Applied math Project1

20180075 Kim myeong cheol

March 2025

1 Code

All implementations for the optimization methods discussed in this report are available on GitHub:

`https://github.com/dayforday2468/Applied_math_project1`

This repository contains the following scripts:

- `brute_force.py`: Implements a brute-force search over a predefined range of (h, k) values to find the optimal parameters.
- `gradient_descent.py`: Uses gradient descent to iteratively update all four parameters (a, b, h, k) to minimize SSE.
- `alternating_optimization_fail.py`: Implements an alternating optimization strategy where (h, k) are updated via gradient descent, and (a, b) are computed via OLS regression in each iteration.

Each script is self-contained and can be executed independently. The results from these implementations were used to compare different optimization methods and assess their effectiveness in parameter estimation for the given model.

2 Problem

Consider more general model.

$$\dot{u} = au^h - bu^k, h < k$$

Logistic model and surface-volume model are some special case of above model. Find the optimal (h, k) pair and explain why $h < k$ is necessary.

2.1 Data

The data used in this project is came from textbook.

Day (n)	**Population** u_n	**Day (n)**	**Population** u_n
0	2	13	513
1	5	14	593
2	14	15	557
3	34	16	560
4	56	17	522
5	94	18	565
6	189	19	517
7	266	20	500
8	330	21	585
9	416	22	500
10	507	23	495
11	580	24	525
12	610	25	510

Table 1: Population growth of *Paramecium aurelia*

Define $\dot{u}_n = \begin{cases} u_{n+1} - u_n & \text{if } u \neq 25 \\ 0 & \text{if } u = 25 \end{cases}$. Then, the gradient of population over time is given as follow.

Day (n)	**Gradient** $\dot{u}_n$	**Day (n)**	**Gradient** $\dot{u}_n$
0	3	13	80
1	9	14	-36
2	20	15	3
3	22	16	-38
4	38	17	43
5	95	18	-48
6	77	19	-17
7	64	20	85
8	86	21	-85
9	91	22	-5
10	73	23	30
11	30	24	-15
12	-97	25	0

Table 2: Gradient of Population over Time

3 Approach

3.1 Search Range

To determine the optimal values of (h, k), we define a search space where h and k vary over a predefined range. Based on domain knowledge and observed behavior of the model, we restrict the values of h and k to the interval $[0, 3]$ with increments of 0.1.

The search grid for (h, k) is given by:

$$h, k \in \{0, 0.1, 0.2, \ldots, 2.9, 3.0\}$$

For each pair (h, k) in this grid, we compute the corresponding parameters (a, b) using OLS regression and evaluate the model fit using the Objective Error Sum (OES).

3.2 OLS Regression for Estimating a and b

To estimate the parameters a and b for a given pair (h, k), we formulate a linear regression model based on the equation:

$$\dot{u}_n = a u_n^h - b u_n^k.$$

For fixed values of h and k, this equation is linear with respect to a and b, allowing us to estimate them using Ordinary Least Squares (OLS).

Let Y be the vector of observed population gradients $\dot{u}_n$, and let the matrix X be constructed as follows:

$$Y = \begin{pmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \vdots \\ \dot{u}_n \end{pmatrix}, \quad X = \begin{pmatrix} u_1^h & -u_1^k \\ u_2^h & -u_2^k \\ \vdots & \vdots \\ u_n^h & -u_n^k \end{pmatrix},$$

where each row corresponds to an observation from the dataset.

The OLS method minimizes the sum of squared errors (SSE):

$$SSE = \sum_n (\dot{u}_n - (a u_n^h - b u_n^k))^2.$$

The optimal values of a and b are obtained by solving the normal equation:

$$\begin{pmatrix} a \\ b \end{pmatrix} = (X^T X)^{-1} X^T Y.$$

This formulation allows us to determine the best-fitting values of a and b for given values of h and k.

3.3 ODE Solve

Once the parameters (a, b, h, k) are estimated, we numerically solve the differential equation:

$$\dot{u} = au^h - bu^k.$$

Using the initial condition $u(0) = u_0$, we integrate the equation over the observed time period using the 'solve_ivp' function in Python. This provides a numerical solution $\hat{u}(t)$, which we compare with the actual data u_n to assess the goodness of fit.

The numerical integration is performed over the interval $[0, T]$, where T is the length of the observed dataset. The integration time steps are chosen to match the observed time points to facilitate comparison. The model error is quantified using the Objective Error Sum (OES):

$$OES = \sum_n (u_n - \hat{u}(t_n))^2.$$

This error metric allows us to select the optimal (h, k) pair that best fits the observed data.

