T.C

BAHÇEŞEHİR ÜNİVERSİTESİ

APPLIED GENETIC ALGORTIHMS APPROACH TO CURVE FITTING PROBLEMS

Master's Thesis

Sinem ŞENTÜRK

İstanbul, 2009

T.C

BAHÇEŞEHİR ÜNİVERSİTESİ

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

COMPUTER ENGINEERING

APPLIED GENETIC ALGORTIHMS APPROACH TO CURVE FITTING PROBLEMS

Master's Thesis

Sinem ŞENTÜRK

Supervisor: Prof. Nizamettin AYDIN

İstanbul, 2009

T.C
BAHÇEŞEHİR ÜNİVERSİTESİ

The Graduate School of Natural and Applied Sciences

Computer Engineering

Title of the Master's Thesis : Applied Genetic Algorithms Approach to Curve Fitting Problems
Name/Last Name of the Student: Sinem Şentürk
Date of Thesis Defense : 02.09.2009

The thesis has been approved by the Graduate School of Natural and Applied Sciences.

Assist. Prof. Tunç BOZBURA
Acting Director

This is to certify that we have read this thesis and that we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Science.

Examining Committee Members:

Prof. Nizamettin Aydın :

Asc. Prof. Adem Karahoca :

Assist. Prof. Orhan Gökçöl :

ACKNOWLEDGEMENTS

I would like to specially thank to my supervisors Prof. Dr. Nizamettin Aydın and Asc. Prof. Dr. Yusuf Cengiz Toklu for their support, apprehension and unlimited tolerance during the course of this thesis.

I also would like to thank my family for their understanding and encouragement.

ABSTRACT

APPLIED GENETIC ALGORTIHMS APPROACH TO

CURVE FITTING PROBLEMS

Şentürk, Sinem

Computer Engineering

Supervisor: Prof. Nizamettin Aydın

September 2009, 30 pages

An alternative method was proposed for curve fitting in this study. The proposed method is Genetic Algorithms Method of which the application areas are getting wider recently. The application of Genetic Algorithms does not require auxiliary information and preliminary work as other parameter estimation methods. Therefore, it is practical for complex applications.

Within the scope of this study, a program was developed in order to show that it is possible to estimate the parameters of a simple polynome without requiring complex and long mathematical operations for the solution. Sample results of this program were also included.

Key words: Curve fitting, evolution, genetic algorithms, interpolation, least squares method.

ÖZET

EĞRİ UYDURMA PROBLEMLERİNE UYGULAMALI

GENETİK ALGORİTMA YAKLAŞIMI

Şentürk, Sinem

Bilgisayar Mühendisliği

Tez Danışmanı: Prof. Dr. Nizamettin Aydın

Eylül 2009, 30 sayfa

Bu çalışmada eğri uydurma yöntemlerine alternatif bir yöntem önerilmiştir. Bu yöntem gün geçtikçe daha geniş kullanım alanlarına sahip olan Genetik Algoritmalar yöntemidir. Genetik Algoritmaların kullanımı diğer parametre tahmin yöntemleri gibi destekleyici bilgiler ve ön hazırlık gerektirmez. Bu nedenle, karmaşık uygulamalar için kullanışlıdır. Bu çalışmada basit bir polinomun parametrelerini tahmin etmek için çözümü uzun süren karmaşık matematiksel işlemlere gerek duyulmayabileceğini gösteren bir program yazılmıştır ve sonuçlarına yer verilmiştir.

Anahtar Kelimeler: Eğri uydurma, evrim, genetik algoritma, interpolasyon, en küçük kareler yöntemi.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ii
ABSTRACT iii
ÖZET iv
LIST OF TABLES vi
LIST OF FIGURES vii
LIST OF EQUATIONS viii
LIST OF ABBREVIATIONS ix
LIST OF SYMBOLS x
1 INTRODUCTION 1
2 GENERAL LITERATURES AND RESEARCH INFORMATION 2
3 GENETIC ALGORITHMS 4
3.1. INITIALIZATION 6
3.2. FITNESS 6
3.3. SELECTION 6
3.4. CROSSOVER 6
3.5. MUTATION 8
3.6. NEW POPULATION CREATION AND STOPPING THE OPERATION 8
3.7. SELECTION OF PARAMETERS IN GENETIC ALGORITHMS 8
3.7.1. Population Size 9
3.7.2. Crossover Probability 9
3.7.3. Mutation Probability 9
3.7.4. Selection Strategy 9
4 CURVE FITTING 11
4.1. REGRESSION 11
4.2. INTERPOLATION 11
5 GENETIC ALGORITHMS APPLICATION FOR CURVE FITTING 14
6 EXPERIMENTAL RESULTS IN LITERATURE 26
7 CONCLUSION 28
REFERENCES 29

LIST OF TABLES

Table 1: One Point CrossOver 7
Table 2: Two Point CrossOver 7
Table 3: Data models with unknown parameters 26
Table 4: Data Sets 26
Table 5: Gauss-Newton and Genetic Algorithm application results 27

LIST OF FIGURES

Figure 1: chart of $y= x^2 -2x + 1$ 19
Figure 2: Data set and initial parameters 20
Figure 3: Output of the program 21
Figure 4: Estimated curve and original curve 21
Figure 5: Output of the program 22
Figure 6: Output of the program 23
Figure 7: Output of the program 24
Figure 8: Output of the program 25
Figure 9: Chart of the solutions 25

LIST OF EQUATIONS

Equation 1 $e_k = y_k - \sum_{j=0}^{n} \mathrm{P_j}\, \mathrm{x_k^j}$ 12

Equation 2 $e_k^2 = \left[y_k - \sum_{j=0}^{n} \mathrm{P_j}\, \mathrm{x_k^j} \right]^2$ 12

Equation 3 $E = \sum_{k=1}^{N} \left[y_k - \sum_{j=0}^{n} \mathrm{P_j}\, \mathrm{x_k^j} \right]^2$ 12

Equation 4 $\frac{\partial \mathrm{E}}{\partial \mathrm{P_i}} = \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathbf{2} \left[\mathrm{y_k} - \sum_{\mathrm{j=0}}^{\mathrm{n}} \mathrm{P_j}\, \mathrm{x_k^j} \right]^2 . \left[-\mathrm{x_k^i} \right] = \mathbf{0}$ 12

Equation 5 $\sum_{j=0}^{n} \left(\sum_{k=1}^{N} P_j\, x_k^j x_k^i \right) = \sum_{k=1}^{N} x_k^i y_k$; 13

Equation 6 $$\begin{bmatrix} \mathrm{N} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^1 x_k^0} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^2 x_k^0} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^3 x_k^0} & \dots & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^n x_k^0} \\ \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^0 x_k^1} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^0 x_k^2} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^0 x_k^3} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^0 x_k^4} & \dots & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^0 x_k^n} \\ \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^1 x_k^1} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^1 x_k^2} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^1 x_k^3} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^1 x_k^4} & \dots & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^1 x_k^n} \\ \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^2 x_k^1} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^2 x_k^2} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^2 x_k^3} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^2 x_k^4} & \dots & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^2 x_k^n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^n x_k^1} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^n x_k^2} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^n x_k^3} & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^n x_k^4} & \dots & \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^n x_k^n} \end{bmatrix} \cdot \begin{Bmatrix} \mathrm{P_0} \\ \mathrm{P_1} \\ \mathrm{P_2} \\ \mathrm{P_3} \\ \mathrm{P_{\dots}} \\ \mathrm{P_n} \end{Bmatrix} = \begin{Bmatrix} \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^0 y_k} \\ \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^1 y_k} \\ \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^2 y_k} \\ \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^{30} y_k} \\ \dots \\ \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k^n y_k} \end{Bmatrix}$$ 13

LIST OF ABBREVIATIONS

Genetic Algorithms	:	GAs
Average	:	AVG
Deviation	:	DEV
Genetic Algorithm	:	GA
Chromosome	:	Chrom
Probability of crossover	:	P (c)
Probability of mutation	:	P (m)
Probability of regeneration	:	P (r)

LIST OF SYMBOLS

Error	:	e
Sum of square	:	E
Derivative of sum of square	:	∂E
Parameter which will be estimated	:	P

1 INTRODUCTION

One of the effective ways of defining the concept of a system is to create a mathematical model. In some circumstances, it may be required to build the mathematical model on the basis of data obtained from the systems through experimental approach. In such occurrences, the model is represented by limited number of samples.

