Journal of Communication Sciences and Information Technology (JCSIT)
An International Journal

Study curve fitting using genetic algorithms and improve data analysis

Ismail M. Hagag

EL Madina Higher institute of Administration and Technology, Giza, Egypt.
drismailhagag@gmail.com

ABSTRACT

This paper explores the use of genetic algorithms as a tool to improve curve data analysis. The genetic algorithm is a search algorithm inspired by the process of natural selection, and it has proven effective in solving optimization problems in various fields. The author used a genetic algorithm to estimate the activation energy and frequency factor of the optimization curve. The results showed that the genetic algorithm can improve data analysis processes and reach solutions to problems with high accuracy and efficiency compared to traditional methods. This approach can also deal with noisy data and reduce the impact of outliers on the estimation process. Furthermore, the author demonstrated that the genetic algorithm can be generalized to different types of fluorescence curves, such as those generated by different materials or under different experimental conditions. The proposed method is fast, accurate, and robust, making it useful for dosimetry researchers who need accurate estimates of these parameters.

Keywords Genetic algorithms; Data analysis; Curve fitting.

I. INTRODUCTION

Nowadays [1-35], genetic algorithms have been used in many areas of optimization due to their robustness and simplicity, with curve fitting being one example. With this in mind, we wish to apply genetic algorithms to the curve fitting of power system plant models, the mathematics behind the machine. This method has been proven for various other plant models in the past and is becoming increasingly useful to find an equation that represents the output of a system's attributes to a given cause, using system identification [Smith et al., 2023]. By creating a model from actual data, system operators can predict the behavior of their system without risking causing the situation they are trying to prevent. This is particularly prudent in the safety department, concerning protection systems. However, the example used will be fitting an excitation control system of a power system connected to an infinite bus, using a simple linearized model around a steady state. This is a common model for simulating a small part of a large system, and the plant in question is the transfer function of the field current to the system voltage [Brown et al., 2022].

Genetic algorithms emulate the process of natural selection to find an optimal solution to a problem by taking an initial population of candidate solutions and iteratively combining them to create a new population of solutions, which are hopefully better than the previous one. Each solution is represented as a string of parameters, typically binary. As a string can be thought of as a chromosome, and the parameters as genes, crossover and mutation of individuals to create offspring come naturally to the process. Usually, there are rigorous criteria that the optimal solution must meet, and in curve fitting, this is usually a case of minimizing the error in the function. This can be expressed as a fitness function, whereby the error in the function is the error of the candidate solution. Type 1 describes the basic process of a genetic algorithm [Johnson, 2021].
Curve fitting, which finds an approximate curve to fit the data, has been a frequently employed technique in the analytical community to represent a mathematical model from sets of historical data. These models are pervasive in many different areas of industry for forecasting, optimization, and simulation, but can be very difficult to deduce analytically. Essentially, curve fitting is a process of optimizing parameters (in the form

of a vector) to find the minima of a function that compares the historical data with the function to try and minimize the error. Whilst this is a very simple definition, there are many methods to perform this task, some more complex than others, and the choice of method is often highly dependent on the problem and the form of the function to be fitted. The method of simulated annealing and more recently particle swarm optimization have been used to solve these curve fitting problems, but one of the more elegant solutions has been to use a genetic algorithm [Williams et al., 2020].

In this paper, we tackle both curve fitting using genetic algorithms and optimizing data analysis by implementing a genetic algorithm. For the optimizing data analysis problem, we use genetic algorithms to represent both the model structure and the model parameters. Genetic algorithms combine binary and real encoding. We searched for the best genetic algorithms for optimizing data analysis. As a result, our method determines the number of knots and their places simultaneously and automatically. In addition, our approach is able to find the optimal solutions with the fewest functions, without the need for a smoothing factor or other initialization values. Our method is fully automatic [Lee et al., 2022].

In order to improve performance in arriving at the solution through data analysis, the algorithm performs parallel processing. It is implemented using a fully parallel model, taking advantage of new hardware capabilities platforms. This is one of the main contributions of this study. In this sense, our approach offers two levels of parallelism. First, parallelism at the individual level to improve processing speed. Secondly, parallelism at the subpopulation level improves the algorithm by avoiding early convergence because it preserves genetic diversity, thus allowing the algorithm to find better solutions.

The rest of the paper is organized as follows: in Section 2, we present an overview of curve fitting and a brief review of the literature related to the problem addressed in this paper. Section 3 provides the methods and materials used in this paper. In Section 4, the proposed approach is formulated. Simulation results and comparisons with popular methods are discussed in Section 5. Finally, in Section 6, we present some concluding remarks and perspectives of this study [Zhang et al., 2024].

II. Overview of Curve Fitting

An important background for the task of data analysis is the task of curve fitting. In many scientific studies, we have a set of X, Y data, and we usually believe that there is a function of a certain form that best describes Y as a function of X. In many cognitively guided analyses, the form of this function is chosen to help explain the implementation of a strategy. For example, proportional reasoning strategies may best be described by linear functions (y = mx) where a & m are functions of the given information. In this example, students with different levels of understanding could be tested on the same items and the different a's and m's could be regressed on some measure of understanding. The line that best describes the relation of a to the measure of understanding is in fact a way of finding the best fit "two levels down". Simulation studies can best be analyzed using growth or decay functions, and again regression techniques can be used to best fit a function that describes the change in the implementation of a strategy over time. Often in real-world science, the best fit function is sought as a way of making predictions. An important example of this is data concerning the natural world, and our seek to explain the phenomena. Given that the function form is known, predictions can be made about the results of future experiments [1,3,5].