4 Result

4.1 Why $h < k$ is necessary

Figure 1 illustrates the Objective Error Sum (OES) as a function of varying h and k. The diagonal region where $h \approx k$ is intentionally left uncomputed due to the instability of the model in this regime.

When $h \approx k$, the equation simplifies to:

$$\dot{u} = (a - b)u^h,$$

which results in either exponential growth or trivial constant behavior, depending on the sign of $(a - b)$. This leads to unrealistic model behavior and a rapid increase in OES, making the estimation unreliable.

Additionally, as observed in Figure 1, the OES surface exhibits a near-symmetric structure about the diagonal $h = k$. This symmetry suggests that allowing both $h < k$ and $h > k$ does not provide additional meaningful information but rather introduces redundancy.

To avoid redundant solutions and ensure a stable model, the constraint $h < k$ is imposed, effectively reducing numerical instabilities, removing unnecessary computations, and improving the search efficiency for the optimal (h, k) pair.

4.2 best model

Table 3 presents the best models obtained from the optimization process, listing the values of a, b, h, k, and their corresponding Objective Error Sum (OES).

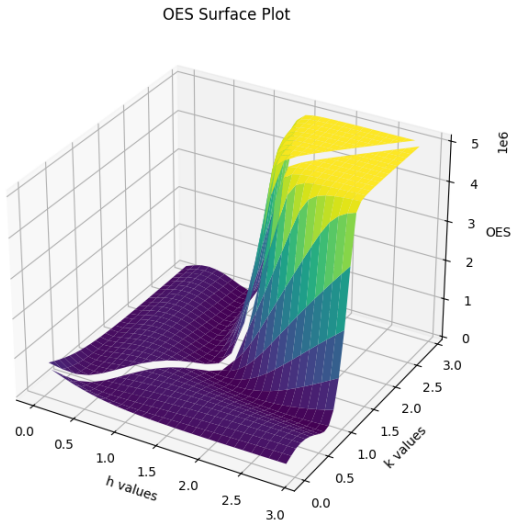


Figure 1: Surface of OES

These models represent the most accurate parameter sets for fitting the observed data.

h	k	a	b	**OES**
2.40	0.90	1.23	0.98	27034.12
0.90	2.40	1.25	1.01	27034.12
0.90	2.50	1.30	1.05	27770.40
2.50	0.90	1.28	1.02	27770.40
1.00	2.00	1.35	1.10	28658.99

Table 3: Best models with optimal (h, k) pairs and their corresponding parameters.

The best models found through optimization are visualized in Figure 2, which compares their predicted dynamics against the observed population data. These models provide the most accurate approximation of the given dataset and demonstrate the effectiveness of the chosen methodology.

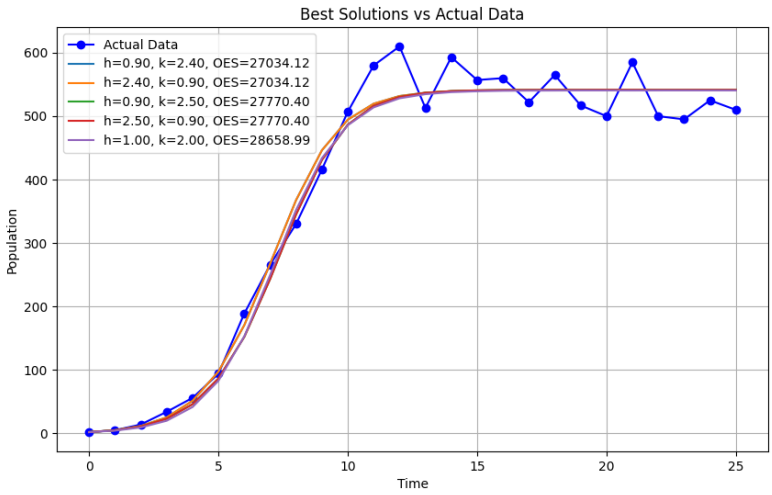


Figure 2: Best solutions

5 Acknowledge

5.1 Gradient Descent

In an effort to optimize the model parameters, we implemented gradient descent, a widely used iterative optimization method. This approach aims to minimize the objective function, Sum of Squared Errors (SSE), by updating the parameters (a, b, h, k) in the direction of their gradients.

