

Particle Swarm Algorithm for the Shortest Cubic Spline Interpolation

Shang Gao¹, Zaiyue Zhang¹, and Cungen Cao²

¹ School of Computer Science and Engineering, Jiangsu University of Science and Technology, Zhenjiang, China
 Email: gao_shang@hotmail.com yzzjzzy@sina.com

² Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
 Email: cgcao@ict.ac.cn

Abstract—The spline technology has applications in CAD, CAM, and computer graphics systems. Based on analysis of cubic spline interpolation, the problem of the shortest cubic spline interpolation is discussed in this paper. Furthermore, the particle swarm algorithm for this problem is presented. At last, an example is given. The result of numerical examples shows the efficacy of the proposed model and algorithm.

Index Terms—cubic spline interpolation, particle swarm algorithm, CAD

I. INTRODUCTION

Splines and particularly cubic splines are very popular models for interpolation. Historically, a "spline" was a common drafting tool, a flexible rod, that was used to help draw smooth curves connecting widely spaced points. The cubic spline curve accomplishes the same result for an interpolation problem. The spline technology has applications in CAD, CAM, and computer graphics systems. We describe cubic splines in this note and discuss their use in interpolation and curve fitting. The cubic spline interpolation is a piecewise continuous curve, passing through each of the values in the table. There is a separate cubic polynomial for each interval, each with its own coefficients. The first derivative and the second derivative of a cubic spline are continuous. The shortest cubic spline interpolation problem is discussed in this paper. Furthermore, the particle swarm algorithm for this problem is presented.

II. CUBIC SPLINE INTERPOLATION

In mathematics, a spline is a special function defined piecewise by polynomials. In interpolating problems, spline interpolation is often preferred to polynomial interpolation because it yields similar results, even when using low degree polynomials, while avoiding Runge's phenomenon for higher degrees.

In the computer science subfields of computer-aided design and computer graphics, the term spline more frequently refers to a piecewise polynomial (parametric) curve. Splines are popular curves in these subfields because of the simplicity of their construction, their ease and accuracy of evaluation, and their capacity to approximate complex shapes through curve fitting and

interactive curve design.

Polynomial parametric curves of high degree have a disadvantage: requirements placed on one stretch of such a curve can have a very strong effect some distance away. In figure 1, the jump in the height of the data points near the middle has a strong effect on the interpolating polynomial curve near the ends.

In contrast, figure 2 show an example of a cubic spline curve through the same data points. Notice how it follows them much more closely.

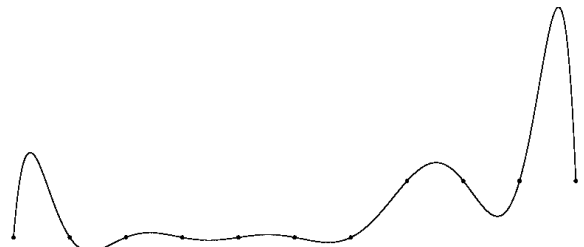


Figure 1 Lagrange interpolation of data points



Figure 2 Spline interpolation of the same data points

The spline curve was constructed by using a different cubic polynomial curve between each two data points. In other words, it is a piecewise cubic curve, made of pieces of different cubic curves glued together. The pieces are so well matched where they are glued that the gluing is not obvious.

Suppose that $\{(x_i, y_i)\}_{i=0}^N$ are $N+1$ points, where $a = x_0, x_1, \dots, x_n = b$. The function $S(x)$ is called a cubic spline if there exist N cubic polynomials and satisfy the properties:

I. $S(x)$ is a cubic polynomial on $[x_{i-1}, x_i]$ ($i = 1, 2, \dots, n$)

II. $S(x_i^-) = S(x_i^+)$ ($i = 1, 2, \dots, n-1$)

III. $S'(x_i^-) = S'(x_i^+)$ ($i = 1, 2, \dots, n-1$)

IV. $S''(x_i^-) = S''(x_i^+)$ ($i = 1, 2, \dots, n-1$)

V. $S(x_i) = y_i$ ($i = 0, 1, \dots, n$).

Supported by National Basic Research Program of Jiangsu Province University (08KJB520003), the National Natural Science Foundation of China(No.60773059).

where c_0 denotes the inertia weight factor; p_{id} is the location of the particle that experiences the best fitness value; P_{gd} is the location of the particles that experience a global best fitness value; c_1 and c_2 are constants and are known as acceleration coefficients; d denotes the dimension of the problem space; $rand_1$, $rand_2$ are random values in the range of (0, 1). For equation (1), the first part represents the inertia of previous velocity; the second part is the ‘‘cognition’’ part, which represents the private thinking by itself; the third part is the ‘‘social’’ part, which represents the cooperation among the particles. If the sum of accelerations would cause the velocity v_{id} , on that dimension to exceed $v_{max,d}$, then v_{id} is limited to $v_{max,d}$. $v_{max,d}$ determines the resolution with which regions between the present position and the target position are searched.

