Three-Dimensional Polynomial Regression Using Genetic Algorithm for Quasi Real-Time Application

Introduction:

Polynomial regression is a process of generating a model to estimate the value of an entity at any given point, within a reasonable margin of error. This can be done using many different methods, but Genetic Algorithm (GA) is an interesting approach.

GA is a means of search and optimization that draws its inspiration from nature (Goldberg, 2006). To simply put it, "a GA harnesses the mechanism of evolution"(Holland, 1992). It closes mimics the procedure involved in the natural selection and evolution of organic life viz. survival of the fittest, gene mutation, adaptation, etc. They utilize strings of binary data, where each bit signifies some characteristics of the problem. The algorithm starts a set of such strings in the beginning which is called the "Initial Population". These binary strings are tested for their fitness and the best set of strings are selected for a process called "Mating". Mating, like its biological counterpart, is a process of generating the next generation of strings by using a method called crossover. Here, a part of the two strings are interchanged to create new strings. The new strings replace the older strings that have a relatively low fitness value. Hence, a new generation is born. This procedure can be summarized as in Figure 1. The process is then repeated several times till the fitness values of all the strings in the population do not change much compared to the previous generation i.e. the solution converges.

These algorithms successfully search/optimize the problem due to the presence of a randomized "population" in the beginning. This sporadic population aides in searching the true optimized solution that some other optimization techniques might not find. Hence, GA are widely used in optimization of non-linear equations. The procedure involved aides in preserving the best fitness value. This method is iterative and hence time consuming. Its applications vary from aircraft design (Holland, 1992), VLSI design to schedule optimization (Ross & Corne, 1994). But its real time applications are nearly non-existent.

A real time application of polynomial regression would be Machine Learning (ML). ML is a phenomenon when a computer/device observes a repeated task and "learns" (Koza, Bennett, Andre, & Keane, 1996). ML uses statistical analysis tools such as polynomial regression to understand the factors and features present in the given data. The algorithm needs a lot of data to "learn" its features. This process is also called "training" the model. A lot of processing power is needed to train the model. With the devices available today, a portable real time application of the same is difficult. A way around this issue is to "train" the model on a workstation and deploy it to the real time device later. This leads to a decrease in the processing requirements of the real time device. This concept can be used as a quasi-real time ML system and applied to the field of prosthesis.