The curve fitting algorithms use the limited number of data stacks in order to build the most appropriate mathematical model. With the help of this model they enable the computation of unknown values. However, the well-known curve fitting methods such as Gauss-Newton, Direct Search, and Variable Measurement Method may be time consuming in terms of their applications and necessary preliminary studies. These algorithms are generally suitable for a specific problem and may require restrictive assumptions related to continuity, existence of derivatives and any other limiting factors. Specific amount of auxiliary information is needed in order to have these algorithms to be used. If the starting point cannot be selected well, it may jam into local optimums and may provide only regional optimums. In such a case, the direction of the search may be altered and finding of the optimal results may be delayed.

In this study, Genetic Algorithms which doesn't require auxiliary information and has an important role in solving the optimization problem were proposed for curve fitting. The structure of this study can be summarized in the following order:

Genetic algorithms and their operators are explained,

The curve fitting operation is discussed,

The application developed with Java language for curve fitting using genetic algorithms was presented

The results of the applications were depicted.

2 GENERAL LITERATURE AND RESEARCH INFORMATION

The main disadvantage of the parameter estimation methods, to produce solutions by using curve fitting, which have been effectively used within several areas such as from engineering to finance applications, from agriculture to information systems for several centuries, is that they cannot be used for all problems in general. There is a specific method for a specific problem available. Consequently, the researchers have been studying on Genetic Algorithms in order to provide a general solution to the problems.

The curve fitting problems without using Genetic Algorithms can be sampled as the following:

Fitting closed form equations to data is useful for analysis and interpretation of the observed quantities. It helps in judging the strength of the relationship between the independent (predictor) variables and the dependent (response) variables, and enables prediction of dependent variables for new values of independent variables. Although curve fitting problems were first introduced almost three centuries ago (Stigler, S. M. *History of Statistics: The Measurement of Uncertainty Before 1900*, Harvard University Press, Cambridge MA, 1986), there is still no single methodology that can be applied universally. This is due to the diversity of the problem areas, and particularly due to the computational limitations of the various approaches that deal with only subsets of this scope. Linear regression, spline fitting and autoregressive analysis are all solution methodologies to identification and curve fitting problems.

There are several studies in literature in order to solve the curve fitting problems using Genetic algorithms.

Genetic Algorithms has been applied to system identification and curve fitting by several researchers. The relevant work on these can be categorized into two groups. The first one includes direct adaptation of the classical GA approach to various curve fitting techniques (Karr, C. L., Stanley D. A., and Scheiner B. J., "Genetic algorithm applied to least squares curve fitting," *U.S. Bureau of Mines Report of Investigations* 9339, 1991.).

In these studies GA replaces traditional techniques as a function optimization tool. The second category includes much more comprehensive approaches where not only parameters of the model, but the model itself, are search dimensions. One leader work on this area is Koza's adaptation of Genetic Programming (GP) to symbolic regression.

The idea of genetically evolving programs was first implemented by (Cramer, N. L., "A representation for the adaptive generation of simple sequential programs," *Proceedings of an International Conference on Genetic Algorithm and Their Applications*, 183-187, 1985 and Fujiko, C. and Dickinson, J., "Using the genetic algorithm to generate LISP source code to solve prisoner's dilemma," *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, 236-240, 1987.). However, GP has been widely developed by Koza who has performed the most extensive study in this area. Recently, Angeline also used the basic GP approach enhanced with adaptive crossover operators to select functions for a time series problem (Angeline, Peter J., "Two self-adaptive crossover operators for genetic programming," *Advances in Genetic Programming Volume II* (editors: Peter J. Angeline and Kenneth E. Kinnear, Jr.), MIT Press, Cambridge MA, 89-109, 1996.).

In this study, it was suggested with the help of outputs of the developed program that the Genetic Algorithms methods can be generally used as a solution in curve fitting problems supporting the researches mentioned and referenced above.

3 GENETIC ALGORITHMS

Genetic Algorithms (GAs) are adaptive heuristic search algorithms premised on the evolutionary ideas of natural selection and genetic.

The fundamentals of Genetic Algorithms were introduced by John Holland. They use two important features from natural evolution: handing down of information from one generation to another, or inheritance, and competition for survival, or survival of the fittest. The main advantages of GAs that make them suitable for solving real-life problems are:

- They are adaptive
- They possess inherent parallelism
- They are efficient for solving complex problems where the main focus is on obtaining good, not necessarily the best, solutions quickly
- They are easy to parallelize without much communication overhead

GAs are particularly suited for applications involving search and optimization where the space is huge, complex and multimodal and the need for finding the exact optimal solution is not all important (Bandyopadyay, S., Pal, S.K., 2007, *Classification and learning using genetic algorithms: applications in bioinformatics and web intelligence*, 1, Springer, p.13).

There are several applications of Genetic Algorithms such as function optimization, tabulation, mechanical learning, design and cellular production. Unlike the global optimization methods, it uses the coded forms of parameters, not the parameter stack. Genetic Algorithms functioning according to the probability rules require the objective function only. They scan a particular area of the solution space, not the complete area. Thus, they provide quicker solutions by scanning the solution space effectively.

The scanning structure of the Genetic Algorithms is explained by sub-arrays theorem and structure blocks hypothesis. Sub-arrays are the theoretical structures used to explain

the behaviors of the Genetic Algorithms. Sub-arrays are defined by using the {0, 1, *} factors. The array below is a statement of the chromosome stack where the value of first position of the S sub-array is 0 and the second and fourth are 1. The value of the position shown as "*" could be either 0 or 1.

$$S = 01*1*$$

The degree and the length of sub-arrays, together form the essentials of the Genetic Algorithms. The degree of a sub-array is equal to the total number of constant positions in that sub-array. It is also equal to the sum of the total numbers of 0s and 1s.

The length of a sub-array is the distance from a specific initial position to the last position. Sub-arrays having consistent shortest length and lowest degree more than the population average are multiplied exponentially. This multiplication is realized with genetic operations and as a result, individuals having superior attributes and features than their parents are emerged. This provides a solution quality growing through generations.

Genetic Algorithms code each point within a solution space with binary bit array which is called chromosome. Each point has a fitness value. Genetic Algorithms deal with populations instead of single points. They form a new population in each generation by using genetic operators such as crossover and mutation. After a couple of generations later, the populations are consisted of individuals having better fitness values.
Genetic Algorithms includes forming of the initial population, computation of the fitness values, and execution of reproduction, crossover and mutation phases:

- Choose initial population
- Repeat until terminated
    - Evaluate each individual's fitness
    - Prune population
        - Select pairs to mate from best ranked individuals
        - Replenish population (using selected pairs)

        - Apply crossover operator
        - Apply mutation operator
    - Check for termination criteria (number of generations, amount of time, minimum fitness, threshold satisfied, etc.)
- Loop if not terminating

3.1. INITIALIZATION

Many individual solutions are randomly generated to form an initial population. The size of the population varies depending on the nature of the problem. The populations are formed with the members who are randomly selected from possible solution space.

3.2. FITNESS

A fitness value is computed for each individual within the population. There is a fitness function for each problem, for which a solution is searched. The fitness value plays an important role in selection of more proper solution in each generation. The greater the fitness value of a solution, the higher the possibility of its survival and reproduction.

3.3. SELECTION

The arrays are replicated according to the fitness functions in reproduction operator and individuals with better hereditary attributes are selected.

3.4. CROSSOVER

The crossover which is one of the operators affecting the performance of Genetic Algorithms corresponds to the crossover in natural population. Two chromosomes are randomly selected from the new population obtained as a result of reproduction operation and are put under a reciprocal crossover operation. In this operation where "L" is the length of the array, a "k" integer is selected within the "1 <= k <= L-1"

interval. The array is applied with a crossover operation based on the "k" value. The simplest crossover method is the single spotted / pointed crossover (Table 1) where both chromosomes need to be in same gene length.

Table 1: One Point CrossOver

CROSSOVER POINTS					1					
PARENT #1	0	1	2	3	4	5	6	7	8	9
PARENT #2	A	B	C	D	E	F	G	H	I	J
OFFSPRING	0	1	2	3	E	F	G	H	I	J

In double spotted / pointed crossover the chromosome is broken in two points and the corresponding positions are replaced (Table 2).