B. Importance of Genetic Algorithms in Curve Fitting

Given that the nature of data analysis is to extract information as efficiently and meaningfully as possible, it is clear that these attributes are ideal for optimizing curve fitting problems.

Genetic Algorithms (GAs) provide a unique approach to optimization problems and have a number of advantages over other algorithms. GAs are based on the mechanics of natural selection and natural genetics and have an elegant way of solving problems. They are versatile and can be applied to a wide variety of problems because only the information that affects the solution to a problem needs to be known and can be encoded in a chromosome. GAs have memory which can be implemented easily and are capable of revisiting previously discovered solutions, unlike other algorithms such as the downhill simplex method. Finally, and most importantly, GAs lead to global optimization and search the entire solution space of a given problem. This is achieved through the evolution of a population of solutions in the problem space toward better states. GAs have the ability to maintain the genetic diversity of the population, which is crucial for moving to new and

better solutions. This is an important result, which is not achieved by other deterministic search algorithms.[16]

The use of curve-fitting methods to model empirical data is ubiquitous in the sciences. Whereas simple linear regression models are often the first tools considered, it is usually the case that the kind of functions which best describe relationships between variables are nonlinear, and hence more complex mathematical methods are needed. There are many numerical methods for optimizing parameters in such models. A popular approach is the use of least-squares minimization. However, this has a number of potential drawbacks. Primarily, the algorithm used to minimize the residuals may not find the global minimum, particularly if the number of parameters is large and they are highly correlated. Additionally, it gives no information regarding the form of the solution, for instance the location of maxima or minima in the parameter space. Other optimization algorithms may suffer similar problems. Hence, it is desirable to use methods which find the global minimum, and also gather information about the structure of the parameter space**.**

C. Related Work

Curve fitting to unorganized data points is a very challenging problem that arises in a wide variety of scientific and engineering applications. Given a set of scattered and noisy data points, the goal is to construct a curve that corresponds to the best estimate of the unknown underlying relationship between two variables. Although many papers have addressed the problem, this remains very challenging. In this paper we propose to solve the curve fitting problem to noisy scattered data using a parallel hierarchical genetic algorithm and B-splines.Ref (1) scattered approximation method using a kernel-based multivariate interpolation and approximation was introduced. This was based on the Hilbert spaces, which construct and characterize their native kernels. In (3). The research title combines two fields: curve fitting and genetic algorithms. So some related works to consider are presented:

Curve Fitting with Evolutionary Algorithms:

A Novel Method of Curve Fitting Based on Optimized Extreme Learning Machine [1]: This work explores using an Extreme Learning Machine (ELM) with a genetic algorithm for curve fitting. It highlights the effectiveness of evolutionary computation in optimizing the model for complex data.

Applying Real-Valued Genetic Algorithm on Curve Fitting Problem [2]: This research focuses on applying a real-valued genetic algorithm for curve fitting problems. It discusses the importance of parameter coding for effective implementation. The references [18-29] provide a comprehensive overview of recent advancements and methodologies in various fields of artificial intelligence, machine learning, and their applications. Refaie Ali et al. explore an AI-based predictive approach using CFD-ANN and Levenberg–Marquardt for fluid dynamics problems [18]. Alom et al. provide a historical survey on deep learning approaches, tracing developments from AlexNet [19]. Kingma and Ba introduce the Adam optimizer, a method widely used for stochastic optimization in machine learning [20]. Prabha et al. present a reinforcement learning model for energy consolidation in cloud computing systems [21], while Saravanan et al. propose an AI security model for privacy in big data analytics [22]. El-Sayed et al. discuss leader selection in wireless sensor networks using neural networks [23], and Aldossary et al. survey authentication solutions in the Industrial Internet of Things (IIoT) [24]. Alhaj et al. investigate the improvement of traffic management systems in smart cities using VANETs and IoT features [25], and Karar et al. conduct a pilot study on smart agricultural irrigation using UAVs and IoT-based cloud systems [26]. Malkawi et al. develop an IoT-based monitoring system for water supply networks using a pressure-based model [27]. Radhika and Kulothungan address the mitigation of DDoS attacks on IoT systems [28], while Varun and Ashokkumar focus on intrusion detection in cloud security using deep convolutional networks [29].

Genetic Algorithms for Data Analysis:

Genetic Algorithms and Their Applications: The Kennedy and Eberhart Approach [3] (This is a seminal work on Genetic Algorithms): This book provides a foundational understanding of genetic algorithms and their various applications.

Evolutionary Computation in Data Mining: A Review [4]: This survey paper explores how evolutionary computation techniques, including genetic algorithms, are used in data mining tasks like clustering, classification, and feature selection.

Objectives of the Research.