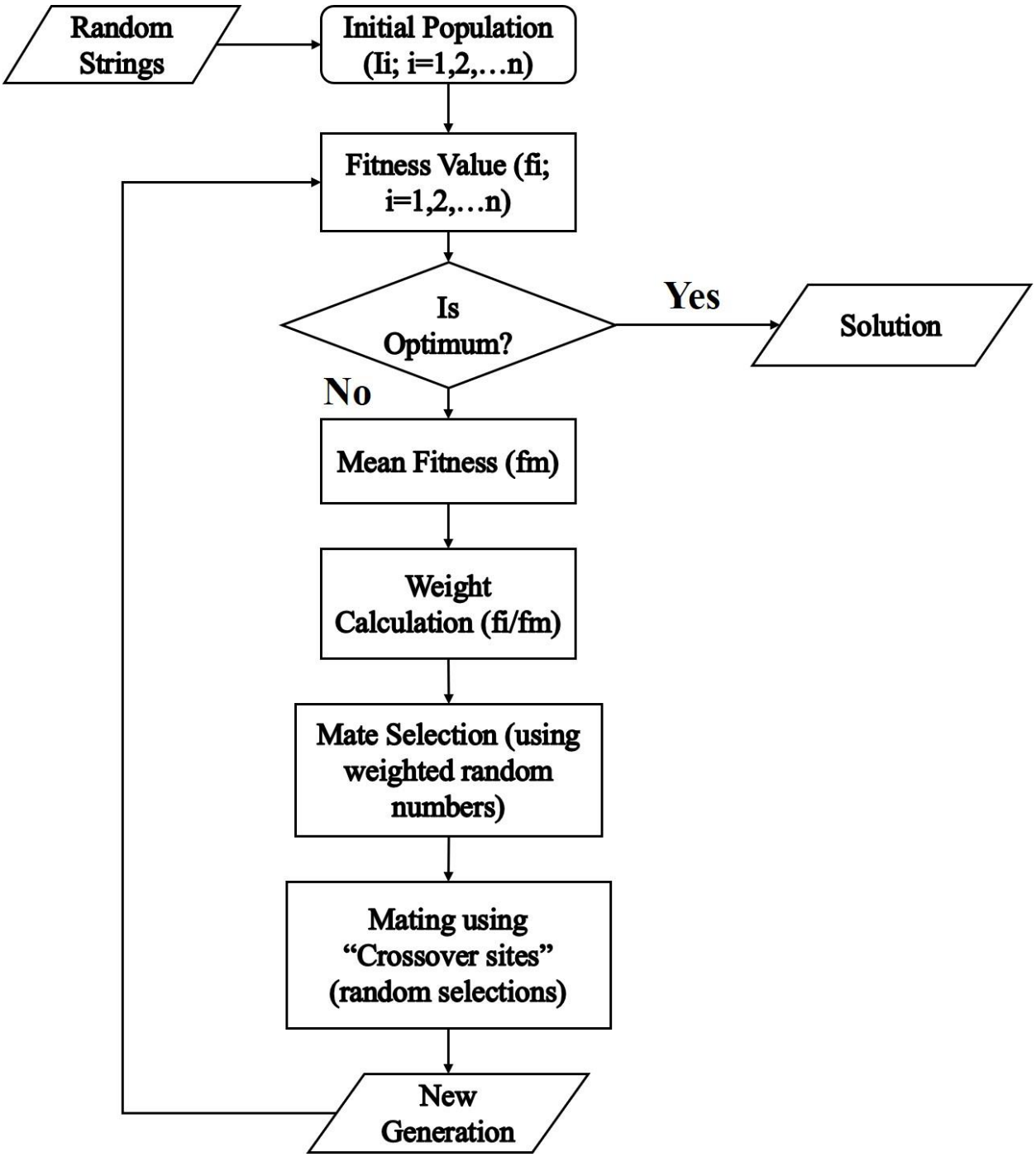

Figure 1. Genetic Algorithm Work-flow

System definition:

The system consists of a Prosthesis (controlled by a motor control system), IMUs (to collect the knee and ankle angle data), and a remote training center (workstation). The subject will walk while the good leg data will be collected for training. During training, the system input will consist of the gait percentage, ankle angle, and knee angle. Figure 2 shows the general process the Machine

learning system will follow. The model parameters obtained from the Genetic algorithm will be used to generate a mathematical model and deployed to the prosthesis. The output of the model will be used to control the prosthesis. Hence, a quasi-real time system will be created.