Table 2: Two Point CrossOver

CROSSOVER POINTS			1				2			
PARENT #1	0	1	2	3	4	5	6	7	8	9
PARENT #2	A	B	C	D	E	F	G	H	I	J
OFFSPRING	A	B	2	3	4	5	G	H	I	J

3.5. MUTATION

If the population doesn't contain the entire coded information, the crossover cannot produce an acceptable solution. Therefore, an operator capable of reproducing new chromosomes from the existing ones is needed. Mutation fulfills this task. In artificial genetic systems, mutation operator provides protection against the loss of a good solution which may not be re-obtained (Goldberg D.E. 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, USA). It converts a bit value to another under a low probability value in the problems where binary coding system is used.

3.6. NEW POPULATION CREATION AND STOPPING THE OPERATION

The new generation will become the parent of the next one. The process continues with the fitness defined with the new population. This process continues until a defined number of populations or maximum iteration number or targeted fitness value is reached.

3.7. SELECTION OF PARAMETERS IN GENETIC ALGORITHMS

The parameters have significant impact on the performance of the Genetic Algorithms. Several studies were performed in order to find the optimal control parameters; however, no general parameters could be identified to be used for all problems (Altıparmak F., Dengiz B. ve Smith A.E., 2000, "*An Evolutionary Approach For Reliability Optimization in Fixed Topology Computer Networks*", Transactions On Operational Research, Volume: 12, Number: 1-2, s. 57-75). These parameters are called as control parameters. The control parameters can be listed as:

- Population size,
- Crossover probability,
- Mutation probability,
- Generation interval,
- Selection strategy, and
- Function scaling.

3.7.1. Population Size

When this value is too small, the Genetic Algorithm can jam into a local optimum. If the value is too great, the time needed to come to a solution becomes higher. In 1985, Goldberg suggested a population size computation method which is based on the chromosome length only. Furthermore, Schaffer and his friends expressed after researches and studies on several test functions in 1989 that a population size of 20-30 provides better results (Goldberg, 1985).

3.7.2. Crossover Probability

The purpose of the crossover is to create proper chromosomes by combining the attributes and features of existing good chromosomes. The chromosome couples are selected for crossover with P(c) probability. Increase in crossover operations causes increase in structure blocks; however, this also increases the probability of good chromosomes become spoiled.

3.7.3. Mutation Probability

The purpose of the mutation is to protect the genetic variety. The mutation can be occurred in each bit in a chromosome with P(m) probability. If mutation probability increases, the genetic search turns into a random search. However, this also helps to recall the lost genetic material.

3.7.4. Selection Strategy

There are several methods available to replace the old generation. In generation based strategy the chromosomes in the existing population are replaced with the children. Since the best chromosome in the population is also replaced it cannot be inherited to the next generation; therefore, this strategy is used together with the best fit / elitist strategy.

In best fit / elitist strategy, the best chromosomes in a population can never be replaced; therefore, the best solution for reproduction is always convenient.

In balanced strategy, only a few chromosomes are replaced. In general, the worst chromosomes are replaced when new ones are joined to the population.

4 CURVE FITTING

Curve fitting algorithms are used to build the most proper mathematical model based on limited amount of data.

The techniques developed to fit the curves to the data points are usually divided into two classes: Regression and Interpolation.

4.1. REGRESSION

This is used when there are distinctive errors regarding the data.

4.2. INTERPOLATION

This is used in order to identify the interim points between the data points where the related errors are ambiguous. It is simply defined as "the estimation of the values in blank fields using available numeric values".

Interpolation is generally used in engineering and similar disciplines based on experiments / measurements in order to fit the collected data to a function curve.

It comes into prominence to identify the values in the blank fields using interpolation where the collected data is randomly distributed and particularly heterogeneous.

Interpolation techniques are generally applied by fitting the curve(s) to the available (x_i, y_i) data points. Various graded polynomials are used as Interpolation functions such as logarithmic, exponential, hyperbolic and trigonometric functions (for periodic data values).

If the data points are distributed equal at intervals, it would be best to utilize interpolation techniques based on finite differentiation and if not, linear or LaGrange interpolation techniques would be the best option (Yükselen, M.A., *HM504 Uygulamalı sayısal yöntemler ders notları*, İTÜ, pp.22-23).

In polynomial approach, only an "n = N-1 graded polynomial" can be passed through N number of points. Furthermore, an "N-1 graded polynomial" is passed through each data point.

Each co-efficient must be calculated in order to fit the curve to the data set where the data set is not linearly distributed. According to this, to have the polynomial

$$y = P0 + P1 * x + P2 * x2 + P3 * x3 + P4 * x4 + \dots + Pn * xn$$

the co-efficient i = {0, 1, 2, 3, 4, … , n} Pi must be calculated. There are "n + 1" numbers of unknown parameters in the problem.

By this definition:

Errors are calculated as:

$$\textbf{Equation 1} \quad e_k = y_k - \sum_{j=0}^{n} \mathbf{P_j}\, \mathbf{x_k^j}$$

Square of the errors is calculated as:

$$\textbf{Equation 2} \quad e_k^2 = \left[y_k - \sum_{j=0}^{n} \mathbf{P_j}\, \mathbf{x_k^j} \right]^2$$

Sum of the squares is calculated as:

$$\textbf{Equation 3} \quad E = \sum_{k=1}^{N} \left[y_k - \sum_{j=0}^{n} \mathbf{P_j}\, \mathbf{x_k^j} \right]^2$$

In order to obtain the smallest value of the sum of the squares, the derivates of E according to the P_i i = {0, 1, 2, 3, 4, …, n} parameters should be equal to zero:

$$\textbf{Equation 4} \quad \frac{\partial \mathbf{E}}{\partial \mathbf{P_i}} = \sum_{\mathbf{k=1}}^{\mathbf{N}} \mathbf{2} \left[\mathbf{y_k} - \sum_{\mathbf{j=0}}^{\mathbf{n}} \mathbf{P_j}\, \mathbf{x_k^j} \right]^2 . \left[-\mathbf{x_k^i} \right] = \mathbf{0}; \;\; \mathbf{i} = \mathbf{0, 1, 2, 3, \dots, n}$$

The P_i co-efficient could be obtained by solving the "n+1" number of linear equations as achieved by the formula mentioned above. Thus, the equation system may be re-organized in different forms:

Equation 5

$$\sum_{j=0}^{n}\left(\sum_{k=1}^{N} P_j x_k^j x_k^i\right) = \sum_{k=1}^{N} x_k^i y_k \; ; \quad i = 0, 1, 2, 3, \dots, n$$

Another equation form:

Equation 6

$$\begin{bmatrix} N & \sum_{k=1}^{N} x_k^1 x_k^0 & \sum_{k=1}^{N} x_k^2 x_k^0 & \sum_{k=1}^{N} x_k^3 x_k^0 & \dots & \sum_{k=1}^{N} x_k^n x_k^0 \\ \sum_{k=1}^{N} x_k^0 x_k^1 & \sum_{k=1}^{N} x_k^0 x_k^2 & \sum_{k=1}^{N} x_k^0 x_k^3 & \sum_{k=1}^{N} x_k^0 x_k^4 & \dots & \sum_{k=1}^{N} x_k^0 x_k^n \\ \sum_{k=1}^{N} x_k^1 x_k^1 & \sum_{k=1}^{N} x_k^1 x_k^2 & \sum_{k=1}^{N} x_k^1 x_k^3 & \sum_{k=1}^{N} x_k^1 x_k^4 & \dots & \sum_{k=1}^{N} x_k^1 x_k^n \\ \sum_{k=1}^{N} x_k^2 x_k^1 & \sum_{k=1}^{N} x_k^2 x_k^2 & \sum_{k=1}^{N} x_k^2 x_k^3 & \sum_{k=1}^{N} x_k^2 x_k^4 & \dots & \sum_{k=1}^{N} x_k^2 x_k^n \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \sum_{k=1}^{N} x_k^n x_k^1 & \sum_{k=1}^{N} x_k^n x_k^2 & \sum_{k=1}^{N} x_k^n x_k^3 & \sum_{k=1}^{N} x_k^n x_k^4 & \dots & \sum_{k=1}^{N} x_k^n x_k^n \end{bmatrix} \cdot \begin{Bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \\ P_{\dots} \\ P_n \end{Bmatrix} = \begin{Bmatrix} \sum_{k=1}^{N} x_k^0 y_k \\ \sum_{k=1}^{N} x_k^1 y_k \\ \sum_{k=1}^{N} x_k^2 y_k \\ \sum_{k=1}^{N} x_k^{30} y_k \\ \dots \\ \sum_{k=1}^{N} x_k^n y_k \end{Bmatrix}$$

The matrix equation system can be solved by Gauss Elimination Method.