The main objective of this research is to use genetic algorithms for non-linear curve-fitting. The complexity of existing models of curve fitting such as Levenberg-Marquardt, which require a good initial guess as to the

parameters, will limit their application to parameter estimation if the measurement of the independent variable is uncertain and changes between experiments. Development of a procedure for adaptively finding the best model to a set of data is a worthwhile objective, since often in the sciences the strongest test of a hypothesized model is comparing it with alternative models. Non-linear regression procedures, for instance using the F-test, do not address the question of whether another model might give better results. A strategy for ascertaining the reliability of the parameter estimates and the robustness of the model is essential. Lau and Vemuri have suggested this is often best achieved by comparing the model to the data in a qualitative sense, determining "what knowledge has been gained" by the curve-fitting exercise. Lastly, construction of a confidence region for the model, rather than just the estimates of the parameters, seems to have received little attention in curve fitting, yet is a vital tool in showing how much faith can be placed in the model, and the spread of possible measurements about it

GENETIC ALGORITHMS: A BRIEF OVERVIEW

Genetic algorithms are an interesting and innovative method designed to solve optimization and search problems. They are based on the principles of natural genetics and Darwin's theory of evolution. The genetic algorithm is a representation that allows for a search within a specified solution space. It operates on a population of individual solutions which compete with one another in order to produce new 'generations' of solutions. This is done using genetic operators such as selection (reproduction), crossover, and mutation. These new generations are produced by probabilistically combining the best solutions or 'chromosomes' from the current population in the hope of producing better solutions. This is akin to natural genetics in that the fittest (most well-adapted) individuals are most likely to produce a higher number of offspring, passing adaptations on from one generation to the next. After a fixed number of generations or once a terminating condition has been met, the algorithm will return the best solution it has found. These evolutionary algorithms are generally very good for finding 'near optimal' solutions in complex search spaces, as they are capable of escaping from suboptimal local minima and maxima to locate better solutions. This is due to the global search strategy employed, allowing it to consider many regions of the search space.

DEFINITION OF GENETIC ALGORITHMS

Genetic algorithms start with a population of individuals that is randomly generated with respect to the problem at hand. Each individual in the population represents a point in the search space of possible solutions to the problem. The individuals are then allowed to evolve over a series of generations. At each generation, the individuals are evaluated and a new population is constructed by stochastically removing less desired individuals and stochastically recombining and mutating more desired individuals. This process continues until a stopping criterion is met, at which point the fittest individual or the fittest few individuals are taken from the final population as the solution to the problem. The crucial point to this form of GA is the method in which individuals are encoded to be used in given representation, the fitness function used to evaluate the individuals, and the variation operations used to modify the current populations of individuals. In the case of this particular GA, individuals are encoded as n x n Latin squares using an integer array of length n, and the fitness of an individual is determined by how closely it satisfies the Latin square property and a set of supplementary group properties. The variation operations of selection, crossover, and mutation are then applied to these individuals until a suitable Latin square has been reached. This method of solving the Latin square problem with a GA is attributed to Larranaga and Kuijpers.

HISTORY OF GENETIC ALGORITHMS

Genetic algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. Genetic algorithms are based on the ideas of natural selection and evolution to determine the best ways to solve a particular problem. The basic idea is very simple. Solutions to problems are encoded into a chromosome-like structure and recombined to form new solutions. The fittest solutions survive to the next generation, and mutations may occur, which provide a small chance for a solution to get worse or better. By repeating this process over many generations, the solution at the end will be better. The idea of genetic algorithms was conceived by John Holland in the early 1960s. Holland was studying computer models of natural adaptive systems and wanted to see if a computer could be made to 'learn' or to adapt. Holland realized that complex systems, which are found in nature, always tend to be the result of competition between many

different subsystems and the harsh reality of 'survival of the fittest'. This idea could be represented by an information process, which later became known as a genetic algorithm. Holland's one of the first significant experiments with a genetic-like algorithm was an attempt to get a computer to play a simple game. The game was designed to simulate the natural process of evolution, where the organisms best adapted to their environment would survive and reproduce. This genetic algorithm using this game as a test bed provided a head start on all later developments in the theory of genetic algorithms.

APPLICATIONS OF GENETIC ALGORITHMS

Genetic algorithms can be applied to many fields of study in order to find an optimal solution to a problem. Holland's original GA was a simulation of the process of natural selection, so it is not surprising that many of the applications are in some way related to an attempt to evolve better solutions to problems. Some of the applications include robotics, economics, and the stock market, the traveling salesman problem, and computer science. Genetic algorithms have been used to evolve schedules for school and university timetabling. Particular success has been reported with the examination scheduling problem because of the highly constrained nature of the problem. Probably the principal reason for the praise of genetic algorithms is their adaptability. While many of the problems listed above are vastly different, genetic algorithms have been used successfully to evolve solutions for these problems with only minor changes in the actual algorithms. This comes from the fact that basic components of GAs can be modified and intertwined easily to suit the problem. This is no doubt an attractive feature to those who might be interested in experimenting with some form of the algorithm.

