DESIGN AND DEVELOPMENT OF INTERPOLATION & APPROXIMATION CURVE CONVERSION BASED ON MONOMIAL FORM

MR. DILOKVITH SAVETSERANEE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF MASTER OF ENGINEERING (COMPUTER ENGINEERING)
FACULTY OF ENGINEERING
KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI
2014

Design and Development of Interpolation & Approximation Curve Conversion Based on Monomial Form

Mr.Dilokvith Savetseranee B.Eng. (Computer Engineering)

A Thesis Submited in Partial Fulfillment of the Requirements for
the Degree of Master of Engineering (Computer Engineering)
Faculty of Engineering
King Mongkut's University of Technology Thonburi
2014

Thesis Committee

……. Chairman of Thesis Committee
(Assoc. Prof. Thumrongrat Amornraksa, Ph.D.)

…… Member and Thesis Advisor
(Assoc. Prof. Natasha Dejdumrong, D.Tech.Sci.)

……. Member
(Jumpol Polvichai, Ph.D.)

……. Member
(Asst. Prof. Pornchai Mongkolnam, Ph.D.)

……. Member
(Asst. Prof. Pizzanu Kanongchaiyos, Ph.D.)

Thesis Title	Design and Development of Interpolation & Approximation Curve Conversion Based on Monomial Form
Thesis Credits	12 credits
Candidate	Mr.Dilokvith Savetseranee
Thesis Advisor	Assoc. Prof. Dr. Natasha Dejdumrong
Program	Master of Engineering
Field of Study	Computer Engineering
Department	Computer Engineering
Faculty	Engineering
Academic Year	2014

Abstract

In Computer Aided Geometric Design (CAGD), complex shapes are characterized by a number of control points and basis function. The utilizations of using curves are represented in the architectural works. Curve can be categorized based on their construction and characteristics into two major groups, namely, approximation curve and interpolation curve. Generally, the interpolation curve is usually found in the cartooning and animation field where the raster image is converted into the interpolated curve. On one hand, the approximation curve is widely used for geometric modeling in CAD and CAM applications. Owing to ease of configuration for the modeling of the interpolation curve and high degree of shape preservation of the approximation curve, the integration of their prominent features is promising to deliver the improvement of the modeling and design for CAGD application and image for CNC machine. In this research, we designed and developed the curve conversion scheme based on the monomial matrix to convert the interpolation curve into the approximation curve and vice versa. Moreover, optimization of degree for output curve is evaluated to reduce the complexity. Finally, the experiments show that our proposed scheme is feasible to implement and outperform the previous research works based on the pseudo inverse model in terms of the functional integration and applicability with CAGD application package.

Keywords : Monomial Form, Monomial Matrix, Relationship of Interpolation and Approximation curve

หัวข้อวิทยานิพนธ์	การออกแบบและพัฒนาเส้นโค้งประมาณค่า และ เส้นโค้งจากการประมาณค่าในช่วง เพื่อการแปลงฐานโมเดล ด้วย เมตริกซ์เอกนาม
หน่วยกิต	12
ผู้เขียน	นายดิลกวิทย์ เศวตเศรนี
อาจารย์ที่ปรึกษา	รศ. ดร.ณัฐชา เดชดำรง
หลักสูตร	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
คณะ	วิศวกรรมศาสตร์
ปีการศึกษา	2557