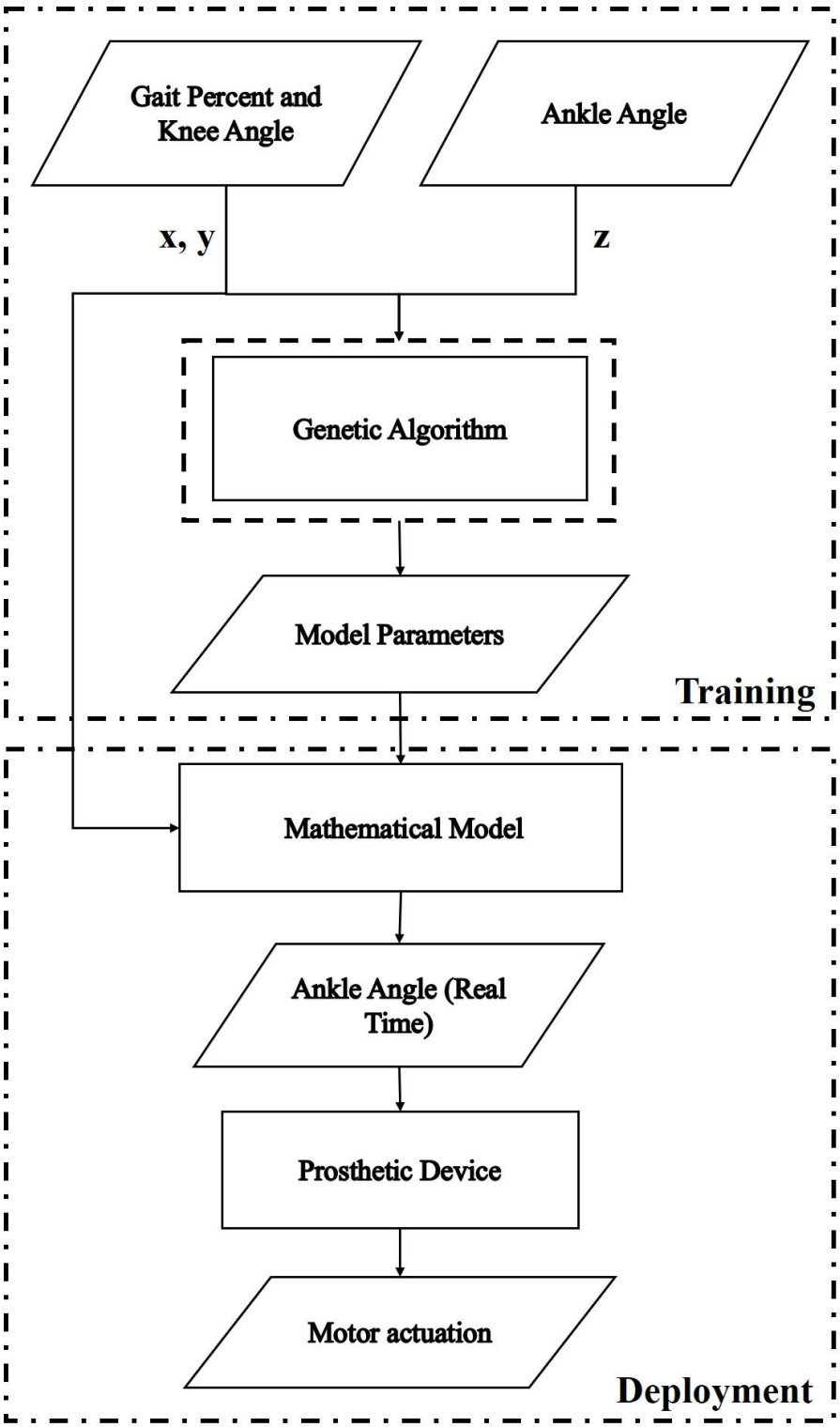

Figure 2. Machine Learning Work-flow

Regression model:

A mathematical model was selected to fit to the given data. The model was of order 4 as shown below:

$$z = p_1 + p_2x + p_3x^2 + p_4x^3 + p_5x^4 + p_6y + p_7y^2 + p_8y^3 + p_9y^4 + p_{10}xy + p_{11}x^2y + p_{12}xy^2 + p_{13}x^3y + p_{14}x^2y^2 + p_{15}xy^3 \quad (0.1)$$

Here, x is the gait cycle percent and y correspond to the knee angle in degrees. The output z is the ankle angle in degrees. p_i are the coefficients of the fit and will be selected by the GA.

Standard Form of the Optimization Problem:

Objective Function:

Minimize R.M.S. $[r_i - z_i]$

Where z will be calculated using **Error! Reference source not found.**, r is the given ankle angle data, and R.M.S. denotes the Root Mean Square of the difference between z and r.

Subject to:

$$-10.23 \leq p_j \leq 10.23$$

Where p_j are the coefficients of the model, and j = 1, 2, …., 15.

The optimization will be attempted and if a unique solution is not found, more constraints will be introduced.

Form of the Optimization Problem for Genetic Algorithm:

GAs will maximize the optimization problem. So, the standard problem must be modified in order to fit the code.

Hence the modified objective function is:

f_i = R.M.S. $[r_i - z_i]$

Maximize $\frac{(f_{i-1}max - f_i)}{f_{i-1}max}$

Where $f_{i-1}max$ is the maximum objective function from the previous generation. This scales the modified objective function from 0 to 1.