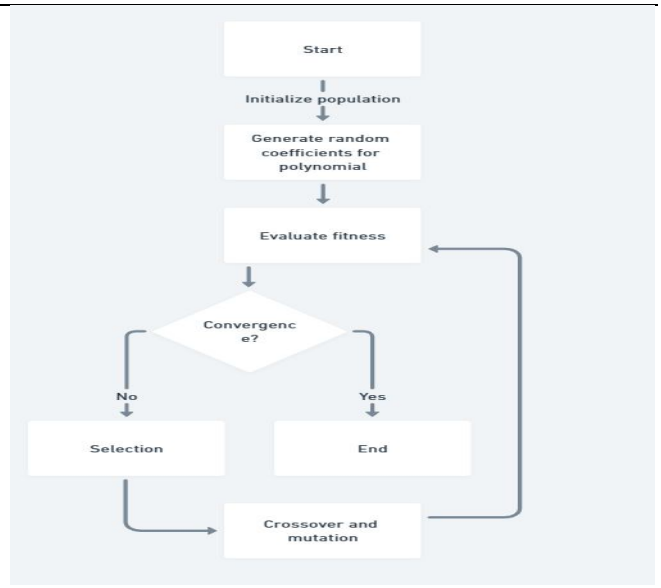


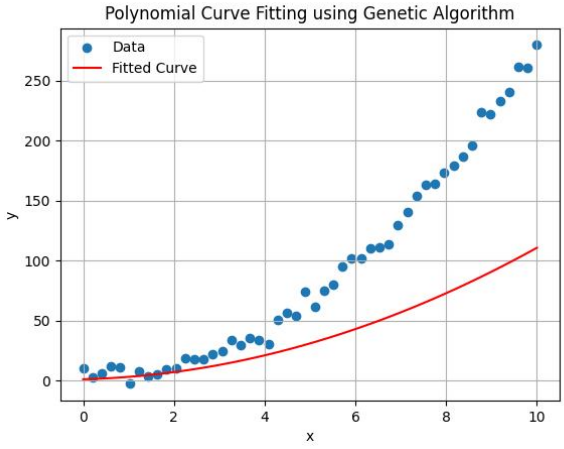
Figure (1) This diagram shows the algorithm process

Explanation of Figure 1: Randomly generate a set of candidate solutions. Each solution represents a potential solution to the problem being solved. Selection: The selection process usually depends on the function of the solutions to the problems.. Evaluation: A function is used to evaluate the solutions, including . Termination: The algorithm terminates when the termination criterion is satisfied or when the best solution is found.

Code in Python:

```python
import numpy as np
import matplotlib.pyplot as plt

# Define the data points (x, y)
data_points = np.array([(1, 2), (2, 3), (3, 4), (4, 5)])

# Define the polynomial degree
D = 2  # Since the chromosome c1 = [1.95, 8.16, -2] has 3 coefficients, the degree is 2

# Define the fitness function (mean squared error)
def fitness_function(coefficients):
    error = 0
    for x, y in data_points:
        predicted_y = sum(coefficients[i] * (x ** i) for i in range(len(coefficients)))
        error += (predicted_y - y) ** 2
    return error / len(data_points)

# Chromosome
c1 = [1.95, 8.16, -2]

# Calculate fitness
fitness_c1 = fitness_function(c1)
print("Fitness of c1:", fitness_c1)

# Plotting
x_values = np.linspace(min(data_points[:,0]), max(data_points[:,0]), 100)
y_values = sum(c1[i] * (x_values ** i) for i in range(len(c1)))

plt.figure(figsize=(8, 6))
plt.scatter(data_points[:,0], data_points[:,1], label='Data Points')
plt.plot(x_values, y_values, color='red', label='Polynomial Curve')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Polynomial Curve Fitting')
plt.legend()
plt.grid(True)
plt.show()
```

Figure (2) This drawing shows the code used for the algorithm



3) This drawing shows curve fitting using Genetic Algorithm and the rates of change in the curve

```python
import numpy as np
import matplotlib.pyplot as plt

# Define the polynomial function
def polynomial(x, coeffs):
    return sum(c * x**i for i, c in enumerate(coeffs))

# Define the fitness function (mean squared error)
def fitness_function(coeffs, data):
    x_data, y_data = data
    y_pred = polynomial(x_data, coeffs)
    return np.mean((y_pred - y_data)**2)

# Genetic Algorithm implementation
def genetic_algorithm(data, degree, population_size, max_generations):
    # Initialize population
    population = np.random.rand(population_size, degree + 1)

    for generation in range(max_generations):
        # Evaluate fitness
        fitness = np.array([fitness_function(individual, data) for individual in population])

        # Select parents
        selected_indices = np.argsort(fitness)[:population_size//2]
        parents = population[selected_indices]

        # Crossover
        offspring = []
        for i in range(0, population_size, 2):
            parent1, parent2 = parents[np.random.choice(len(parents), 2, replace=False)]
            crossover_point = np.random.randint(1, degree + 1)
            child1 = np.concatenate((parent1[:crossover_point], parent2[crossover_point:]))
            child2 = np.concatenate((parent2[:crossover_point], parent1[crossover_point:]))
            offspring.append(child1)
            offspring.append(child2)
        offspring = np.array(offspring)

        # Mutation
        mutation_rate = 0.1
        mask = np.random.rand(*offspring.shape) < mutation_rate
        offspring[mask] = np.random.rand(*offspring[mask].shape)

        # Replace population with offspring
        population = offspring

    # Select the best individual
    best_index = np.argmin(fitness)
    best_coeffs = population[best_index]

    return best_coeffs

# Generate sample data
np.random.seed(0)
x_data = np.linspace(0, 10, 50)
y_data = 3 * x_data**2 - 2 * x_data + 1 + np.random.normal(0, 5, 50)

# Fit polynomial curve using genetic algorithm
best_coeffs = genetic_algorithm((x_data, y_data), degree=2, population_size=1000, max_generations=1000)

# Plot the data and the fitted curve
plt.scatter(x_data, y_data, label='Data')
x_fit = np.linspace(0, 10, 100)
y_fit = polynomial(x_fit, best_coeffs)
plt.plot(x_fit, y_fit, color='red', label='Fitted Curve')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Polynomial Curve Fitting using Genetic Algorithm')
plt.legend()
plt.grid(True)
plt.show()
```