The PSO algorithm for the shortest Bezier curve can be described as follows:

I) Initialize particle

- a) Set constants c_0 , c_1 and c_2 .
- b) Randomly initialize particle positions.
- c) Randomly initialize particle velocities.

II) Do:

a) For each particle:

1) Calculate fitness value (the arc length of the curve L).

2) If the fitness value is better than the best fitness value P_{id} in history.

3) Set current value as the new P_{id} .
End

b) For each particle:

1) Find in the particle neighborhood, the particle with the best fitness P_{gd} .

2) Calculate particle velocity according to the velocity equation (6).

3) Apply the velocity constriction.

4) Update particle position according to the position equation (7).

5) Apply the position constriction.

End

While maximum iterations or minimum error criteria is not attained.

IV. EXAMPLES

Find the shortest cubic spline $S(x)$ for following points (Table I) .

TABLE I.
FIVE SPLINE POINTS

x_i	0	1	2	3	4
y_i	1	0.5	2	3	1.5

In our implementation, the following parameters are set as default values: $n_p = 50$, $n_{max} = 50$, and $v_{max} = 0.5$. We can find the solution $m_0 = -0.2299$ and $m_4 = -2.1155$, and also we can get the shortest

length of the curve $L^* = 6.3198$.Using equation (2), the shortest cubic spline is

$$S(x) = \begin{cases} 1.1105x^3 - 1.3806x^2 - 0.2299x + 1 & \text{if } 0 \leq x < 1 \\ -0.7914x^3 + 4.3251x^2 - 5.9365x + 2.9019 & \text{if } 1 \leq x < 2 \\ -0.4450x^3 + 2.2467x^2 - 1.7788x - 1.7364 & \text{if } 2 \leq x < 3 \\ 0.5713x^3 - 6.9001x^2 + 25.6616x - 27.3098 & \text{if } 3 \leq x \leq 4 \end{cases} \quad (8)$$

The shortest cubic spline is shown in Figure 3.

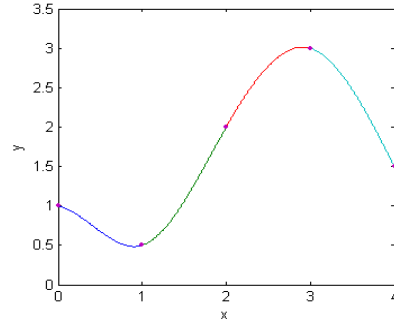


Figure 3. The shortest cubic spline

V. CONCLUSIONS

Cubic splines are popular because they are easy to implement and produce a curve that appears to be seamless. It is significant to solve the shortest cubic spline problem in saving material and saving energy. This paper paid little attention on the chasing method and composite Simpson rule. We can acquire the detail computation methods through references. In practical applications, we can use Matlab toolbox directly to accomplish chasing method and composite Simpson rule. In this paper, the shortest cubic spline problem is researched on the clamped spline, and the curvature-adjusted cubic spline and periodic spline can be similarly processed.

ACKNOWLEDGMENT

This work was partially supported by National Basic Research Program of Jiangsu Province University (08KJB520003) , the National Natural Science Foundation of China(No.60773059) and the Open Project Program of the State Key Lab of CAD&CG.

REFERENCES

- [1] R.L. Burden and J.D. Faires, Numerical Analysis, Higher Education Press & Thomson Learning, Inc., 2001, pp.141-150, 408-409.
- [2] J.H. Mathews and K.D. Fink, Numerical Methods Using MATLAB, Publishing House of Electronics Industry, 2002, pp.280-290.
- [3] M. I. Syam, ‘‘Cubic spline interpolation predictors over implicitly defined curves’’, Journal of Computational and Applied Mathematics, 2003, 157(2), pp.283-295.
- [4] R. C. Eberhart and J. Kennedy, ‘‘A New Optimizer Using Particles Swarm Theory’’, Proc. 6th International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995:39-43.
- [5] Y. H. Shi and R. C. Eberhart, ‘‘A Modified Particle Swarm Optimizer’’, IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9, 1998:69-73.