mean squared error (MSE) and mean absolute error (MAE). We find observable differences in the quality of the final fit for the different objective functions. The MSE objective function was shown to have higher mean and median R^2 values compared to the MAE objective function. That is, when using the MSE objective function, better fits were found on average when using the SA algorithm. A potential reason for this could be that MSE more harshly penalizes errors compared to MAE, with the difference between a good and a bad parameter set being more pronounced, which benefited the SA algorithm. Perhaps this could have been resolved by parameter tuning for each objective function independently, but this was beyond the time and computational budget of this study.

Next, we compare the local hill-climbing algorithm Nelder-Mead with the global approximation algorithm SA. We find that Nelder-Mead is more sensitive to the choice of initial guess compared to the SA algorithm. Furthermore, the SA algorithm is able to find good approximate global optimum frequently regardless of initial guess when the Nelder-Mead algorithm gets stuck in a local minimum. This of course comes at a cost: the SA algorithm uses more objective function evaluations and has to be run on average at least twice to find a decent approximate global optimum. The Nelder-Mead algorithm does manage to find a better global optimum than SA when the initial guess sits on the gradient of a global optimum. However, in general it is hard to find such good initial guess, and an approximate global optimizer might be preferred. If, however, a precise local optimum is preferred over an approximate global optimum, local optimization may still be a good choice.

Finally, we investigate the effect that removing data points has on the performance of the SA algorithm. We begin with fixing one dimension and removing data points in reverse chronological order from the other dimension. From this, we find the critical number of data points for both dimensions. Removing > 20 data points from the x dimension greatly affects the mean performance of SA. The y dimension seems less important for the fitting process: only when > 80 points are removed does the mean MSE increase slightly. Despite these numbers, we find that the SA algorithm is able to find good approximate global optimum even when one whole dimension is removed and when two reduced dimensions are merged into one data set. We further try to analyze the effect of data removal by trying to identify what kind of data points are most critical. We hypothesize that extrema data points far away from the mean are more critical to the process than points close to the mean. However, our experimental results are inconclusive — we can draw no conclusion on this.

This work can be extended in the following ways. First, a fair comparison between Nelder-Mead and SA can be done by constructing experiments with a fixed number of objective function evaluations allowed for both methods. Second, a comparison of the performance of the SA algorithm with homogeneous and inhomogeneous Markov chains can be conducted to assess the differences. Finally, more research and experimentation regarding parameter tuning can be done by for example doing parameter tuning with multiple objective functions.

References

Aarts, E. H., Korst, J. H., & Van Laarhoven, P. J. (2018). 4. simulated annealing. *Local search in combinatorial optimization* (pp. 91–120). Princeton University Press.

Burke, E. K., & Kendall, G. (2014). *Search methodologies: Introductory tutorials in optimization and decision support techniques*. Springer.

Chen, H., Flann, N. S., & Watson, D. W. (1998). Parallel genetic simulated annealing: A massively parallel simd algorithm. *IEEE transactions on parallel and distributed systems*, *9*(2), 126–136.

David, V., Chardy, P., & Sautour, B. (2006). Fitting a predator–prey model to zooplankton time-series data in the gironde estuary (france): Ecological significance of the parameters. *Estuarine, Coastal and Shelf Science*, *67*(4), 605–617. https://doi.org/10.1016/j.ecss.2006.01.003

de Roos, A. M. (2011). *Modeling population dynamics*. Institute for Biodiversity; Ecosystem Dynamics.

Eiben, A. (2020). Evolving robot software and hardware. *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 1–4.

Golub, G. H., & Van der Vorst, H. A. (2000). Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, *123*(1-2), 35–65.

Hindmarsh, A. C. (1983). Odepack, a systematized collection of ode solvers. *Scientific computing*, 55–64.

Jost, C., & Arditi, R. (2001). From pattern to process: Identifying predator–prey models from time-series data. *Population Ecology*, *43*(3), 229–243. https://doi.org/https://doi.org/10.1007/s10144-001-8187-3

Jost, C., & Arditi, R. (2000). Identifying predator–prey processes from time-series. *Theoretical Population Biology*, *57*(4), 325–337. https://doi.org/10.1006/tpbi.2000.1463

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671–680.

Lotka, A. J. (1910). Contribution to the theory of periodic reactions. *The Journal of Physical Chemistry*, *14*(3), 271–274. https://doi.org/10.1021/j150111a004

Mahdi, W., Medjahed, S. A., & Ouali, M. (2017). Performance analysis of simulated annealing cooling schedules in the context of dense image matching. *Computación y Sistemas*, *21*(3), 493–501.