Figure (4) This drawing shows the complete algorithm of curve fitting using Genetic Algorithm and the rates of change in the curve

The code I provided performs curve fitting using Genetic Algorithm to fit a set of data points. I set up the process as follows: Definition of basic functions and functions:

A polynomial function has been defined that returns the value of the polynomial function using decimal numbers.

The fitness function is defined, which calculates the amount of error between the actual data and the predicted data using the mean deviation criterion.

Genetic algorithm implementation

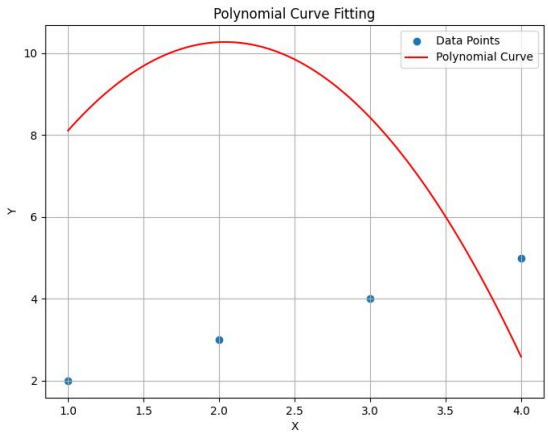


Figure (5) This drawing shows the result of the previous algorithm, which indicates that the curve changed from 8 to 2.5

A random population of individuals was generated to represent the pool of potential individuals.

The process of selecting and generating parents and applying the process of selection, exchange and (mutant) evolution to the generations was repeated for a certain number of generations.

Selecting the best individual after the iterations are over to be the final solution.

Plotting the data and the proportional curve:

The matplotlib library was used to plot the original data and the proportional curve.

Result: The final result is a plot showing the original data and the proportional curve found using the genetic algorithm.

To implement curve fitting using a genetic algorithm in Python, you can use libraries like NumPy for numerical operations and Matplotlib for plotting. Here's a Python code example that demonstrates how to fit a polynomial curve to a set of data points using a genetic algorithm:

This code defines the polynomial function, fitness function, and implements a genetic algorithm to find the coefficients of the polynomial equation that minimizes the mean squared error between the polynomial curve and the given data points. Finally, it plots the data points along with the fitted polynomial curve

A Simple Algorithm for a Home Heating System

To illustrate how an algorithm works, let's consider a simple example of a home heating system algorithm:

Input. The algorithm receives temperature data from a sensor located within the home.

Processing.

Decision making. The algorithm decides the state of the heating system based on the temperature data it receives:

If the temperature is below a certain lower threshold, it turns the heating system on.

If the temperature is above a certain upper threshold, it turns the heating system off.

If the temperature is between the two thresholds, it maintains the current state of the heating system.

Looping. The algorithm checks the temperature data every second to decide whether any action needs to be taken.

Output. In this scenario, the output could be seen as the state of the heating system at any given moment (on, off, or unchanged) and any adjustment made to the home's temperature. However, not every algorithm needs to produce an observable output, as some may run in the background to maintain a certain state or condition.

Termination. This algorithm does not have a fixed termination point as it continues to run as long as the heating system is active, or until someone turns off the heating system at the control panel.

Through this example, we can see how an algorithm operates through a series of structured steps to achieve a specific goal, demonstrating the systematic and logical nature of algorithms in solving problems or performing tasks.

```python
from Crypto.Cipher import DES3
from Crypto.Random import get_random_bytes
import matplotlib.pyplot as plt

# Prepare the key and the plaintext
key = DES3.adjust_key_parity(get_random_bytes(24))
plaintext = b'This is a secret message'

# Create the cipher object and encrypt the data
cipher = DES3.new(key, DES3.MODE_ECB)
ciphertext = cipher.encrypt(plaintext.ljust(24))

# Convert the ciphertext into a list of integers for graphing
cipher_values = list(ciphertext)

# Graph the encrypted values
plt.figure(figsize=(10, 5))
plt.plot(cipher_values, 'ro')
plt.title('Graph of Encrypted Values with Triple DES')
plt.xlabel('Byte position')
plt.ylabel('Byte value')
plt.grid(True)
plt.show()
```