บทคัดย่อ

ในสายงานคอมพิวเตอร์เพื่อการออกแบบ ได้นำเส้นโค้งมาใช้ในอุตสาหกรรมการออกแบบและสถาปัตยกรรม ด้วยคุณลักษณะและโครงสร้างในการขึ้นรูปสามารถแบ่งเส้นโค้งออกเป็นสองประเภท ได้แก่ เส้นโค้งประมาณค่าและเส้นโค้งจากการประมาณค่าในช่วง สำหรับเส้นโค้งจากการประมาณค่าในช่วง นิยมใช้ในการออกแบบ การ์ตูน อุตสาหกรรมการสร้างภาพเคลื่อนไหวที่มีการแปลงรูปภาพประเภทแรสเตอร์ ให้อยู่ในโครงสร้างที่มีเส้นโค้งจากการประมาณค่าในช่วง ในทางกลับกันเส้นโค้งประมาณค่า ได้นำมาใช้อย่างแพร่หลายในอุตสาหกรรมงานออกแบบรวมทั้งการออกแบบเชิงอุตสาหกรรม เนื่องจากการขึ้นรูปด้วยเส้นโค้งจากการประมาณค่าในช่วงที่มีดีกรีของตัวควบคุมลักษณะ มีค่าที่สูง สามารถลดดีกรีลงเมื่อถูกเปลี่ยนให้อยู่ในรูปเส้นโค้งประมาณค่าได้ เพื่อนำไปพัฒนา การสร้างโมเดล งานออกแบบสำหรับโปรแกรมประยุกต์ในกลุ่มงานคอมพิวเตอร์เพื่อการออกแบบและ รูปที่ใช้สำหรับเครื่องขึ้นรูปชิ้นงานทางอุตสาหกรรมซีเอ็นซี ในงานวิจัยนี้จะทำการศึกษาและพัฒนาความสัมพันธ์ของเส้นโค้งประมาณค่าและเส้นโค้งจากการประมาณค่าในช่วง เพื่อให้สามารถแปลงฐานของเส้นโค้งระหว่างเส้นโค้งสองกลุ่มข้างต้น ด้วยวิธีการใช้ เมทริกซ์เอกนามเป็นวิธีรการแปลงโมเดลของเส้นโค้งได้ทั้งสองทิศทาง สำหรับงานวิจัยชิ้นนี้เป็นการปรับปรุง พัฒนาขอบเขตและประสิทธิภาพจากงานวิจัยเดิม ที่ใช้วิธีการผกผันในการสร้างโมเดลประมาณเส้นโค้ง ให้สามารถนำไปใช้ได้จริงกับโปรแกรมประยุกต์สำหรับการออกแบบ

คำสำคัญ : เมตริกซ์เอกนาม /การแปลงโมเดล/ ความสัมพันธ์ระหว่างเส้นโค้งประมาณค่า และ เส้นโค้งจากการประมาณค่าในช่วง

ACKNOWLEDGEMENTS

I couldn’t have completed this project without the help of my advisor, Assoc. Prof. Natasha Dejdumrong, D.Tech.Sci. My professor dedicated her time to guide me and to make sure this research work ran smoothly. Moreover, I was fully supported by members in laboratory of CAGD and Animation (CADLAB), the department of computer engineering, KMUTT.

Finally, I would like to express my gratitude to the committee for their valuable suggestions in the thesis examination.

CONTENTS

CONTENTS (Cont.)

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Statement of the Problem

Geometric modeling is one of the fundamental research issues in computer graphics. Even though substantial works have proposed the solutions for graphical design and geometric modeling, efficient and advanced modeling techniques for geometry representation are still crucially required for modern applications. In this kind of research area, most solutions focus on devising mathematical algorithms to represent the objects and shapes.

Regarding the shape-based modeling, a primary shape is simply modeled by a construction formula and fixed control points for a common object's design such as lines, triangular, square circle, ellipse and etc. On the other hand, a complex shape or free-form object is much more difficult to invoke a simple formula and typical fixed control points to model. Thus, the complex shape requires advanced and detailed design of construction formula to generate control points. Consequently, more research has devoted to solve the complex shape representation.

In Computer Aided Geometric Design (CAGD), complex shapes are defined in the form of curves. The utilizations of curves are represented in the architectural works. For example, curves can help modelers draft a modern car structure. They provide simple tools for modelers in some parts of drawing and sketching. Curves are important to geometric modeling used to sketch a part of objects and represent structural shapes.

Curve can be separated based on their construction and characteristics into two major groups, namely, approximation curve and interpolation curve. Approximation curves are composed of control points around the curve line. These control points are used to set the direction and to generate their blending function to construct a curve. The examples of these models are Bézier Curve, Wang-Ball Curve, Said Ball Curve and etc. This kind of curve has a good shape preserving properties and is popular in the CAGD field. For the interpolation curve, it is a curve whose line must pass along all control points. In many applications, interpolation curves are typically used to sketch and design process because its control points that is on the curve is easy to configure the shape.

Besides, the interpolation curve is attractive in its utilities and complex applications. However, the complex blending functions and complicated methods require much more computation time and resource consumption. Thus, both curve forms have their own strengths and weaknesses.

In this research, we aim to improve geometric modeling applicable in CAGD applications. Due to a high degree of the availability of detected points along the shape line of the interpolation curve scheme, we thus integrate this scheme to provide more efficiency to the approximation curve which is a foundation approach for CAGD application. Here, the relationship between the approximation curve and the interpolation curve is examined.

In addition, the mathematic monomial form is employed to construct the conversion model to assemble the relation between the approximation curve and interpolation

curve. This could enable the migration of geometric modeling to CAGD done with the efficient transition and less computation cost.