Matai, R., Singh, S. P., & Mittal, M. L. (2010). Traveling salesman problem: An overview of applications, formulations, and solution approaches. *Traveling salesman problem, theory and applications*, *1*.

Natural solvers. (n.d.).

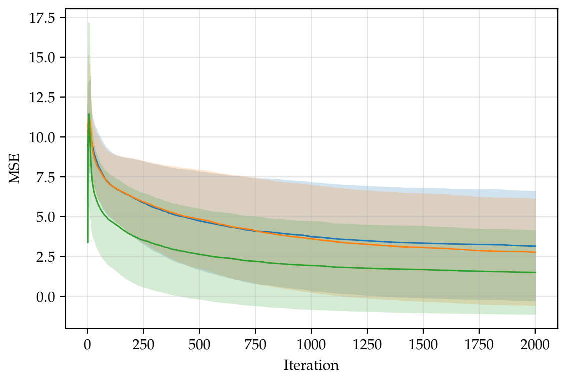
Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The computer journal*, *7*(4), 308–313.

Pao, D., Lam, S., & Fong, A. (1999). Parallel implementation of simulated annealing using transaction processing. *IEE Proceedings-Computers and Digital Techniques*, *146*(2), 107–113.

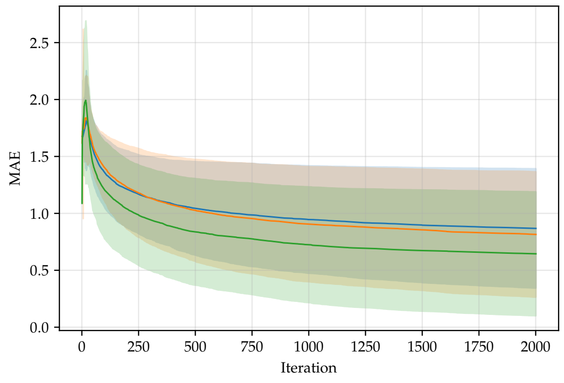
Sigmund, K. (1993). *Games of life: Explorations in ecology, evolution and behaviour*. Oxford University Press, Inc.

Tsuzuki, M. d. S. G., & Martins, T. d. C. (2014). *Simulated annealing: Strategies, potential uses and advantages.* Nova Science Publishers, Inc.

Appendix

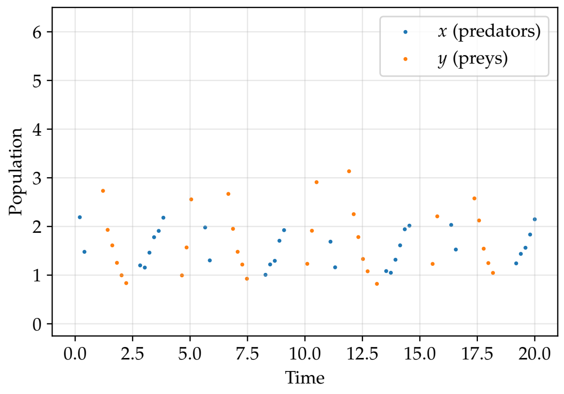


(a) MSE Objective Function

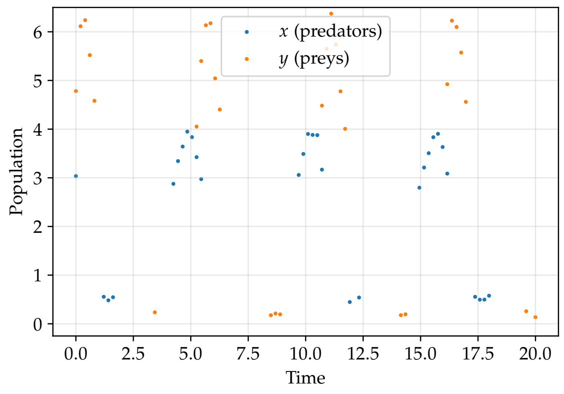


(b) MAE Objective Function

Figure 11: *Trajectories of the cost of the parameter state during the simulation, shown for both the MSE and MAE objective function. The SA algorithm was run with $T_0 = 200$, quadratic cooling, and 2000 iterations. We ran 500 simulation, showing both the mean and the standard deviation around it.*



(a) Removal of the data points furthest from the mean.



(b) Removal of the data points closest to the mean.

Figure 12: *Example of the extrema data point removal experiment. Within this example 70% of the data points were removed. Removal is done for both population's time series.*