Application:

The probability of crossover was 0.95, for mutation was 0.01, and a normal fitness value and roulette-wheel selection was used. The following data were obtained using a non-normalized side constraint:

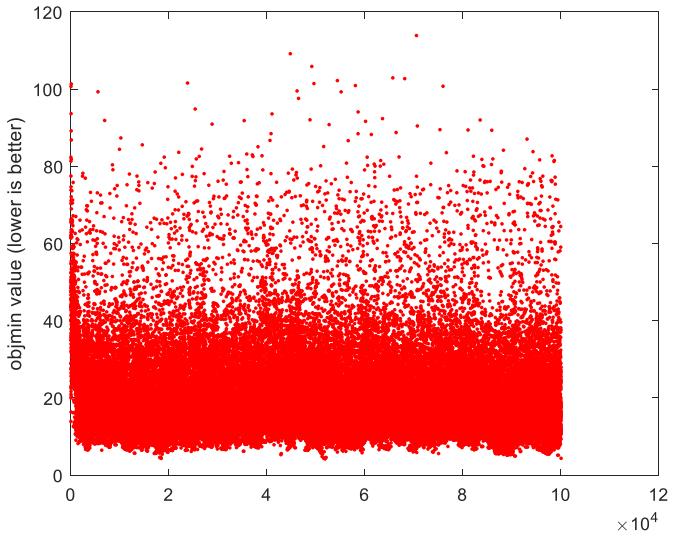

Figure 3. Plot of objective function value versus iterations for non-normalized side constraints.

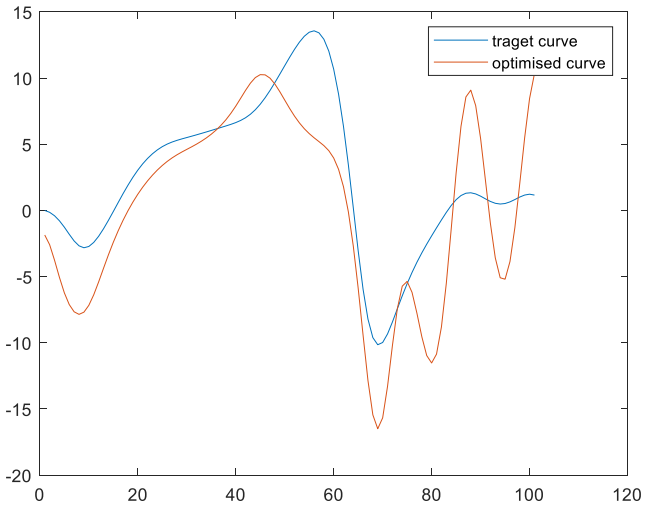

Figure 4. Plot of target curve and optimized curve for non-normalized side constraints.

The optimized curve had a higher R.M.S. error when compared to the target curve. The side constraints needed normalizing since the polynomial was sensitive to some design variables. Hence, the side constraints were changed by trial and error. The new side constraints were as follows:

$$9 < p_1 < 11;$$

$$3 < p_2 < 6;$$

$$-10 < p_3 < -9;$$

$$-3 < p_4 < -1;$$

$$-11 < p_5 < -9;$$

$$-7 < p_6 < -5;$$

$$-2 < p_7 < 0;$$

$$4 < p_8 < 6;$$

$$-3 < p_9 < -1;$$

$$2 < p_{10} < 4;$$

$$-3 < p_{11} < -1;$$

$$2 < p_{12} < 5;$$

$$8 < p_{13} < 10;$$

$$-4 < p_{14} < -3;$$

$$0 < p_{15} < 1;$$

This led to the following results:

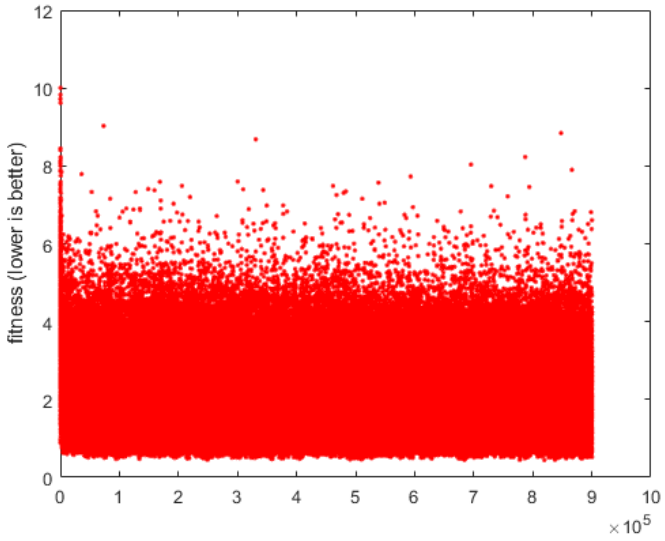

Figure 5. Plot of objective function value versus iterations for normalized side constraints.

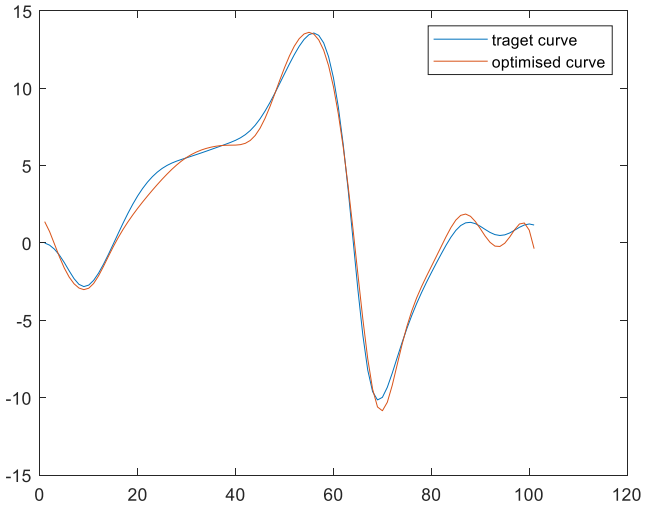

Figure 6. Plot of target curve and optimized curve for normalized side constraints.

The design variables jumped around due to the random nature of Genetic Algorithms. They converged around 1200th generation. Figure 7 shows the plot of the design variable history.

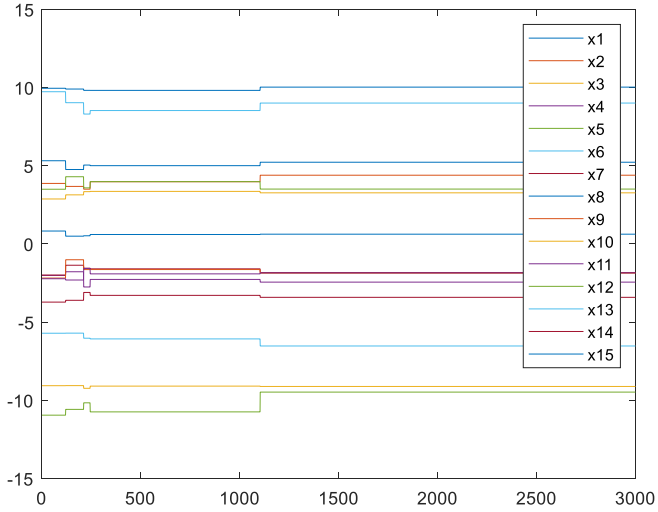

Figure 7. Plot of design variable history vs generations for normalized side constraints.

The converged values of the design variable were:

P = [1.003e+01, 4.396, -9.114, -2.441, -9.475, -6.525, -1.826, 5.226, -1.870, 3.269, -1.850, 3.510, 9.011, -3.414, 6.246e-01]

Conclusion:

The curve fitting was a success. The R.M.S. error was 0.5 after the Genetic Algorithm optimization. This proves the initial concept that genetic algorithms can be used to curve fit. It should be noted that there exist better techniques to achieve this. Genetic algorithm is a very slow

and resource intensive method. It is suitable for nonlinear optimizations where there exist multiple extremums. But still, the author explored the Genetic Algorithms and applied them to a problem while tackling the problems encountered.