1.2 Research Objectives

The objectives of this research are to:

1.Propose an extended monomial form method in curve applications as a mathematical relationship representation of the approximation curve and interpolation curve.

2.Apply monomial matrices as a conversion application that is proven by Mathematica Software.

1.3 Scope and Limitation of the Study

1. Investigate and specify the relationship between approximation curve and interpolation curve.

2. Develop monomial matrices to demonstrate the relations among each of the curve models.

3. Evaluate the efficiency and performance of the proposed monomial conversion model and show that it is well applicable in Mathematica programming.

1.4 Overview of Thesis

This document is divided into five chapters: Chapter 1 presents a problem statement, objectives of the research and desired output. Chapter 2 deals with the related backgrounds, theories and previous works. Chapter 3 explains the concept of conversion algorithm and methods which are used to apply in approximating the interpolated control points. Chapter 4 demonstrates experimental results used to describe beneficial purposes as a conversion application. Finally, Chapter 5 summarizes research and suggestions for further development.

CHAPTER 2 LITERATURE REVIEW

This chapter reviews the background of fundamental concepts of geometric modeling, curve modeling, curve properties, using curve with low complexity, and Chebyshev polynomial. Then, the previous research is discussed.

2.1 Geometric Modeling

Geometric modeling is generally considered as mutual fields of applied mathematics and computational geometry that provide the methods, algorithms, or formula in the form of mathematical expression of shapes. The shape studied is mostly represented in two or three dimensions. Two dimensional models are useful in computer typography and technical drawing. Three dimensional models are widely used in many fields such as mechanical engineering, image processing, and architectural design. However, the ground concept of geometric modeling is curves. In Computer Aided Geometric Design (CAGD), shapes are specifically defined in terms of curves and surfaces. The property of shapes is also regarded in CAGD. Examples of curves and surfaces are Bézier, Said-Ball, Wang-Ball curves, and surfaces. In our research work, we will concentrate on CAGD curves.

2.2 Curve Modeling

Curves in Computer Aided Design can be classified into two main categories: approximation curve or CAGD curve and interpolation curves. For the approximation curve, it is the composed of control points out of the curve line. Thus, it can be called approximation control point curve. These control points are used to set the direction and shape construction of curves. For the interpolation curve, it is a curve where the line must pass along all control points called interpolation control point curve.

In many applications, interpolation curves are typically used to sketch and design a process. Cartoon animation, fine art design, and industrial product design. On another hand, approximation curves are popular in the CAGD curve design such as machine design and civil structure design. Curve designs are constructed by two principle parts: a set of control points and blending function. A set of control points specifies the position and direction of curve while a blending function is used to control the shape of curves in terms of polynomials equation. In our work, we will use the term "polynomial curves" to classify a curve model.

2.2.1 Polynomial Curves

There are several polynomial curves in CAGD, such as Bézier, Said-Ball and Wang-Ball curves. These polynomial curves can be constructed by using their control points and blending functions. Both components influence the shape of polynomial curves. In each curve, their characteristics are similar but they are different in their blending functions.

2.2.2 Approximation Control Point Curve (CAGD Curves)

Approximation curve is a curve where all control points do not need to locate on the created curve. This kind of curve is widely adopted because it has a high degree of shape preservation as well as its ease of use. This curve can also be constructed by using its control points and its blending functions. Approximation curve is very useful in computer aided design work. There are several models to use in each appropriated application; mechanical project, architecture software and 3D printing. The most popular kind is Bézier Curve. Figure 2.1 presents an approximated control point curve.

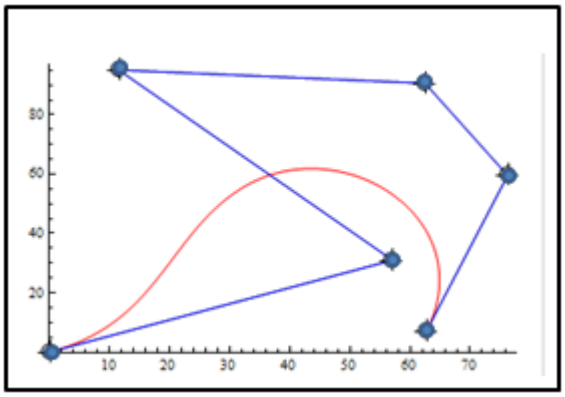


Figure 2.1 Approximation Curve: Character and Position of Control Point

In order to understand each model of approximation curve, the history of each model is reviewed as follows.

