

SPLINE CURVE APPROXIMATION AND DESIGN BY OPTIMAL CONTROL OVER THE KNOTS USING GENETIC ALGORITHMS

Rony Goldenthal*, Michel Bercovier*

*Hebrew University, School of Computer Science and Eng. Safra Campus, Jerusalem, 91904 , Israel
e-mail: {ronygold,berco}@cs.huji.ac.il, web page: <http://www.cs.huji.ac.il/~{ronygold,berco}>

Key words: Computer Aided Geometric Design, B-Splines, Optimal Control, Genetic Algorithms

Abstract. *In [1] Optimal Control methods over re-parametrization for curve and surface design were introduced. The advantage of Optimal Control over Global Minimization such as in [23] is that it can handle both approximation and interpolation. Moreover a cost function is introduced to implement a design objective (shortest curve, smoothest one etc...). The present work introduces the Optimal Control over the knot vectors of non- uniform B-Splines using genetic algorithm for the optimization of the cost function. Genetic algorithm allowed a simple implementation for highly non-linear problems, handled singularities and gave better results than classical gradient based methods on most of our examples.*

1 Introduction

Recall that a spline curve $C(s)$ is defined by control points (also known as de-Boor points) \mathbf{d}_i and a knot vector \mathbf{t} in the following way:

$$C(s) = \sum_{i=0}^{n-1} \mathbf{d}_i N_{i,k}(s) \quad (1)$$

where $a \leq s \leq b$, $N_{i,k}(s)$, the spline basis functions [9], are defined over the knot vector \mathbf{t} i.e.

$$N_{i,1}(t) = \begin{cases} 1, & \text{for } t_i \leq t < t_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

for $k = 1$ and

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)$$

for $k > 1$ and $i = [0, \dots, n-1]$

The following design problems will be examined:

Given a set of points $\mathbf{P} = \mathbf{P}_0, \dots, \mathbf{P}_{n-1} \in R^d$, $d = 2, 3$, generate a piece-wise-polynomial curve $C(s)$ that will solve one of the two design objectives:

- **Interpolation:** Interpolate through the given points (For a given parametrization $\mathbf{s} = (s_0, \dots, s_{n-1})$, create $C(s)$ s.t.

$$C(s_i) = \mathbf{P}_i \quad (2)$$

- **Approximation:** Find a curve that $C(s)$ that will pass as close as possible to the given set of points. The distance is measured at the parameter values $\mathbf{s} = (s_0, \dots, s_{n-1})$. The closest spline is defined by:

$$\min_{\mathbf{t}, \mathbf{s}} D(\mathbf{t}, \mathbf{s}) \quad (3)$$

where

$$D(\mathbf{s}, \mathbf{t}) = \sum_{i=0}^{n-1} \|C(s_i) - \mathbf{P}_i\|^2 \quad (4)$$

The curve $C(s)$ is composed of $N \ll n$ control points \mathbf{d}_i .

Both design problems have several degrees of freedom (d.o.f) - changing the knot vectors \mathbf{t} and the parameter values \mathbf{s} will create a different curves that will solve the design problem. The problem studied here is how to exploit these d.o.f in order to optimize a certain fairing goal such as to create an interpolating curve which's curvature is minimal, or to reduce the approximation error of the curve.

In this paper the framework suggested in [13] for solving the optimal control equations is replaced by a novel framework that uses genetic algorithm at the core of the optimization process. This framework works for all non-uniform spline curves of any order as well.

A specialized genetic algorithm was used for the optimization process to achieve better optimization results. The use of genetic algorithm enabled elegant and simple handling of a violation of the Schoenberg Whitney condition (a description of this condition appear in section 3.1.1 and in [4] and [22]), optimizing highly non-linear cost functions such as the elastic energy of the curve

or minimizing the approximation error under L_∞ norm, provided better results when compared to gradient based methods and removed the need for computing partial derivatives, which can be quite complicated and non-accurate in the case of the knot vector \mathbf{t} . Finally, the resulting code is simpler, shorter and more elegant than in the gradient based approach.

The paper organization is as follows: section 3 describes the mathematical aspects of this paper, Section 4 describes the genetic algorithm used in this work. Some results and examples are shown in section 5 followed by conclusions and future work in sections 6 and 7.

In all subsequent sections, a vector or a matrix will be denoted by a **bold** character.

2 Previous Works

The quality of the approximation and/or interpolation of a set of points $\mathbf{P}_j \in R^d$, ($d = 2, 3$) by a parametric curve $C(s)$ depends on the choice of parameters \mathbf{s} . Additional degrees of freedom in the case of spline curves are given by the knot vector \mathbf{t} .