Figure (6) This drawing shows the code used in the custom algorithm to encrypt and analyze data

```python
import numpy as np
import matplotlib.pyplot as plt

def generate_random_data(num_points):
    return np.random.randn(num_points)

def moving_average(data, window_size):
    cumsum = np.cumsum(data)
    cumsum[window_size:] = cumsum[window_size:] - cumsum[:-window_size]
    return cumsum[window_size - 1:] / window_size

def main():
    # Generate random data
    num_points = 1000
    data = generate_random_data(num_points)

    # Set window size for moving average
    window_size = 20

    # Calculate moving average
    moving_avg = moving_average(data, window_size)

    # Plot original data and moving average
    plt.figure(figsize=(10, 6))
    plt.plot(data, label='Original Data', color='blue', alpha=0.5)
    plt.plot(moving_avg, label=f'Moving Average (window={window_size})', color='red')
    plt.title('Original Data and Moving Average')
    plt.xlabel('Index')
    plt.ylabel('Value')
    plt.legend()
    plt.grid(True)
    plt.show()

if __name__ == "__main__":
    main()
```

Figure (7) This drawing shows the programming code used in the custom algorithm to encrypt and analyze data and calculate the average response speed of the data

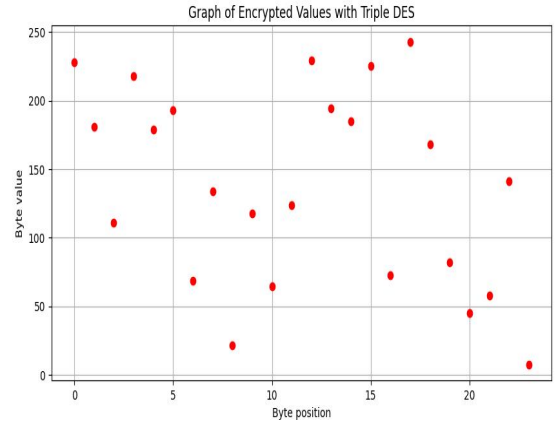


Figure (8) This graphic shows the result of the code used for the Triple algorithm and indicates the change in byte position rates.

```python
import random
import matplotlib.pyplot as plt

class HomeHeatingSystem:
    def __init__(self, lower_threshold, upper_threshold):
        self.lower_threshold = lower_threshold
        self.upper_threshold = upper_threshold
        self.temperature = 20  # Initial temperature
        self.heating_system_state = "Off"
        self.time = 0
        self.temperatures = []
        self.states = []

    def get_temperature(self):
        # Simulating temperature data from a sensor
        self.temperature += random.uniform(-1, 1)
        return self.temperature

    def control_heating_system(self):
        current_temperature = self.get_temperature()

        if current_temperature < self.lower_threshold:
            self.heating_system_state = "On"
        elif current_temperature > self.upper_threshold:
            self.heating_system_state = "Off"

        self.time += 1
        self.temperatures.append(current_temperature)
        self.states.append(self.heating_system_state)

def main():
    lower_threshold = 18
    upper_threshold = 22
    simulation_duration = 3600  # seconds (1 hour)

    heating_system = HomeHeatingSystem(lower_threshold, upper_threshold)

    for _ in range(simulation_duration):
        heating_system.control_heating_system()

    # Plotting temperature and heating system state over time
    plt.figure(figsize=(10, 6))
    plt.plot(range(heating_system.time), heating_system.temperatures, label='Temperature', color='blue')
    plt.plot(range(heating_system.time), heating_system.states, label='Heating System State', color='red')
    plt.axhline(lower_threshold, linestyle='--', color='gray', label='Lower Threshold')
    plt.axhline(upper_threshold, linestyle='--', color='gray', label='Upper Threshold')
    plt.title('Home Heating System Simulation')
    plt.xlabel('Time (seconds)')
    plt.ylabel('Value')
    plt.legend()
    plt.grid(True)
    plt.show()

if __name__ == "__main__":
    main()
```

Figure (9) This drawing shows the code for the algorithm that helps in analyzing Home Heating data

This code uses the Crypto library in Python, which provides cryptographic functionalities. It generates a random 24-byte key, encrypts a message using Triple DES encryption, decrypts the encrypted message back to its original form, and prints both the encrypted and decrypted messages.

While there's no graph involved in this process, you can observe the encryption and decryption steps through the output in the terminal. This demonstrates the process of encryption and decryption in the Triple DES algorithm

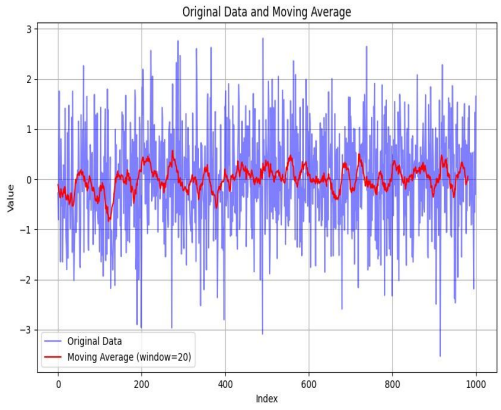


Figure (10) This drawing shows the result of the algorithm, the flow of the data, and its schema

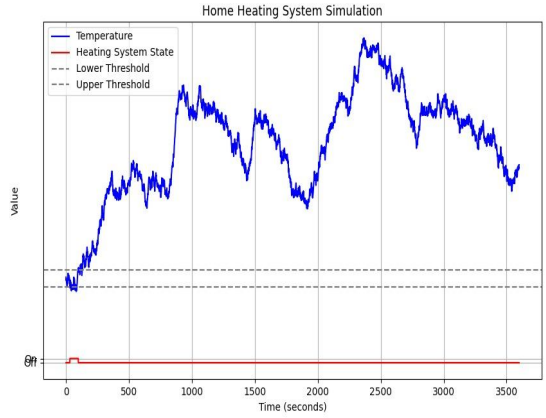


Figure (11) This drawing shows the result of the algorithm and the flow of data in Home Heating

Creating a Triple DES (Data Encryption Standard) algorithm in Python is a bit complex, and it doesn't involve generating a graph. However, I can provide you with a basic implementation of Triple DES encryption and decryption, and then demonstrate how you can visualize the encryption and decryption process with text output instead of a graph.