1. **Compute the gradients:** We first compute the partial derivatives of the objective function with respect to each parameter:

$$\frac{\partial SSE}{\partial a}, \quad \frac{\partial SSE}{\partial b}, \quad \frac{\partial SSE}{\partial h}, \quad \frac{\partial SSE}{\partial k}$$

   using finite difference approximation:

$$\frac{\partial SSE}{\partial \theta} \approx \frac{SSE(\theta + \epsilon) - SSE(\theta)}{\epsilon}.$$

2. **Update the parameters:** Each parameter is updated in the direction

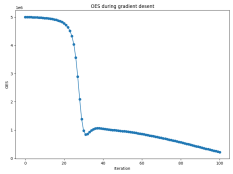
opposite to its gradient:

$$a \leftarrow a - \eta \frac{\partial SSE}{\partial a}/N,$$
$$b \leftarrow b - \eta \frac{\partial SSE}{\partial b}/N,$$
$$h \leftarrow h - \eta \frac{\partial SSE}{\partial h}/N,$$
$$k \leftarrow k - \eta \frac{\partial SSE}{\partial k}/N.$$

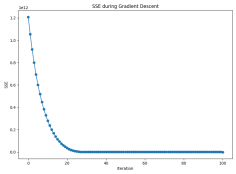
Here, η is the learning rate that controls the step size and $N = \frac{\partial SSE}{\partial a} + \frac{\partial SSE}{\partial b} + \frac{\partial SSE}{\partial h} + \frac{\partial SSE}{\partial k}$ is a normalizer.

3. **Iterate until convergence:** We repeat the above steps for a predefined number of iterations or until there is no improvement in SSE.

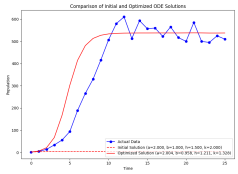
Depending on the initial parameters, the convergence procedures and the outputs look quite different. Here are some results which are generated by initial parameters $(2, 1, 1.5, 2)$ and $(2, 1.5, 1, 2)$:



(a) OES during gradient desent



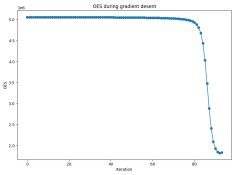
(b) SSE during gradient desent



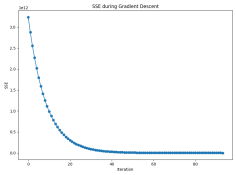
(c) initial and optimized solutions

Figure 3: Result with $(2, 1, 1.5, 2)$

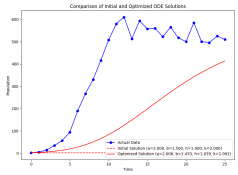
When the parameters are $(2, 1, 1.5, 2)$, SSE is consistently dropped until the end. However, OES is increased at some point which may imply overfitting. The numerical solution shows that the initial flat solution becomes a solution which looks like Logistic growth graph.



(a) OES during gradient desent



(b) SSE during gradient desent



(c) initial and optimized solutions

Figure 4: Result with $(2, 1.5, 1, 2)$

On the other hand, when the parameters are $(2, 1.5, 1, 2)$, both SSE and OSE are dropped consistently during gradient desent procedure. However, the optimized numerical solution's fit is not good. It looks like underestimated. To solve this problem, the longer iteration can be suggested.

5.2 Alternating optimazation

In the initial stages of this project, we attempted an alternating optimization approach to determine the optimal values of (h, k, a, b). This method involved:

1. Initialize the variables h and k randomly.
2. Applying Ordinary Least Squares (OLS) regression to estimate a and b given h and K.
3. Using gradient desent to update h and k given a and b with setting SSE as objective function.
4. Iterating step 2 and 3 until no improvement.

Although this approach seemed promising, we encountered significant challenges:

1. When h and k were updated using gradient descent, the optimal values of a and b also changed due to the OLS regression, leading to minimal or no improvement in the objective function.
2. In most cases, the optimal values of a and b for a given pair (h, k) resulted in a local minimum, preventing h and k from updating further. As a result, alternating optimization failed to refine h and k effectively.
3. We observed that when a and b were fixed at arbitrary values, gradient descent successfully updated h and k. However, when a and b were optimized at each step, the updates for h and k stagnated, indicating that the best (h, k) remained unchanged.

As a result, alternating optimization was not effective in this case, as the interplay between OLS regression and gradient descent led to an optimization loop where h and k did not change significantly. This suggests that a direct gradient-based approach for all four parameters (a, b, h, k) might be a more suitable alternative.