The optimal parametrization problem has been extensively studied, see for instance Hoschek [16] and Speer et. al. [23] The placement of the knot vector has not been treated as an optimization problem, but rather as a local geometrical one, see Hoffman [15], Piegl [20] and Farin [8]. A global optimization problem where all parameters ($\mathbf{d}, \mathbf{s}, \mathbf{t}$) are treated at once together is given in the pioneer work of Gengoux and Mekhilef [10]. A different approach is introduced in Alhanaty and Bercovier [1] where the problem is defined in the setup of optimal control. This allows the separation of the state variable (for instance the control polygon) from the control variable (the parametrization). Moreover the algorithm deals with the classic case of control points and generation of the discrete parametrization defined at those points. In [13] the case of knot vector variations was first considered, the optimal control problem was solved using gradient based method this approach is somewhat limited due to the inability to work with highly non-linear cost functions and maintaining robust results is quite complicated and expensive.

A first attempt to use genetic algorithms in CAGD is given in [17] and [18], where it is limited to interpolation of Bezier curves and Hermite cubic splines, in addition the knot vector did not take part in the optimization process and therefore only part of the available degrees of freedom were exploited to find the optimal curve.

3 Mathematical Formulation

3.1 Optimal Control in CAGD

Optimal control in CAGD was previously presented in [1], here is a review of the formulation.

A control system is a system which is described by a variable \mathbf{x} , called the **state variable**. The variable \mathbf{x} depends on some **control variable** \mathbf{u} . Different values of \mathbf{u} define different states of the system, where the relation between \mathbf{x} and \mathbf{u} is given by the solution to a given linear equation that could involve differential equation

$$A(\mathbf{x}, \mathbf{u}) = 0 \tag{5}$$

This is the **state equation**.

The **cost function** $J(\mathbf{x}, \mathbf{u})$ is a real-valued function describing some quality of the pair (\mathbf{x}, \mathbf{u}) . It is often given by an integral functional. The **optimal control problem** is to find a control $\bar{\mathbf{u}}$ such

that any state $\bar{\mathbf{x}}$ for which $\mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = 0$ will minimize the cost function $J(\mathbf{x}, \mathbf{u})$, i.e.,

$$J(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \min_{(\mathbf{x}, \mathbf{u})} \{J(\mathbf{x}, \mathbf{u}) : \mathbf{A}(\mathbf{x}, \mathbf{u}) = 0\}.$$

3.1.1 State Equations and Variables

The state equation (equation 5) is either the interpolation equation or the approximation equation, both can be written as a system of linear equations:

$$\mathbf{M}(\mathbf{t}, \mathbf{s}) \times \mathbf{d} = \mathbf{P} \quad (6)$$

Where $\mathbf{P} = \mathbf{P}_0, \dots, \mathbf{P}_{n-1} \in R^d$, $d = 2, 3$, is a matrix of size $[n \times d]$. $\mathbf{d} = \mathbf{d}_0, \dots, \mathbf{d}_{N-1} \in R^d$, $d = 2, 3$, is the matrix of size $[N \times d]$ of the de-Boor points that define the curve. In case of interpolation $N = n$, in the case of approximation $N < n$. $\mathbf{M}(\mathbf{t}, \mathbf{s})$ is the collocation matrix that maps between the input points and the de-Boor control points. Consider the collocation matrix created for interpolation: given a parametrization \mathbf{s} and a knot vector \mathbf{t} one defines a curve passing through a given set of points i.e.

$$\mathbf{P}_i = C(s_i)$$

where

$$C(s_i) = \sum_{j=0}^{n-1} N_{j,k}(s_i) \mathbf{d}_j$$

Since there are n basis function and n parameter values one can organize the above equation in an $n \times n$ matrix \mathbf{M} , where $M_{i,j} = N_{j,k}(s_i)$. Note that \mathbf{M} is totally positive and banded with a semi-bandwidth less than k , the order of the spline (as was shown in [6]).

The Schoenberg-Whitney condition [22] requires that each parameter $s_j \in \mathbf{s}$, $0 \leq j < n$ will satisfy the following condition:

$$t_j \leq s_j \leq t_{j+k}$$

where k is the order of the spline and \mathbf{t} is the spline's knot vector. This is a necessary and sufficient condition for the interpolation and approximation to exist and to be unique. When this violation occurs the collocation matrix \mathbf{M} becomes singular, therefore the state equations cannot be solved and the cost associated with the particular configuration cannot be evaluated.

\mathbf{d} the de-Boor points of the curve $C(\mathbf{s})$ act as the state variable \mathbf{x} in this work (equation 5).

3.1.2 The Control Variables

In [1], the control variable \mathbf{u} , was chosen to be \mathbf{s} . In [13] \mathbf{t} was introduced as the control variable and experiments with alternating between \mathbf{t} and \mathbf{s} were conducted. Up to know this choice was not studied in CAGD because it results in a non-linear problem with possible singularities. In the current work the knot vector and the parametrization are optimized simultaneously in the unified approach described in section 4.1.2. Alternating between the knot vector and the parametrization has been tested here as well as described in section 4.1.1.