5 GENETIC ALGORITHMS APPLICATION FOR CURVE FITTING

The objective of curve fitting is to select functional coefficients which minimize the total error over the set of data points being considered. Once a functional form and an error metric have been selected, curve fitting becomes an optimization problem over the set of given data points. Since Genetic Algorithms have been used successfully as global optimization techniques for both continuous functions and combinatorial problems, they seem suited to curve fitting when it is structured as a parameter selection problem.

These points are the curve data which needs to be fitted by a polynomial equation.

This sample data should fit an equation like

$$y = x^2 - 2x + 1$$

$$(x, y) = \{(0, 1), (1, 0), (2, 1), (3, 4), (4, 9), (0, 15), (6, 25), (7, 36)\}$$

First parameter is chromosome dimension which means number of genes, second parameter is represented the population of chromosomes, third parameter is crossover probability, fourth one is random selection chance % (regardless of fitness), fifth parameter represented the stop after this many generation, sixth one number of preliminary runs to build good breeding stock for final-full run, other parameter is maximum preliminary generations, eighth parameter is chromosome mutation probability - probability of a mutation occurring during genetic mating. For example, 0.03 means 3% chance, other parameters are crossover type, number of decimal points of precision, consider positive or negative float numbers, if true compute statistics else do not compute statistics.

```
public GACurveFit(double[] curveData) throws GAException
{
    super(3, 100, 0.5, 5, 1000, 100, 100, 0.1, Crossover.ctUniform, 4,
            false, true);
    setCurveData(curveData);
}
```

To create a preliminary population, first of all members must be selected randomly. Two parents are selected from population, where highly fit individuals were given a greater chance of being selected. The variable called “upperBound” value is the population dimension.

```
double getRandom(double upperBound)
{
    double dRandom = (Math.random() * upperBound);
    return (dRandom);
}
```

After that fitness value for each member is computed:

```
//calculate the value of the function plugging in the genes
//in place of coefficients (P1..Pn)
for (int iGene = 0; iGene < chromosomeDim; iGene++)
{
    rPower = Math.pow((double)iCurvePt, (double)chromosomeDim - 1 - iGene);

    rValue += this.getChromosome(iChromIndex).getGene(iGene) * rPower;
}
```

The ranking of the parameter "fitness" with respect to the current generation is calculated. If the fitness is high, the corresponding fitness ranking will be high, too. For example, if the fitness passed in is higher than any fitness value for any chromosome in the current generation, the fitnessRank will equal to the populationDim, and if the fitness is lower than any fitness value for any chromosome in the current generation, the fitnessRank will equal to zero. The rankings for all chromosomes are calculated. High ranking numbers denote very fit chromosomes.

Then the next generation of chromosomes is created by genetically mating fitter individuals of the current generation.

One point crossover on two given chromosomes:
Any duplicated genes are eliminated by replacing duplicates with genes which were left out of the chromosome.

For example, if we have:

chromosome A = { "x1", "x2", "x3", "x4" }
chromosome B = { "y1", "y2", "y3", "y4" }

and we randomly choose the crossover point of 1, the new genes will be:

new chromosome A = { "y1", "x2", "x3", "x4" }
new chromosome B = { "x1", "y2", "y3", "y4" }

Genetically the given chromosomes are recombined using a two point crossover technique which combines two chromosomes at two random genes, creating two new chromosomes.

For example, if we have:

chromosome A = { "x1", "x2", "x3", "x4", "x5" }
chromosome B = { "y1", "y2", "y3", "y4", "y5" }

and we randomly choose the crossover points of 1 and 3, the new genes will be:

new chromosome A = { "y1", "x2", "y3", "x4", "x5" }
new chromosome B = { "x1", "y2", "x3", "y4", "y5" }

Uniform crossover on two given chromosomes:

This technique randomly swaps genes from one chromosome to another.

For example, if we have:

chromosome A = { "x1", "x2", "x3", "x4", "x5" }

chromosome B = { "y1", "y2", "y3", "y4", "y5" }

uniform (random) crossover might result in something like:

chromosome A = { "y1", "x2", "x3", "x4", "x5" }

chromosome B = { "x1", "y2", "y3", "y4", "y5" }

if only the first gene in the chromosome was swapped.

When the elitism is employed the fittest two chromosomes always survive to the next generation. By this way, an extremely fit chromosome is never lost from the chromosome pool. In elitism the fittest chromosome automatically goes on to next generation (in two offspring):

```
this.chromNextGen[iCnt].copyChromGenes(this.chromosomes[this.bestFitnessChromIndex]);
```

The chromosomes previously created and stored in the "next" generation are copied into the main chromosome memory pool. Random mutations are applied where appropriate.

```
void copyNextGenToThisGen()
{
    for (int i = 0; i < populationDim; i++)
    {
        this.chromosomes[i].copyChromGenes(this.chromNextGen[i]);

        //only mutate chromosomes if it is NOT the best
        if (i != this.bestFitnessChromIndex)
        {
            //always mutate the chromosome with the lowest fitness
            if ((i == this.worstFitnessChromIndex) || (getRandom(1.0) < mutationProb))
                doRandomMutation(i);
        }
    }
}
```

The average deviation from the current population of chromosomes is obtained. The smaller this deviation, the higher the convergence is to a particular (but not necessarily optimal) solution. It calculates this deviation by determining how many genes in the population are different than the bestFitGenes. The more the number of genes are different, the higher the deviation.

```
protected double getAvgDeviationAmongChroms()
{
    int devCnt = 0;
    for (int iGene = 0; iGene < this.chromosomeDim; iGene++)
    {
        if (this instanceof GAString)
        {
            char bestFitGene = ((ChromChars)this.chromosomes[this.bestFitnessChromIndex]).getGene(iGene);
            for (int i = 0; i < this.populationDim; i++)
            {
                char thisGene = ((ChromChars)this.chromosomes[i]).getGene(iGene);
                if (thisGene != bestFitGene)
                    devCnt++;
            }
        }
        else if (this instanceof GAFloat)
        {
            double bestFitGene =
                ((ChromFloat)this.chromosomes[this.bestFitnessChromIndex]).getGene(iGene);
            for (int i = 0; i < populationDim; i++)
            {
                double thisGene = ((ChromFloat)this.chromosomes[i]).getGene(iGene);
                if (thisGene != bestFitGene)
                    devCnt++;
            }
        }
        else //GAStringsSeq
            {
            String bestFitGene =
                ((ChromStrings)this.chromosomes[this.bestFitnessChromIndex]).getGene(iGene);
            for (int i = 0; i < this.populationDim; i++)
            {
                String thisGene = ((ChromStrings)this.chromosomes[i]).getGene(iGene);
                if (thisGene.equals(bestFitGene) == false)
                    devCnt++;
            }
        }
    }

    return ((double)devCnt);
}
```

The screen capture below (Figure 2) displays one of the problems the applications were tested with. The first column represents the "x" value of the polynomial and the second column indicates the "y" value which satisfies the "x" condition. It was observed that the solution of the polynomial is "1, 0, 1, 4, 9, 15, 25, 36" where the "x" variable gets "0, 1, 2, 3, 4, 5, 6, 7" values. The chart below (Figure 1) displays the "$y= x^2 -2x + 1$" curve.