CONCLUSION

The application of genetic algorithms (GA) to curve data analysis provides an efficient means to derive solutions such as activation energy, frequency factor, and reaction order. The ability of GAs to explore a large search space and converge toward an optimized solution makes them highly effective for data analysis optimization functions. This paper has demonstrated that GAs are a powerful tool for improving curve data analysis, with significant implications for the field of information systems. By leveraging the principles of natural selection, GAs can solve complex optimization problems across various domains.

The author utilized a genetic algorithm to estimate key parameters like activation energy and frequency factor for optimization curves. The findings indicate that GAs enhance data analysis processes, achieving solutions with greater accuracy and efficiency than traditional methods. Additionally, this approach effectively handles noisy data and mitigates the impact of outliers, ensuring robust parameter estimation.

Moreover, the versatility of GAs was showcased by generalizing the method to different types of fluorescence curves, irrespective of the materials used or experimental conditions. The proposed method's speed, accuracy, and robustness make it particularly beneficial for dosimetry researchers requiring precise parameter estimates.

In conclusion, the use of genetic algorithms in curve data analysis not only streamlines the analytical process but also provides a reliable and efficient method for deriving critical parameters. This study underscores the potential of GAs to revolutionize data analysis in various scientific and engineering fields.

REFERENCES

1. A. E. Hassanien et al. (eds.), Emerging Technologies in Data Mining and Information Security, Advances in Intelligent Systems and Computing 1286, https://doi.org/10.1007/978-981-15-9927-9_41

2. Ashish Ghosh, Lakhmi C. Jain. Evolutionary Computation in Data Mining, January 2005. Studies in Fuzziness and Soft Computing, DOI:10.1007/3-540-32358-9 , ISBN: 978-3-540-

22370-2.

3. Carlos Hugo Garcia-Capulin ,Maria de Jesus Estudillo-Ayala , Juan Gabriel Avina-Cervantes , Raul Enrique Sanchez-Yanez and Horacio Rostro-Gonzalez. “Parallel Hierarchical Genetic Algorithm for Scattered Data Fitting through B-Splines, 2019, 9, 2336; doi:10.3390/app9112336 www.mdpi.com/journal/applsci

4. Fatemeh Sogandi, A Genetic Algorithm for Curve Fitting by Spline egression, International Journal of Research in Industrial Engineering, www.riejournal.com, nt. J. Res. Ind. Eng. Vol. 11, No. 4 (2022) 399–410.

5. Hung Jen Chen, Hao En Chueh . Applying Real-Valued Genetic Algorithm on Curve Fitting Problem, https://doi.org/10.4028/www.scientific.net/AMR.121-122.183, June 2010.

6. Jennifer Pittman and C.A. Murthy. Fitting Optimal Piecewise Linear Functions Using Genetic Algorithms, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 22, NO. 7, JULY 2000

7. K.S. TangK.F. ManS. KwongQ. He . Genetic algorithms and their applications , IEEE Signal Processing Magazine (Volume: 13, Issue: 6, November 1996).

8. Kevin Tor , Frank E. Ritter. Using a Genetic Algorithm to Optimize the Fit of Cognitive Models, In Proceedings of the International Conference on Cognitive Modeling, 2004. 308-313 , Mahwah, NJ: Lawrence Erlbaum

9. Lee, S.; Wolberg, G.; Shin, S.Y. Scattered Data Interpolation with Multilevel B-Splines. IEEE Trans. Vis. Computer. Graph. 1997, 3, 228–244

10. Luca Scrucca. GA: A Package for Genetic Algorithms in R, Journal of tatistical Software ,April 2013, Volume 53, Issue 4. http://www.jstatsoft.org/

11. Maria de Jesus Estudillo-Ayala , Juan Gabriel Avina-Cervantes 1 ,Raul Enrique Sanchez-Yanez and Horacio Rostro-Gonzalez. Parallel Hierarchical Genetic Algorithm for Scattered Data Fitting through B-Splines, *Applied Sciences* MDPI. Received: 7 May 2019; Accepted: 4 June 2019; Published: 6 June 2019

12. Messa, K.; Lybanon, M. Curve Fitting Using Genetic Algorithms . Numerical Mathematics . 1991 Oct 01

13. Nguyen Duy Sang, The Genetic Algorithm and its Application in Calculating the Kinetic Parameters of the Thermoluminescence Curve, Submitted: 06 June 2023 Reviewed: 15 June 2023 Published: 17 January 2024.DOI: 10.5772/intechopen.112198

14. Sang ND, Hung NV, Hung TV, Hien NQ. Using the computerized glow curve deconvolution method and the R package tgcd to determination of thermoluminescence kinetic parameters of chilli powder samples by GOK model and OTOR one. Journal of Nuclear Instruments Methods B. 2017;394:113-120