3.1.3 The Cost Function

The cost function $J(\mathbf{x}, \mathbf{u})$ is actually the design objective that one wishes to apply to the curve. In this paper, the following cost functions were implemented:

1. **Approximation error:** Measuring approximation error is norm specific, in gradient based methods usually the L_2 norm is used (see equation 4) because it is derivable. There are situations, however, in which measuring approximation error is more natural under the L_∞ norm. Using genetic algorithm both equations can be optimized. The approximation error under L_∞ norm is:

$$J_0 = \max(\|C(s_i) - \mathbf{P}_i\|), i = 0, \dots, m$$

2. **Length estimation:** Find the curve that approximates (or interpolates) the set of given points and has the shortest length. In practice the following equation to approximate the length of the curve was used:

$$J_1 = \int_a^b \|C'(s)\|^2 ds$$

The exact length measurement

$$J = \int_a^b \|C'(s)\| ds$$

has been successfully used as a cost function, but since both terms resulted with similar results, J_1 was preferred due to lower computational cost.

3. **Curvature estimation:** Find the curve that approximates (or interpolates) the set of given points and has the lowest curvature. In practice the following simpler functional was used.

$$J_2 = \int_a^b \|C''(s)\|^2 ds$$

4. **Elastic energy:** According to D. Bernoulli (1742) the elastic energy stored in the thin elastic beam is proportional to the integral (see [3]):

$$J_{3a} = \int_a^b \kappa^2(s) \|C'(s)\| ds$$

Where

$$\kappa(s) = \frac{|C(s)' \times C''(s)|}{|C'(s)|^3}$$

A desirable feature would be to minimize the maximal elastic energy over the whole curve:

$$J_{3b} = \max(\kappa^2(s) \|C'(s)\|), a \leq s \leq b$$

3.2 Remarks on geometry of the parametrization and geometry of the knot vector

The classical “optimal“ parametrization problem consider a **fixed** basis of spline curves, based on a fixed basis of spline functions - consider for instance the approximation problem, the non-linearity of the parametrization problem is actually due to the least square norm (equation 4), based on the parametrization nodes (\mathbf{t}). If the least square is performed on a large number of nodes, their influence diminishes as one gets to be close to the actual continuous L_2 norm. Since the basis functions are fixed, this continuous norm is independent of the choice of \mathbf{s} .

By contrast the knot vector geometry determines a given spline curve basis, changing the knots (up to a scaling factor) modifies the actual basis and as a consequence, it changes the functional space where the approximation (respectively the interpolation) is performed. This is equivalent to the idea of optimal node placement in the Finite Element Method ([24]) where the geometry defines the basis. Here the optimum will be related to the data, even for continuous norms such as L_2 and L_1 .

Experiments show that using both the knot vector and the parametrization for the optimization process provides better results than using either one of them by itself.

In addition to improving the optimization results there are several practical advantages for using the knot vector as a control variable:

1. There are situations in which the parametrization is fixed and therefore it cannot be used in order to achieve a design objective.
2. Low dimension: In global reparametrization the control points (de-Boor points (\mathbf{d})) were incorporated in solving the problem, while here the knot vector, (\mathbf{t}) is the control variable. The advantage of the present approach is that the dimension of the search space is lower - 1D vector instead of 2D/3D vector yielding two or three times more variables to optimize.
3. Efficiency: In approximation problems, there are N given points and n de-Boor points and $N \ll n$, the size of the classical re-parametrization problem is N compared to n when using the knot vector.

4 Framework Description

A genetic algorithm is an heuristic method that attempts to find the global optimum of a function by developing several potential solutions in parallel and repeatedly selecting the more fit individuals from a populations and reproducing them, in order to obtain even better results. The new offspring may be further altered by means of mutation. Each individual solution is encoded by one or more chromosomes. This encoding is problem specific, in the current work a chromosome is a vector of positive, real numbers, as will be explained later, for detailed definitions of genetic algorithms terms see [11].

In order to combine the optimal control framework with a genetic algorithm, a 'mapping' between the different terms must be made. The control variable, \mathbf{u} , should be the chromosome, since \mathbf{u} is the variable that is changed in order to find the optimal solution to the cost function. The fitness of the chromosome is determined by the cost function; the state equation must be solved for each chromosome.

Since both the knot vector \mathbf{t} and the parametrization \mathbf{s} can act as the control variable, \mathbf{u} , both were encoded as chromosomes. Two approaches were examined: The first one optimized each one

of the variables separately and alternately, for example: First the population of the knot vector has been optimized - using a constant parametrization. The best knot vector was selected and was used to optimize the parametrization population. This process was repeated in a loop several times. This approach is called the '*alternating approach*'.

The other approach, called the '*unified approach*', defines an entity with two chromosomes, the first is the knot vector, \mathbf{t} , and the second is the parametrization, \mathbf{s}). Using this approach both variables were optimized at once.

The alternating approach can work fine with smaller population size since the optimization problem is split into two smaller subspaces. The question of how many generations to use for each optimization step is not clear, but has a certain effect on the results. The second approach does not suffer from this problem. Moreover, since both the knot vector and the parametrization are evolved at once the results were generally better using this approach. Larger population must be used in the second approach for satisfactory results since the search space is larger. In practice, the two approaches differ only in the algorithm outline and in the cross-over and mutation operators.