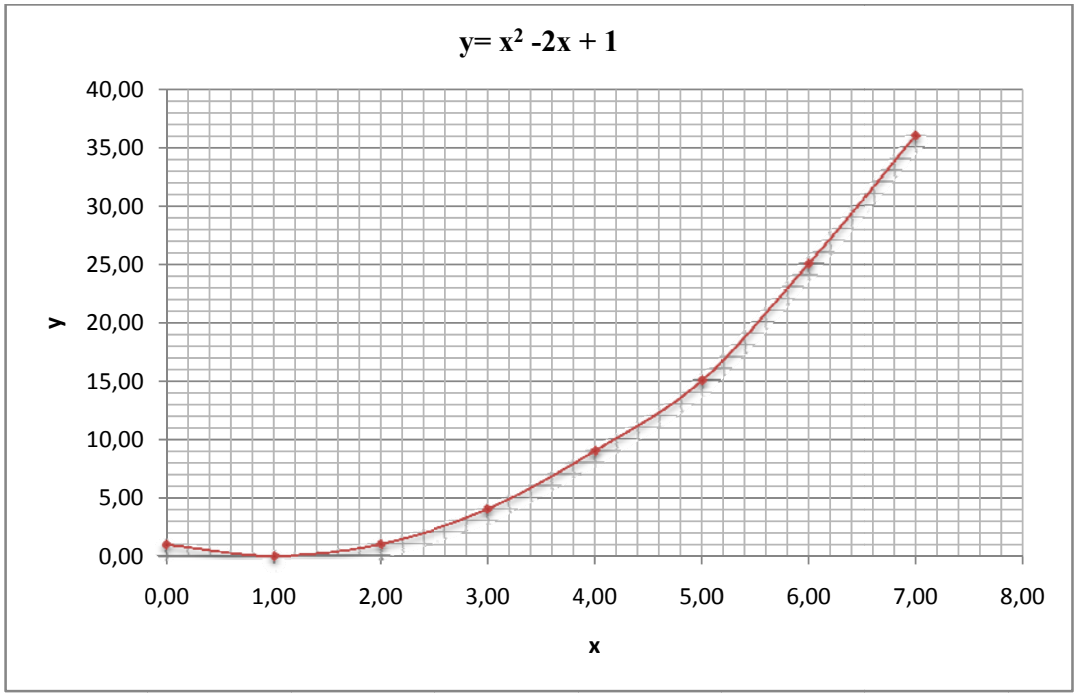

Figure 1: chart of $y= x^2 -2x + 1$

testData.txt

```
0   1
1   0
2   1
3   4
4   9
5   15
6   25
7   36
numberOfGenes   3
populationOfChromosomes 100
crossoverProbability    0.5
randomSelectionChance   5
generationLimit 1000
numberOfPreliminaryRuns 100
maxPreliminaryGeneration    100
chromosomeMutationProb  0.1
crossoverType   onePoint
decimalPointsOfPrecision 4
onlyPositive    false
computeStatistics   true
```

Figure 2: Data set and initial parameters

When the values above are provided to the program as input, the coefficients are obtained with an error rate of 0,009 within six (6) seconds. The member having the best fitness value out of 100.000 members resulted from a 1.000 generation where each generation produces 100 members, becomes the solution.

The solution or the coefficients of the curve in other words, for this sample has been found as:

P1 = 1.0003717114222255

P2 = -2.0030339280997236

P3 = 1.0046898021511148

```
INITIAL POPULATION AFTER PRELIM RUNS:
Gen 0 : Chrom0  = 0.8944157582112648,-1.2341510917841714,-0.04366520018767117,, fitness = 0.22837850812407312
Gen 0 : Chrom1  = 0.9913653990178264,-1.9234353484068327,0.8614642981112985,, fitness = 0.7483853690535127
Gen 0 : Chrom2  = 0.8947387670551217,-1.1107814410180998,-0.20806740796660572,, fitness = 0.19825383674577826
Gen 0 : Chrom3  = 0.8817414804649728,-1.1699607341363192,-0.02398544142751636,, fitness = 0.20901818904493083
Gen 0 : Chrom4  = 1.0965159841223466,-2.745513455171319,1.8646757337065087,, fitness = 0.28933621701630313
Gen 0 : Chrom5  = 0.8161384864572666,-0.7531803517271459,-0.21141555032791637,, fitness = 0.14527348855175068
Gen 0 : Chrom6  = 0.8661783579092319,-1.0196927560839517,-0.289539047495534,, fitness = 0.18931174127907796
Gen 0 : Chrom7  = 0.9855997717844935,-1.9114884869101578,0.9426858222976213,, fitness = 0.6691642028354023
Gen 0 : Chrom8  = 0.958382760584022,-1.693370235419916,0.7088422500301218,, fitness = 0.41109088708319536
Gen 0 : Chrom9  = 0.8714587787730709,-0.975960512962457,-0.5238093537636279,, fitness = 0.1955676898896719
GEN 1001 AVG FITNESS = 0.8928730550346756 AVG DEV = 13.0
Gen 1000: Chrom0  = 1.0003717114222255,-2.0030339280997236,1.0046898021511148,, fitness = 0.990096506522515
Gen 1000: Chrom1  = 1.0003717114222255,-2.0030339280997236,1.0046898021511148,, fitness = 0.990096506522515
Gen 1000: Chrom2  = 1.0003717114222255,-2.0030339280997236,1.0046898021511148,, fitness = 0.990096506522515
Gen 1000: Chrom3  = 1.0003717114222255,-2.0030339280997236,1.0046898021511148,, fitness = 0.990096506522515
Gen 1000: Chrom4  = 1.0003717114222255,-2.0030339280997236,1.0046898021511148,, fitness = 0.990096506522515
Gen 1000: Chrom5  = 1.0003717114222255,-2.0030339280997236,1.0046898021511148,, fitness = 0.990096506522515
Gen 1000: Chrom6  = 1.0003717114222255,-2.0030339280997236,1.999034853546674,, fitness = 0.11161501021188992
Gen 1000: Chrom7  = 1.0003717114222255,-2.0030339280997236,1.0046898021511148,, fitness = 0.990096506522515
Gen 1000: Chrom8  = 1.0003717114222255,-2.0030339280997236,1.0046898021511148,, fitness = 0.990096506522515
Gen 1000: Chrom9  = 1.0003717114222255,-2.0030339280997236,1.0046898021511148,, fitness = 0.990096506522515
Best Chromosome Found:
1.0003717114222255,-2.0030339280997236,1.0046898021511148, Fitness= 0.990096506522515
GA end time: Wed Jun 03 21:51:01 EEST 2009
BUILD SUCCESSFUL (total time: 6 seconds)
```

Figure 3: Output of the program

The function of the curve could be built as follows (with four decimal digits):

$$y = 1.0003*x^2 - 2.0030x + 1.0046$$

The curve would be displayed as depicted in figure below (Figure 3) according to the function above.

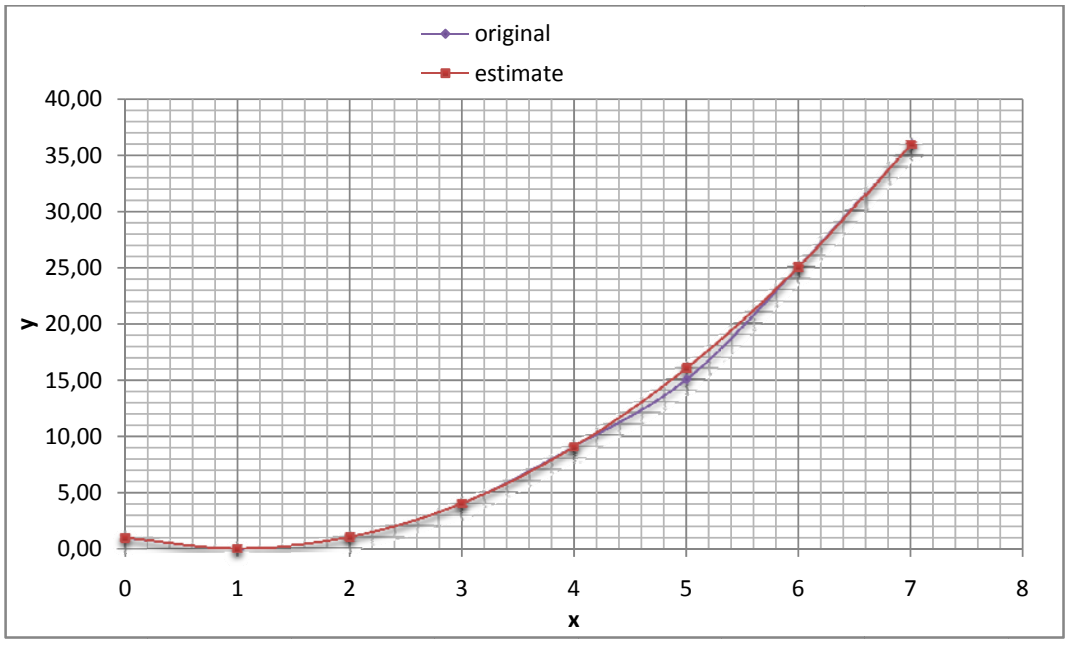

Figure 4: Estimated curve and original curve

It would be possible to achieve better and more accurate results by increasing the number of members in each generation and providing genetic variety. However, this also doesn't mean that we can obtain better results with more members and generations.