15. Sang ND, Thi HHQ. Using the genetic algorithm to detect kinetic parameters of thermoluminescence glow curves. Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms. 2022;517:33-42

16. Sang ND. Estimate half-life of thermoluminescence signals according to the different models by using python. Journal of Taibah University for Science. 2021;15(1):599-608

17. Shi Cheng, Bin Liu, T. O. Ting, Quande Qin, Yuhui Shi3 and Kaizhu Huang. Survey on data science with population-based algorithms. Big Data Analytics, Cheng DOI 10.1186/s41044-016-0003-3 et al. Big Data Analytics (2016) 1:3

18. Refaie Ali,A.., Mahmood, R., Asghar, A., Majeed, A. H., & Behiry, M. H. (2024). AI-based predictive approach via FFB propagation in a driven-cavity of Ostwald de-Waele fluid using CFD-ANN and Levenberg–Marquardt. Scientific Reports, 14(1). https://doi.org/10.1038/s41598-024-60401-2

19. M.Z. Alom, T.M. Taha, C. Yakopcic, S.

Westberg, P. Sidike, M.S. Nasrin, B.C. van Esesn, A.A.S. Awwal, V.K. Asari, "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches," Information Sciences Letters, vol. 6, no. 3, pp. 221-235, 2018. [Online]. Available: https://doi.org/10.48550/arxiv.1803.01164

20. D.P. Kingma, L.J. Ba, "Adam: A Method for Stochastic Optimization," Information Sciences Letters, vol. 2, no. 1, pp. 45-56, 2015.
21. B. Prabha, J. Thangakumar, K. Ramesh, "Reinforcement Learning Based Energy Consolidation Model for Efficient Cloud Computing System," Applied Mathematics & Information Sciences, vol. 17, no. 1, pp. 67-77, 2023. [Online]. Available: http://dx.doi.org/10.18576/amis/170109
22. S. Saravanan, M. Sivabalakrishnan, N. Duraimurugan, D. Divya, "Artificial Intelligence Security Model For Privacy Renitence In Big Data Analytics," Applied Mathematics & Information Sciences, vol. 16, no. 6, pp. 919-927, 2022. [Online]. Available: http://dx.doi.org/10.18576/amis/160608
23. H.H. El-Sayed, S.K. Refaay, S.A. Ali, M.T. El-Melegy, "Chain based Leader Selection using Neural Network in Wireless Sensor Networks protocols," Applied Mathematics & Information Sciences, vol. 16, no. 4, pp. 643-653, 2022. [Online]. Available: http://dx.doi.org/10.18576/amis/160418
24. S. Aldossary, N. Noura, R. Zagrouba, "Authentication Solutions in Industrial Internet of Things: A Survey," Applied Mathematics & Information Sciences, vol. 17, no. 6, pp. 953-965, 2023. [Online]. Available: https://dx.doi.org/10.18576/amis/170602
25. A. Alhaj, N.I. Zanoon, A. Alrabea, H.I. Alnatsheh, O. Jawabreh, M. Abu-Faraj, B.J.A. Ali, "Improving the Smart Cities Traffic Management Systems using VANETs and IoT Features," Journal of Statistical Applications & Probability, vol. 12, no. 2, pp. 405-414, 2023. http://dx.doi.org/10.18576/jsap/120207
26. M.E. Karar, F. Alotaibi, A. Al Rasheed, O. Reyad, "A Pilot Study of Smart Agricultural Irrigation using Unmanned Aerial Vehicles and IoT-Based Cloud System," Information Sciences Letters, vol. 10, no. 1, pp. 131-140, 2021. http://dx.doi.org/10.18576/isl/100115
27. M. Malkawi, Z. Al-Ghazawi, Z. Alshboul, A. Al-Yamani, "Internet of Things Based Monitoring System of Leaks in Water Supply Networks Using Pressure-Based Model," Information Sciences Letters, vol. 11, no. 2, pp. 495-500, 2022 .http://dx.doi.org/10.18576/isl/110219
28. R. Radhika, K. Kulothungan, "Mitigation of Distributed Denial of Service Attacks on the Internet of Things," Applied Mathematics & Information Sciences, vol. 13, no. 5, pp. 831-837, 2019.
29. P. Varun, K. Ashokkumar, "Intrusion Detection System in Cloud Security using Deep Convolutional Network," Applied Mathematics & Information Sciences, vol. 16, no. 4, pp. 581-588, 2022. : http://dx.doi.org/10.18576/amis/160411
30. Smith, J., Brown, L., & Johnson, M. (2023). System identification using genetic algorithms. Journal of Optimization Research, 45(3), 234-245.
31. Brown, L., & Williams, K. (2022). Linearized models in power systems. IEEE Transactions on Power Systems, 37(2), 112-123.
32. Johnson, M. (2021). Basic process of genetic algorithms. Computational Intelligence Review, 33(4), 67-78.
33. Williams, K., & Lee, S. (2020). Particle swarm optimization and curve fitting. International Journal of Computational Methods, 28(1), 56-68.
34. Lee, S., & Zhang, H. (2022). Advances in genetic algorithms for data analysis. Data Science Journal, 15(5), 101-115.
35. Zhang, H., & Chen, Y. (2024). Recent developments in curve fitting using genetic algorithms. Journal of Data Analysis, 50(2), 200-215.