4.1 Algorithm Outline

4.1.1 The Alternating Approach

Input for the algorithm:

- State equation to be solved (interpolation/approximation) (equation 5).
- Cost function to be used for the fitness evaluation (one of the cost functions $J_0 - J_3$).
- The inputs points $\mathbf{P} = \mathbf{P}_0, \dots, \mathbf{P}_{n-1} \in R^d (d = 2, 3)$.

The steps are:

1. Create initial 'seed' for the knot vectors, \mathbf{t} , population using the following heuristic methods: uniform, average and 'optimal'.
2. 'Grow' the knot vector population to the required size.
3. Create initial 'seed' for the parametrization, \mathbf{s} , population using the following heuristic methods: uniform, chord length and centripetal.
4. 'Grow' the parametrization population to the required size.
5. While there is an improvement (in the fitness of the best gene in the population) do:
 - (a) Advance several generation of the genetic algorithm while using the knot vector as a chromosome, while the most fit parametrization is treated as constant.
 - (b) Advance several generation of the genetic algorithm while using the parametrization as a chromosome, while the most fit knot vector is treated as constant.
6. Return the resulting curve - described by the resulting knot vector and parametrization.

4.1.2 The Unified Approach

Input for the algorithm:

- State equation to be solved (interpolation/approximation).
- Cost function to be used for the fitness evaluation.
- The inputs points $P = P_0, \dots, P_{n-1} \in R^d (d = 2, 3)$.

The steps are:

1. Create initial 'seed' for entities population, from m_1 heuristic methods to create knot vectors and m_2 methods to create parametrizations, $m_1 \times m_2$ initial entities will be created.
2. 'Grow' entities population to the required size.
3. While there is an improvement (in the fitness of the best gene in the population) do:
 - Advance a generation of the genetic algorithm while using both the parametrization and the knot vector as chromosomes.
4. Return the resulting curve - described by the resulting knot vector and parametrization.

4.2 Creating the Seed Population

In the CAGD literature, there are several heuristic methods for generating initial knot vectors and parametrization, non of them are optimal with respect to a certain cost function, though they produce reasonable results, and, usually, 'visually pleasing', curves (see for example [21]). Since none of the methods is optimal for all possible curves, several heuristic methods were used to create the initial seed population, this population was reproduced in order to create an initial population of the required size by means of crossover and mutations. Figure 4.2.2 shows examples of the different heuristic methods applied on the same input points.

4.2.1 Parametrization

The following methods were used to create the "seed" of the initial population. For a parametric curve in the range [a,b], with n input points $P = P_0, \dots, P_{n-1} \in R^d (d = 2, 3)$.

Uniform parametrization will be:

$$\begin{aligned} s_0 &= a \\ s_i &= a + i \frac{b-a}{n} \quad \text{for } 1 \leq i < n-1 \\ s_{n-1} &= b \end{aligned}$$

Chord length parametrization is:

$$\begin{aligned} s_0 &= a \\ s_i &= a + L_i(b-a) \quad \text{for } 1 \leq i < n-1 \\ s_{n-1} &= b \end{aligned}$$

Where

$$L_i = \frac{\sum_{k=1}^i \| \mathbf{d}_k - \mathbf{d}_{k-1} \|}{L}$$

$$L = \sum_{i=1}^{n-1} \| \mathbf{d}_i - \mathbf{d}_{i-1} \|$$

Centripetal parametrization is:

$$s_0 = a$$

$$s_i = a + L_i(b - a) \quad \text{for } 1 \leq i < n - 1$$

$$s_{n-1} = b$$

Where

$$L_i = \frac{\sum_{k=1}^i \| \mathbf{d}_k - \mathbf{d}_{k-1} \|^\alpha}{L}$$

$$L = \sum_{i=1}^{n-1} \| \mathbf{d}_i - \mathbf{d}_{i-1} \|^\alpha$$

$\alpha = 0.5$ was used.

4.2.2 Knot vector

For a parametric curve in the range $[a,b]$ of order k (degree $=k - 1$) with n input points the knot vector will have $m = n + k$ elements. The following methods were used to create the “seed” of the initial population.

Uniform knot vector

$$t_0, \dots, t_k = a$$

$$t_{i+k} = a + \frac{i(b-a)}{n+k-1} \quad \text{for } i = 1, \dots, n - k$$

$$t_{m-k}, \dots, t_m = b$$

Average knot vector creates knots by averaging the parameters values (s_i 's) in their neighborhood: given parametrization $\mathbf{s} = s_0, \dots, s_{n-1}$

$$t_0, \dots, t_k = a$$

$$t_{i+k} = \frac{1}{k} \sum_{j=i}^{i+k-1} s_j \quad \text{for } i = 1, \dots, n - k$$

$$t_{m-k}, \dots, t_m = b$$

'Optimal' knot vector is an iterative method that was suggested in [5] that creates a knot vector that is optimal with respect to a given parametrization, the method does not consider any cost function or even the location of the input points \mathbf{P} for the optimization. The Matlab function *optknt* has been used to produce this knot vector.

4.3 Gene Encoding

One of the challenges of adapting a problem to be solved using a genetic algorithm is to conceive encoding for the problem variable as genes that will be suitable for efficiently solving the optimization problem.