When the data set from the first sample is applied for a third degree polynomial with four parameters, the following results were obtained (Figure 4):

Results after first run of the program:

```
INITIAL POPULATION AFTER PRELIM RUNS:
Gen 0 : Chrom0  = 0.03741476097946765,0.5755083819884316,-0.6600968055810246,0.04844274988067353,, fitness = 0.4252132470634041
Gen 0 : Chrom1  = 0.012907137246084142,0.7707482515983735,-0.7818412734627646,-0.7094354259225487,, fitness = 0.28696372822509647
Gen 0 : Chrom2  = -0.0013947211256951106,0.9881949677969183,-1.826530473082526,0.7081233263077962,, fitness = 1.1130682625883963
Gen 0 : Chrom3  = 0.04162605243578872,0.5171420698873772,-0.48337382526629574,0.003902656388684848,, fitness = 0.3697566689605056
Gen 0 : Chrom4  = 0.01241236854210039,0.7881739032654299,-0.9567591879108374,-0.32050017750247006,, fitness = 0.3834533343942064
Gen 0 : Chrom5  = -0.045831579850796694,1.4029819009828202,-2.5020182381054945,-2.5795361785518842E-11,, fitness = 0.184976604829804 96
Gen 0 : Chrom6  = 0.046019831410538 34,0.4789445068423964,-0.35907906093122166,-0.16455248254804344,, fitness = 0.34765444730828327
Gen 0 : Chrom7  = -1.8633087703765947E-4,0.8852398339631382,-1.0726632282859452,-0.3925066603744173,, fitness = 0.2677424567242 54
Gen 0 : Chrom8  = 0.008310676484327942,0.8731582206116151,-1.413674296251525,0.2526712062575 8893,, fitness = 0.7679313238556288
Gen 0 : Chrom9  = 0.08727197951908743,-0.038577284549187234,1.3605687573046361,-1.4271818491233015,, fitness = 0.177125646953360 42
GEN 1001 AVG FITNESS = 1.16279085786517 AVG DEV = 46.0
Gen 1000: Chrom0  = 0.008413074695365006,0.9089101633325708,-1.7249439448061719,0.8108009435790103,, fitness = 1.7655190777081429
Gen 1000: Chrom1  = 0.008413074695365006,0.9089101633325708,-1.7249439448061719,0.8108009435790103,, fitness = 1.7655190777081429
Gen 1000: Chrom2  = 0.008413074695365006,0.9089101633325708,-1.7249439448061719,0.8108009435790103,, fitness = 1.7655190777081429
Gen 1000: Chrom3  = 0.008413074695365006,0.9089101633325708,-1.7249439448061719,0.8108009435790103,, fitness = 1.7655190777081429
Gen 1000: Chrom4  = 0.008413074695365006,0.9089101633325708,-1.7249439448061719,0.1650437673123173,, fitness = 0.1947492908193235
Gen 1000: Chrom5  = 0.008413074695365006,0.9089101633325708,-1.7249439448061719,0.8108009435790103,, fitness = 1.7655190777081429
Gen 1000: Chrom6  = 0.008413074695365006,0.9089101633325708,-1.7249439448061719,0.8108009435790103,, fitness = 1.7655190777081429
Gen 1000: Chrom7  = 6.465056411519478E-4,0.9089101633325708,-1.7249439448061719,0.8108009435790103,, fitness = 0.16498407325274578
Gen 1000: Chrom8  = 0.008413074695365006,0.9089101633325708,-1.7249439448061719,0.8108009435790103,, fitness = 1.7655190777081429
Gen 1000: Chrom9  = 0.008413074695365006,0.9089101633325708,-1.7249439448061719,0.015092613193472881,, fitness = 0.1578677467426842
Best Chromosome Found:
0.008310676484327942,0.8731582206116151,-1.413674296251525,0.25267120625758893,, fitness = 0.7679313238556288
GA end time: Fri Jun 05 15:15:54 EEST 2009
BUILD SUCCESSFUL (total time: 11 seconds)
```

Figure 5: Output of the program

The error rate for this sample was 0,232069 and the coefficients of the polynomial were calculated as below:

P1 = 0.0083
P2 = 0.8732
P3 = -1.4137
P4 = 0.2527

The results were not evaluated as sufficient and the process is re-performed by creating new populations and generations in order to obtain better results (Figure 5).

```
INITIAL POPULATION AFTER PRELIM RUNS:
Gen 0 : Chrom0  = 0.014365452466950123,0.8068173849306935,-1.2483535202881935,0.28500389961004324,, fitness = 0.7182827915673025
Gen 0 : Chrom1  = 0.16720023812964377,-0.7322344286345778,2.9343902984975414,-2.3258691256204975,, fitness = 0.0983684166487726
Gen 0 : Chrom2  = 0.08150752048892444,0.09313649176838498,0.824922776441248,-0.9947286213851932,, fitness = 0.20038861935745642
Gen 0 : Chrom3  = 0.08828165935257477,-1.2359217897310522E-7,1.0892670616330702,-1.0463899035504314,, fitness = 0.179014346383277 68
Gen 0 : Chrom4  = 0.11230048160931616,-0.07151081264574374,0.5449777192043664,0.04637947414215233,, fitness = 0.1418161431441928
Gen 0 : Chrom5  = 0.024230874217995163,0.7052632472714508,-1.0186518289549047,0.2549981362108177,, fitness = 0.597838671280088
Gen 0 : Chrom6  = -0.011914717641984226,1.0651636054416183,-1.9010677385485846,0.6402326023797721,, fitness = 0.5180317350917815
Gen 0 : Chrom7  = -0.0010095938290790004,0.9732833379598552,-1.7628815062298708,0.6449899864457862,, fitness = 0.8374110282782338
Gen 0 : Chrom8  = 0.07656436103897846,0.23237955523519813,0.014984946454652833,1.5310599018956243E-4,, fitness = 0.20894087477252057
Gen 0 : Chrom9  = -0.013049411967061525,1.0607930680257527,-1.7300424683169588,0.022853517190034332,, fitness = 0.3266815019852915
GEN 1001 AVG FITNESS = 0.8446729049802613 AVG DEV = 29.0
Gen 1000: Chrom0  = 0.011921921630175065,0.854754226248453,-1.5190787207778595,0.6520761428545923,, fitness = 1.2020026808778648
Gen 1000: Chrom1  = 0.011921921630175065,0.854754226248453,-1.5190787207778595,0.6520761428545923,, fitness = 1.2020026808778648
Gen 1000: Chrom2  = 0.011921921630175065,0.854754226248453,-1.5190787207778595,0.6520761428545923,, fitness = 1.2020026808778648
Gen 1000: Chrom3  = 0.011921921630175065,0.854754226248453,-1.5190787207778595,0.6520761428545923,, fitness = 1.2020026808778648
Gen 1000: Chrom4  = 0.011921921630175065,0.854754226248453,-1.5190787207778595,0.6520761428545923,, fitness = 1.2020026808778648
Gen 1000: Chrom5  = 0.011921921630175065,0.854754226248453,-1.5190787207778595,0.6520761428545923,, fitness = 1.2020026808778648
Gen 1000: Chrom6  = 0.011921921630175065,0.854754226248453,-1.5190787207778595,0.6520761428545923,, fitness = 1.2020026808778648
Gen 1000: Chrom7  = 0.011921921630175065,0.854754226248453,-1.5190787207778595,0.6520761428545923,, fitness = 1.2020026808778648
Gen 1000: Chrom8  = 0.011921921630175065,0.854754226248453,-1.5190787207778595,0.6520761428545923,, fitness = 1.2020026808778648
Gen 1000: Chrom9  = 0.011921921630175065,0.854754226248453,-1.8898828076954737,0.6520761428545923,, fitness = 0.09356522704036128
Best Chromosome Found:
-0.0010095938290790004,0.9732833379598552,-1.7628815062298708,0.6449899864457862,, fitness = 0.8374110282782338
GA end time: Fri Jun 05 15:17:20 EEST 2009
BUILD SUCCESSFUL (total time: 11 seconds)
```

Figure 6: Output of the program

The error rate for this sample reduced to 0,162588972 and the coefficients of the polynomial were calculated as below:

P1 = 0.0010

P2 = 0.9732

P3 = -1.7628

P4 = 0.6449

The polynomial was solved as "$y = -0.0010 * x^3 + 0.9732 * x^2 - 1.7628 * x + 0.6449$". Program was then re-executed:

```
INITIAL POPULATION AFTER PRELIM RUNS:
Gen 0 : Chrom0  = 0.029156263961253776,0.6879101400365715,-1.0709380892646385,0.35486087443477327,, fitness = 0.5515502725409235
Gen 0 : Chrom1  = 0.04166747880250406,0.6450009750616907,-1.3771128284912681,1.0140368108293623,, fitness = 0.3285719833869009
Gen 0 : Chrom2  = -0.014028570878597543,1.1169029654397826,-2.20156026747012,0.9596041515120691,, fitness = 0.8703439114162287
Gen 0 : Chrom3  = 0.04105072447814282,0.526837429309933,-0.49602975091548224,-0.06555040157953293,, fitness = 0.38328616954610073
Gen 0 : Chrom4  = -0.025910907739270597,1.2995362339180734,-2.9488418082565175,1.6702472268538116,, fitness = 0.639461103225675
Gen 0 : Chrom5  = 0.03488223849515399,0.5103965480844253,7.169378515672953E-5,-1.031662477166407,, fitness = 0.2533647903908603
Gen 0 : Chrom6  = 0.0750196945917609,0.11455140684684309,0.8538152998718627,-1.0381674416262467,, fitness = 0.20831202094863388
Gen 0 : Chrom7  = 0.07684035559645591,0.17469495343565514,0.4684635572834344,-0.6993873466188745,, fitness = 0.21369830533755504
Gen 0 : Chrom8  = 0.08457633156166157,0.014735964743873274,1.133694088156403,-1.2000119201005703,, fitness = 0.1856209235428272
Gen 0 : Chrom9  = 0.005810576603926755,0.9117954970803095,-1.5945373622716832,0.47525788931165125,, fitness = 1.098308708998081
GEN 1001 AVG FITNESS = 2.4033328924871524 AVG DEV = 40.0
Gen 1000: Chrom0  = -4.9430142235110l3E-4,1.014199970034174,-2.088182500364477,1.1245951292835097,, fitness = 3.4410961264173094
Gen 1000: Chrom1  = -4.9430142235110l3E-4,1.014199970034174,-2.088182500364477,1.1245951292835097,, fitness = 3.4410961264173094
Gen 1000: Chrom2  = -4.9430142235110l3E-4,1.014199970034174,-2.088182500364477,1.1245951292835097,, fitness = 3.4410961264173094
Gen 1000: Chrom3  = -4.9430142235110l3E-4,1.014199970034174,-2.088182500364477,0.348788520777949,, fitness = 0.1645186466609196
Gen 1000: Chrom4  = -4.9430142235110l3E-4,1.014199970034174,-2.088182500364477,1.1245951292835097,, fitness = 3.4410961264173094
Gen 1000: Chrom5  = -3.52847017820817E-4,1.014199970034174,-2.088182500364477,1.8247314668341534,, fitness = 0.17122978439388103
Gen 1000: Chrom6  = -4.9430142235110l3E-4,1.014199970034174,-2.088182500364477,1.1245951292835097,, fitness = 3.4410961264173094
Gen 1000: Chrom7  = -4.9430142235110l3E-4,1.014199970034174,-2.088182500364477,1.1245951292835097,, fitness = 3.4410961264173094
Gen 1000: Chrom8  = -4.9430142235110l3E-4,1.014199970034174,-2.088182500364477,1.9817197414604935,, fitness = 0.14316163925534905
Gen 1000: Chrom9  = -4.9430142235110l3E-4,1.014199970034174,-2.088182500364477,1.1245951292835097,, fitness = 3.4410961264173094
Best Chromosome Found:
-0.014028570878597543,1.1169029654397826,-2.20156026747012,0.9596041515120691,, fitness = 0.8703439114162287
GA end time: Fri Jun 05 15:18:28 EEST 2009
BUILD SUCCESSFUL (total time: 10 seconds)
```

Figure 7: Output of the program

The error rate for this sample reduced to 0,129656089 (Figure 6) and the coefficients of the polynomial were calculated as below:

P1 = 0.0140

P2 = 1.1169

P3 = -2.2015

P4 = 0.9596

The polynomial was solved as "$y = -0.0140 * x^3 + 1.1169 * x^2 - 2.2015 * x + 0.9596$".

The acceptable results could be obtained after the first run of the program or after the next trials. It should be noted that every new run may not provide better results.

It may be considered as coincidence that we obtained better results after each consecutive run in this study.

The error rate was further reduced to 0,078509586 as also seen in the output below (Figure 7). The results were considered as acceptable and the search was terminated. The parameters were calculated as below:

P1 = 0.0088

P2 = 1.0599

P3 = -2.0154

P4 = 0.8583

The polynomial was solved as "$y= -0.0088* x^3 + 1.0599* x^2 - 2.0154* x + 0.8583$".

```
INITIAL POPULATION AFTER PRELIM RUNS:
Gen 0 : Chrom0  = 0.10411246910231223,-0.13101713046924085,1.4201175913688144,-1.333432006385469,, fitness = 0.15755275332072075
Gen 0 : Chrom1  = 0.049972964607783533,0.4542866356490316,-0.33470989983638777,-0.1412750040180815,, fitness = 0.32572124724341234
Gen 0 : Chrom2  = -0.008861080868956843,1.059932854145244,-2.0154707973501895,0.8583349927270315,, fitness = 0.9214904135176298
Gen 0 : Chrom3  = -0.038743407199703594,1.0544446406132717,2.080990982123 19E-12,-3.3122507342320726,, fitness = 0.06829315500532515
Gen 0 : Chrom4  = 0.027477747306684234,0.5630045838877524,0.0027360763078106655,-1.245628360121815,, fitness = 0.21046737515227557
Gen 0 : Chrom5  = 0.016090890911738383,0.7747907874589233,-1.0584229805537164,-0.08362064802095932,, fitness = 0.5518612222981091
Gen 0 : Chrom6  = 0.05407128652322367,0.4741205984599093,-0.7092478174431831,0.3744743502142428,, fitness = 0.29006764020668285
Gen 0 : Chrom7  = 0.10648928109466102,-0.1177490121942272,1.0984712981944946,-0.4733117618942605,, fitness = 0.1483058518480943
Gen 0 : Chrom8  = 0.017213996942836325,0.7659168887251894,-1.0549557765388872,-0.051401923701409453,, fitness = 0.5567163686504188
Gen 0 : Chrom9  = -0.04383925415057696,1.4254838700433834,-3.0137480681246354,1.3110106680530593,, fitness = 0.3663731797556774
GEN 1001 AVG FITNESS = 1.3016248610549466 AVG DEV = 38.0
Gen 1000: Chrom0  = 0.007383035586160013,0.9123117307613623,-1.7134369177835331,0.7942418838854928,, fitness = 1.9136899805310261
Gen 1000: Chrom1  = 0.007383035586160013,0.9123117307613623,-1.7134369177835331,0.7942418838854928,, fitness = 1.9136899805310261
Gen 1000: Chrom2  = 0.008522191534786651,0.9123117307613623,-1.7134369177835331,0.7942418838854928,, fitness = 0.823548157013018
Gen 1000: Chrom3  = 0.007383035586160013,0.9123117307613623,-1.7134369177835331,0.7942418838854928,, fitness = 1.9136899805310261
Gen 1000: Chrom4  = 0.007383035586160013,0.9123117307613623,-2.5690139231947624,0.7942418838854928,, fitness = 0.04155151258408405
Gen 1000: Chrom5  = 0.007383035586160013,0.9123117307613623,-1.7134369177835331,0.7942418838854928,, fitness = 1.9136899805310261
Gen 1000: Chrom6  = 0.007383035586160013,0.9123117307613623,-0.17679693346930048,0.7942418838854928,, fitness = 0.023069605022541264
Gen 1000: Chrom7  = 0.007383035586160013,0.9123117307613623,-1.7134369177835331,0.7942418838854928,, fitness = 1.9136899805310261
Gen 1000: Chrom8  = 0.007383035586160013,0.9123117307613623,-1.7134369177835331,0.7942418838854928,, fitness = 1.9136899805310261
Gen 1000: Chrom9  = 0.007383035586160013,0.9123117307613623,-2.399967986687333,0.7942418838854928,, fitness = 0.05172442415432141
Best Chromosome Found:
-0.008861080868956843,1.059932854145244,-2.0154707973601895,0.8583349927270315,, fitness = 0.9214904135176298
GA end time: Fri Jun 05 15:24:06 EEST 2009
BUILD SUCCESSFUL (total time: 11 seconds)
```