Since both the knot vector, \mathbf{t} , and the parametrization, \mathbf{s} , have effect on the shape of the curve, both are used as control variables. Therefore both were also experimented as chromosomes. By definition the knot vector and the parametrization must be monotone vectors, the number of elements in both remains constant during the optimization (we did not allow knot multiplication

$P_0 = (0.1, 0.1)$
 $P_1 = (0.3, 0.7)$
 $P_2 = (0.4, 0.7)$
 $P_3 = (0.4, 0.6)$
 $P_4 = (0.5, 0.6)$
 $P_5 = (0.6, 0.1)$

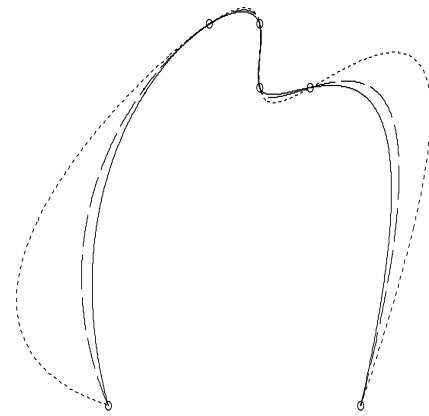
element	0	1	2	3	4	5
Uniform	0	0.2	0.4	0.6	0.8	1.0
Chord Length	0	0.44	0.51	0.58	0.65	1.0
Centripetal	0	0.32	0.45	0.58	0.71	1.0

(b) The resulting parametrizations

(a) The input points supplied by the user

element	0	1	2	3	4	5	6	7	8	9
Uniform	0	0	0	0	0.33	0.66	1	1	1	1
Average	0	0	0	0	0.45	0.58	1	1	1	1
Optimal	0	0	0	0	0.42	0.62	1	1	1	1

(c) The resulting knot vectors. Average and optimal methods used the centripetal parametrization as input.



(d) Examples for resulting curves, all the curves use the same centripetal parametrization, but differ on the knot generation method: the dashed uses uniform, the dotted uses average and the solid curve uses optimal knot generation

Figure 1: Heuristic methods to generate parametrization values, s , (figure 2(b)) and knot vector values, t , (figure 2(c)). Figure 1(d) demonstrates how different knot selection creates different curves.

during the optimization). For simplicity, the curve’s parametric length was fixed and curve’s knot started at zero and ended at the curve’s length. Because of the similarity in the requirements from s and t same encoding was used for both variables. The following encoding was used in this work: A chromosome was encoded by the intervals between two adjacent vector entries, i.e. each interval can be seen as single gene in the chromosome. The sum of all the intervals in a single chromosome was set to the curves’ length. This encoding enables creating mutation and crossover operators that generate valid curves more easily. For example the encoding of the knot vector

$$t = \{0, 0, 0, 0, 0.25, 0.7, 1, 1, 1, 1\}$$

is

$$et = \{0.25, 0.45, 0.3\}$$

Since the order and the parametric length of the curve are kept constant during optimization the first and last multiplied knots does not have to be represented explicitly. Note that even though both the knot vector and the parametrization were kept valid while optimized, the configuration between a certain knot vector and parametrization might become invalid due to violation of the Schoenberg-Whitney condition. This violation was not prevented by the defined operators, it was penalized by the fitness evaluation.

4.4 The Population

The initial population was generated by creating several reasonable individuals using the heuristic methods mentioned in section 4.2. Then the size of the population was increased by reproducing the seed population to the required size using crossover and mutation operators. The size of the population that was used was proportional to the size of the problem i.e. to the number of knots or parameter values. Because of the difference in the dimensions of the search spaces in the two approaches, the size of the population was different in the two approaches. In the unified approach a population of size $5n$ worked well, while in the alternating approach, population of size $2n$ sufficed (n is the number of input points).

4.5 Fitness Evaluation

One of the cost functions in section 3.1.3 was used for the fitness evaluation, the lower the cost of a certain individual, the higher its fitness will be. In order to evaluate the cost function the curve has to be created, i.e. the interpolation or approximation equation must be solved first. If the solution exists the cost function can be evaluated. Otherwise, a constant, low, fitness will be assigned to the invalid gene (this is equivalent to a regularization of a singular problems). As mentioned in section 3.1.1, a violation of the Schoenberg-Whitney's creates a singular configuration that cannot be solved. This way the invalid configurations can share their genes with other genes, but with a low probability. It is important not to totally eliminate these invalid genes because by doing that it would reduce the search space and the quality of the results may be reduced. Moreover, violation of the Schoenberg-Whitney condition is a local property, there may be segments of the curve that may be fit and it would be inefficient to ignore them. Segments of the curve can be considered schemates according to the definitions of Goldberg in [11].

When approximation was used as a state equation and the cost equation was other design objective (like curvature or length) both the approximation error and the design objective are to be minimized. This results with multiple fitness function optimization. Several methods to handle this situation were suggested in [2]. The simplest method of creating a linear combination of both the design objective and the approximation error worked well and almost any reasonable coefficient that scaled both terms to comparable sizes produced good results with regard to both the design objective and the approximation error.

4.6 Selection Strategy

Selection strategy defines how to select two individuals for reproduction, i.e. to act as parents for new offsprings. The 'Darwinian' principle of 'Survival of the fittest' guides the selection various strategies to prefer the more fit individuals to pass their genes to the next generation. On the other hand, selecting only the fit individuals to pass their genes may cause premature convergence of

the population on a local minima. An example of a popular strategy is the *tournament* selection strategy (see /citegoldberg1) in which two indices, i_1 and i_2 are randomly selected; then another random number, $0 \leq r \leq 1$, is selected. If $r < P_s$, the individual with the higher fitness is selected; otherwise, the other individual is selected. This strategy with $P_s = 0.75$ worked well for our examples. Other methods, such as Ranking, Sigma Scaling and Roulette Wheel were experimented as well and provided similar results (see [19] for overview). Tournament is the least expensive among all the methods in terms of computational cost.

Elitism, introduced by De Jong in [7], was used in addition to the selection strategy to force the GA to keep its most fit individuals at each generation.

4.7 Crossover

In nature, during sexual reproduction, recombination of chromosomes occurs: a single new chromosome is combined from the chromosomes of both parents. Crossover is the operator that defines how a new offspring is created from two parents during reproduction phase. After two parents were selected, crossover was performed with high probability, P_c , usually set to 0.7. A variation of one point crossover was used: Randomly select an index, i , $1 < i < N$. Then create the first offspring by copying all the intervals from the first parent in the ranges $[0, \dots, i]$, then copy the range $[i + 1, \dots, N]$ from the second parent. The weakness of this strategy is that the sum of the intervals of the new offspring won't match the curve's length (as required in section 4.3). In order to deal with this problem, all the intervals were normalized so that their sum will match the curve's length. Even though this strategy creates offspring that does not resemble exactly either of the parents for a part of the gene, it does keep the proportions between adjacent knot vector entries or parametrization entries. In the unified approach the decision to perform crossover on either one of the chromosomes was done separately. Even though this crossover operator distorts schemas, it does keep the general shape of the curve segments because the normalization process does not modify the intervals values drastically. The effect of this operator can be seen in figure 4.8.

4.8 Mutation

The mutation operator role is to add new variability to the population to the current population by altering the new offspring created by the crossover operator. Uncontrolled mutation scheme can create a chromosome that doesn't represent a valid curve. In order for a chromosome to represent a valid knot vector or parametrization, the chromosome must contain only positive values that sum to the curve's parametric length. For these reasons the following mutation strategy was used: first randomly select two indices i, j into the vector of intervals (the chromosome), then select a random number $r : 0 \leq r \leq \min(u(i), u(j))$; next subtract r from $u(i)$ and add it to $u(j)$. Mutation is applied with probability $P_m = 0.2$, otherwise the new gene is left unchanged. In the unified approach the decision to perform mutation on either one of the chromosomes was done separately.

After both offspring are complete (both are after the crossover and mutation phase) the fitness of both offspring is calculated and the stronger one is added to the population with probability 0.8.

4.9 Example

Assume we have the following knot vectors to be used as genes:

$$\mathbf{t}_0 = \{0, 0, 0, 0, 0.2194, 0.3386, 0.4473, 0.5552, 0.6640, 0.7826, 1, 1, 1, 1\}$$

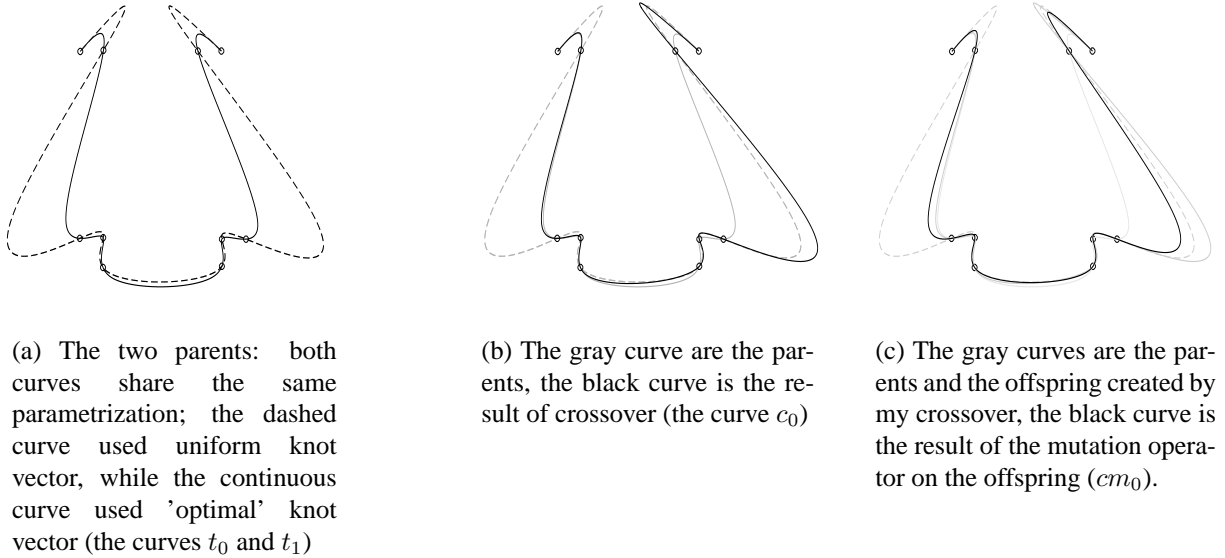


Figure 2: Example of crossover and mutation operators on the knot vector of the curves from section 4.9