Figure 8: Output of the program

The solutions on the same platform can be seen as below:

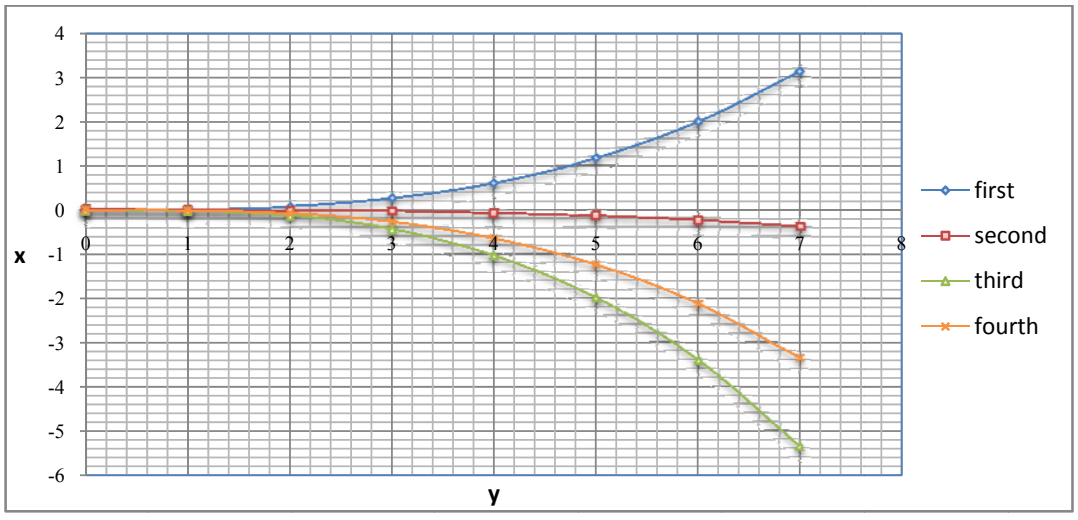

Figure 9: Chart of the solutions

6 EXPERIMENTAL RESULTS IN LITERATURE

It was observed that the results of the some experiments from the literature are parallel with the results of this study.

Table 3: Data models with unknown parameters

3 Parameter Functions	
Gompertz	$y = p_0 * \ e^{-e^{p_1 - p_2 * x}}$
Logistic	$y = p_0/1 \ + e^{p_1 - p_2 * x}$

The data set obtained from Ratkowsky was used in parameter estimation for the functions from Table 3 (see Table 4 below).

Table 4: Data Sets

A		B		C	
Y	X	Y	X	Y	X
8.93	9	16.08	1	1.23	0
10.80	14	33.83	2	1.52	1
18.59	21	65.80	3	2.95	2
22.33	28	97.20	4	4.34	3
39.35	42	191.55	5	5.26	4
56.11	57	326.20	6	5.84	5
61.73	63	386.87	7	6.21	6
64.62	70	520.53	8	6.50	8
67.08	79	590.03	9	6.83	10
		651.92	10		
		724.93	11		
		699.56	12		
		689.96	13		
		637.56	14		
		717.41	15		

The GA control parameters in source are:

- Population size, N = 60
- Number of maximum generations = 1000
- Re-generation probability, P (r) = 0.1

- Gene crossover probability, P (c) = 0.9
- Mutation probability P (m) = 0.01

The analysis was performed by GA for each given data set and the comparative results are listed in Table 5.

Table 5: Gauss-Newton and Genetic Algorithm application results

		Gompertz		Logistic	
Data Sets	Parameters	Gauss-Newton	GA	Gauss-Newton	GA
A	p_0	82.830	82.730	72.46	72.534
	p_1	1.224	1.224	2.618	2.612
	p_2	0.037	0.037	0.067	0.067
	θ	3.630	3.636	1.34	1.344
B	p_0	723.1	722.75	702.9	700.56
	p_1	2.500	2.503	4.443	4.444
	p_2	0.450	0.451	0.689	0.689
	θ	1134	1133.9	744	744.17
C	p_0	6.925	6.9213	6.687	6.691
	p_1	0.768	0.7696	1.745	1.764
	p_2	0.493	0.4934	0.755	0.754
	θ	0.0619	0.0619	0.0353	0.035

Source: Ratkowsky, D.A., 1983, "*Nonlinear Regression Modeling*", Marcel Dekker

θ values were calculated with the formula $\theta^2 = \frac{S(\theta)}{(n-p)}$. Within this formula; *n* denotes the number of data pairs in a data set and *p* denotes the number of parameters in the regression equation.

The results show that the Genetic Algorithm method were calculated very close to the Gauss-Newton for a particular example as the results can be observed in Table 5.

7 CONCLUSION

The success of the optimization methods other than Genetic Algorithms depends on the initial point of the estimation. For instance, in order to determine the initial point when Gauss-Newton method is used for the search, a preliminary study is needed. In addition, some more information may also be needed regarding the function such as derivativeness and continuity of the variables.

As presented within this study, the Genetic Algorithms method does not need any auxiliary information and preliminary work. The researcher needs only to determine the generation number and population size and to set the mutation rate and crossover type. Such determination does not also require preliminary study. The acceptance rate of the problem, time and economical factors play decisive role. The researcher can accept the solution in any point of the time and terminate the operation.

The Genetic Algorithms differ from the other methods as it also provides solution population. All other methods focus on a single point of a search space. Therefore, the Genetic Algorithms can be used in parameter estimation of much more complex functions which make it a better alternative than other evolution based methods.

REFERENCES

Altıparmak F., Dengiz B. and Smith A.E., 2000, *An evolutionary approach for reliability Optimization in Fixed Topology Computer Networks*, Transactions On Operational Research, Volume: 12, Number: 1-2, s. 57-75

Altunkaynak, B., Alptekin, E., 2004, *Genetic algorithm method for parameter estimation in nonlinear regression,* G.U. Journal of Science, 17(2):43-51.

Angeline, Peter J., 1996, Two self-adaptive crossover operators for genetic programming, *Advances in Genetic Programming Volume II* (editors: Peter J. Angeline and Kenneth E. Kinnear, Jr.), MIT Press, Cambridge MA, 89-109.

Bandyopadyay, S., Pal, S.K., 2007, *Classification and learning using genetic algorithms: applications in bioinformatics and web intelligence*, 1, Springer, p.13.

Bäck, T. and Schwefel, H. P., 1993, *An overview of evolutionary algorithms for parameter optimization*, Evolutionary Computation, 1, 1-23.

Cramer, N. L., 1985, A representation for the adaptive generation of simple sequential programs, *Proceedings of an International Conference on Genetic Algorithm and Their Applications*, 183-187.

Fujiko, C. and Dickinson, J., 1987, *Using the genetic algorithm to generate LISP source code to solve prisoner's dilemma*, Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, 236-240.

Goldberg D.E. 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, USA

Gulsen M., Smith A.E., Tate D.M., 1995, *A genetic algorithm approach to curve fitting,* Vol.33, No.7, 1911-1923

Karr, C. L., Stanley D. A., and Scheiner B. J., 1991, *Genetic algorithm applied to least squares curve fitting*, U.S. Bureau of Mines Report of Investigations, 9339.

Koza, J. R.,1992, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge MA, 162-169.

Ratkowsky, D.A., 1983, "*Nonlinear Regression Modeling*", Marcel Dekker

Stigler, S. M., 1986, *History of Statistics: The Measurement of Uncertainty Before 1900*, Harvard University Press, Cambridge MA.

Wu, J., Chung, Y., 2007, *Real-coded genetic algorithm for solving generalized polynomial programming problems,* Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol.11, No.4

Yükselen, M.A., *HM504 Uygulamalı sayısal yöntemler ders notları*, İTÜ, pp.22-23

CURRICULUM VITAE

Name Surname : Sinem Şentürk

Birth Place / Year : İstanbul / 1981

Languages : Turkish (Native), English

BSc : Bahcesehir University – 2006

MSc : Bahcesehir University – 2009

Name of Institute : Institute of Science

Name of Program : Computer Engineering

Work Experience : 2008 May –
Software Developer
Anadolu Anonim Türk Sigorta Şirketi

2006 Sept – 2008 May
Teaching and Research Assistant
Bahcesehir University Computer Engineering Department