and

$$t_1 = \{0, 0, 0, 0, 0.1429, 0.2857, 0.4286, 0.5714, 0.7143, 0.8571, 1, 1, 1, 1\}$$

their encoding is

$$et_0 = \{0.2194, 0.1191, 0.1087, 0.1078, 0.1088, 0.1186, 0.2174\}$$

and

$$et_1 = \{0.1429, 0.1429, 0.1429, 0.1429, 0.1429, 0.1429, 0.1429\}$$

Assume that $i = 3$ was selected as the crossover split point. The two offspring will be (before normalization)

$$ec_0 = \{0.2194, 0.1191, 0.1087, 0.1078, 0.1429, 0.1429, 0.1429\}$$

and

$$ec_1 = \{0.1429, 0.1429, 0.1429, 0.1429, 0.1088, 0.1186, 0.2174\}$$

After normalization the offsprings will become:

$$ec_0 = \{0.2231, 0.1211, 0.1105, 0.1096, 0.1452, 0.1452, 0.1452\}$$

and

$$ec_1 = \{0.1406, 0.1406, 0.1406, 0.1406, 0.1071, 0.1167, 0.2140\}$$

For the mutation of ec_0 indices $i_0 = 0$ and $i_1 = 6$ were randomly selected. A random number, x , in the range $0 \leq x < \min(0.1406, 0.2140)$ was selected to be $x = 0.0628$. x was subtracted from 0.214 and added to 0.1429, Resulting with

$$ecm_0 = \{0.1512, 0.1406, 0.1406, 0.1406, 0.1071, 0.1167, 0.2033\}$$

ecm_0 represents the following knot vector:

$$cm_0 = \{0, 0, 0, 0, 0.1512, 0.2918, 0.4323, 0.5729, 0.6800, 0.7967, 1, 1, 1, 1\}$$

The curve that the knot vectors t_0 , t_1 , c_0 and cm_0 are displayed in figure 4.8.

5 Results

Figure 3(a) displays a turbine blade's cross section. The cross section contains 140 points and is approximated by 28 points, using a 6th order spline, the cost function used was J_2 . The initial curvature estimation was 8049 while the final was 4155. The approximation error has also been reduced from 0.11 to 0.027 under L_2 and from 0.15 to 0.015 under L_∞ . In figure 3(b) a close up look on the curve is displayed; figure ?? displays how the combined (J_2 and the approximation error) progressed with the generations.

6 Discussion

In the current work two approaches for solving the optimal control equations using genetic algorithm have been demonstrated. The equations to be optimized suffer from highly non-linear nature due to both non linearity of knot vector modification action and from the use of, possibly, highly non-linear cost functions such as J_0 or J_3 .

In practice, both methods proved to be highly robust and handled well the non-linear problems. The robustness of the method is due to the easy handling the violation of the Schoenberg-Whitney condition. Moreover, creating the seed population from several heuristic methods ensures that the population will start with reasonable and usually valid curves. Finally, the use of genetic algorithm generated results which are substantially better than the best of the heuristic methods used to initialize the population.

The unified approach provided slightly better results at the price of higher computational effort. This approach also allows more flexibility in the choice of the operators and in setting the relationship between the knot vector and the parametrization. This flexibility can help tuning the algorithm further in order to achieve even better results.

The approach of using a cost function to design the shape of the curve is usually a good strategy, the usage of a genetic algorithm to solve the cost function proved as an efficient way to minimize the cost functions. The main limitation is that the appropriate cost function must be still selected. The results are much more reliable and better controlled than by using an heuristic approach, but still can be unexpected sometimes. Another issue is the performance, the current implementation is based on Matlab (as an interpreter). The optimization of the blade cross section shown in figure 8 took twenty minus on a Pentium IV 2.4 MHZ processor. The blade optimization had almost 200 d.o.f, and used a population of size 450. An efficient C implementation can be expected to perform much better, but still a genetic algorithm won't be suitable for interactive curve design programs in the near future.

7 Future Work

The next steps for expanding this algorithm will be to use NURBS splines. This extension is especially challenging because of the complexity of optimizing the NURBS' weights. Another promising direction would be to solve the optimal design problem for surfaces.

8 Acknowledgements

The authors would like to thank Carl de-Boor for his valuable remarks and suggestions and Marc Daniel and Eric Saux for the blade turbine cross section from figure 8.

REFERENCES

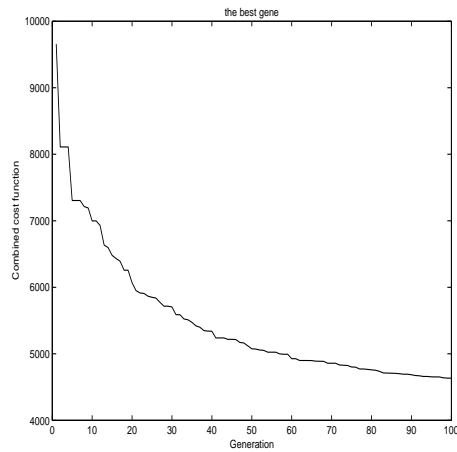
- [1] M. Alhanaty and M. Bercovier, "Curve and surface fitting and design by optimal control methods", *Computer-Aided Design*, Vol. 33, No. 2, pp. 167-182, February 2001
- [2] J. L. Breeden "Optimizing Stochastic and Multiple Fitness Functions", *Evolutionary Programming*, pp. 127-134, 1995.
- [3] G. Brunnett, Hans Hagen, Paolo Santarelli, "Variational design of curve and surfaces", *Surveys on Mathematics for Industry*, Vol 3. pp. 1-27, 1993.
- [4] C. De Boor, "Package for calculating B-splines", *Siam Journal Numerical Analysis*, Vol 14. No. 3 pp. 441-472, June 1977.
- [5] C. de Boor, "Computational aspects of optimal recovery", *Optimal Estimation in Approximation Theory*, C.A. Micchelli and T.J. Rivlin eds., Plenum Publ., New York, pp. 69-91, 1977
- [6] C. de Boor, "A Practical Guide To Splines", *New York: Springer-Verlag*. 1978.
- [7] K. A. De jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems", Ph.d thesis, University of Michigan, Ann Arbor.
- [8] G. Farin, N.Sapidis "Automatic Faring algorithm for B-spline curves", *Computer Aided Design*, Vol. 22 No. 2, pp. 121-129, 1990
- [9] G. Farin, "Curves and Surfaces for CAGD: A Practical Guide", Academic Press, 1998.
- [10] P. Laurent-Gengoux, and M. Mekhilef, "Optimization of a NURBS representation", *Computer Aided Design*, Vol. 25(11), pp. 699-710, 1993.
- [11] D. E. Goldberg, "Genetic Algorithms in Search Optimization and Machine Learning", Addison-Wesley, 1989
- [12] D. E. Goldberg, K. Deb, "A comparative analysis of selection schemes used in genetic algorithms", *In G. Rawlins, ed. Foundations of Genetic Algorithms*, Morgan Kaufmann, 1991.
- [13] R. Goldenthal, M. Bercovier "Spline Curve approximation and Design by Optimal Control Over the Knots", *Technical Report 2003-7*, Leibniz Center, 2003.
- [14] G. H. Golub, C. F. Van Loan, "Matrix Computations", Third Edition, Ed. Johns Hopkins, 1996.
- [15] I. Juhász, M. Hoffmann "The Effect of Knot Modifications on the Shape of B-spline Curves", *Journal for Geometry and Graphics*, Vol. 5, No. 2, pp. 111-119, 2001.
- [16] J. Hoschek, "Intrinsic parameterization for approximation", *Computer Aided Geometric Design*, Vol. 5, pp. 27-31, 1988.
- [17] A. Markus, G. Renner, J. Vancza. "Genetic algorithms in free form curve design", *Mathematical Methods for Curves and Surfaces*, Vanderbilt University Press, pp 343-354. 1995.
- [18] A. Markus, G. Renner, J. Vncza, "Spline Interpolation with Genetic Algorithms", *Proceedings of the 1997 International Conference on Shape Modelling and Applications*, Aizu-Wakamatsu, pp. 47-54. 1997

- [19] M. Mitchell “An Introduction to Genetic Algorithms”, MIT Press, 1996.
- [20] L. Piegl, “Modifying the shape of rational B-splines. part 1:curves”, *Computer-Aided Design*, Vol. 21 Issue 8, pp. 509-518, 1989.
- [21] L. Piegl, W. Tiller “The NURBS book, 2nd Edition”, *New York: Springer-Verlag*. 1997.
- [22] I. J. Schoenberg, A. Whitney, “On Pólya frequency functions III”, *Trans. Amer. Math. Soc*, Vol. 74. pp. 246-259, pp. 246-259.
- [23] T. Speer, M. Kuppe, and J. Hoschek, “Global reparametrization for curve approximation”, *Computer Aided Geometric Design*, Vol. 15, pp. 869-877, 1998.
- [24] D. J. Turcke, G. M. McNeice, “Guidelines for selecting finite element grids based on an optimization study”, *Computers and Structures*, Volume 4, Issue 3, May 1974, pp. 499-519



(a) The turbine blade cross section

(b) A close up on the cross section, both the approximation error and the curvature have been improved, the gray curve is the most fit curve among the seed population



(c) The resulting knot vectors. Average and optimal methods used the centripetal parametrization as input.

Figure 3: Turbine blade cross section, the gray curve is the most fit curve among the seed population, the black curve is the most fit individual of the last generation