2.2.2.1 Bézier Curves

Bézier curve can be constructed by using either de Casteljau algorithm in 1957 or a model based on Bernstein polynomials proposed by Bézier [1]. The curve is presented by two components: Bernstein polynomials and Control Point. Bézier is popular because there is clear Bernstein polynomial form and the influence of control point is equal at every point. However, this model is not practical if there is a high degree of curve. Therefore, a subsequent solution called Said-Ball algorithm was developed to address the problem of high degree curve modeling.

2.2.2.2 Said-Ball Curves

In 1974, Ball defined a set of basis function for cubic curve and Bicubic surfaces. Said [2] investigated Said-Ball curves by generalizing the Ball model to support a higher degree of curves. Said-Ball curve is similar to Bézier curve but they are different in their blending function. Afterwards, Wang et. Al. [3] developed the recursive algorithm for Said-Ball curves. It creates the simplicity for Said-Ball curve construction. Hence, Said was the best algorithm to draw a curve during that time. Unfortunately, Said-Ball is more complex to use compared to Wang-Ball algorithm.

2.2.2.3 Wang-Ball Curves

Wang-Ball curve was investigated by Wang [4] in 1987 but publicized later in 1996 by Wang et.al [3] with its degree elevation and degree reduction. A Wang-Ball curve can be computed more efficiently than Bézier and Said-Ball curve because it requires linear time complexity *(n)*.

Subsequently, in 2001 Dejdumrong et al. [5] extended to the rational Wang-Ball curve. The problem of Wang-Ball curve is about its low influence of control points in curve control. The idea is that changing the shape of curve can be made through the change of control points.

2.2.2.4 DP Curves

In 2003, Delgado and Peña [6] developed DP curves to represent a curve having linear computational complexity. Researchers proved that blending basis of DP curves is a normalized positive basis which claims that DP curves have a shape preserving property.

2.2.2.5 NB1 Curves

NB1 and NB2 were first investigated by Wu [7] in 2000. Later, in 2007, Yu and Chen [8] introduced the basis functions and their properties of NB1 curves (NB2 curve in Wu). NB1 curve construction can be divided into two parts. In the first half, NB1 curve is computed by using NB1 algorithm. In the second half, it is computed similar to Bézier algorithm. NB1 curve is located between Said-Ball and Wang-Ball curves.

2.2.2.6 Dejdumrong Curve

In 2008, a new algorithm of plotting curves was presented by Dejdumrong [5]. The curve recursive algorithm is a linear computation form similar to Wang-Ball and DP curves. The proposed algorithm can compute both non-rational and rational Bézier curves within a linear time complexity. The algorithm of Dejdumrong was proven to be more efficient than the de Casteljau algorithm because it has a linear complexity which can be computed easily. Dejdumrong curve calculation was faster than the other curves except Wang-Ball and DP curves. However, the shape of Dejdumrong curve was closest to that of Bézier curve. Furthermore, Dejdumrong curve can be plotted faster than Bézier curve.

2.2.3 Interpolation Curve

Interpolation curve is a kind of curve which has a set of control points located on the generated curve. This curve is used in the mathematics field of numerical analysis to give the interpolation equation for a given set of data. In CAGD, it models a type of curve segment. In many applications, the interpolation curve is typically used to compare with two sets of data which are used in statistical comparisons. For example, the information from financial investment can be obtained from stock information. The information from statistical works can be used for the country's development in many aspects such as educational development, economic development, social development, agricultural development, industrial development, and public health development. The results from statistical analysis are important for human and society. Figure2.2 presents a sample of interpolation curve.

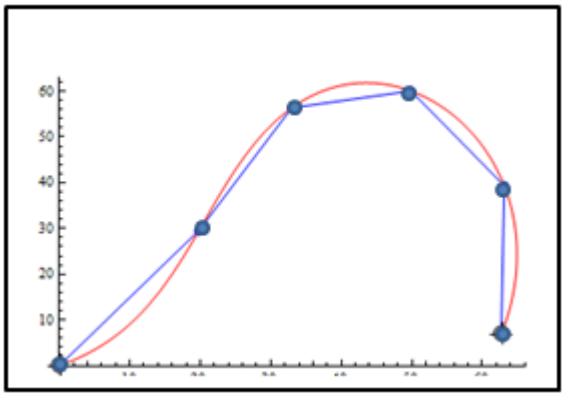


Figure 2.2 Interpolation Curve: Relationship between Shape and Control Points

2.2.3.1 Newton-Lagrange Curves

Newton and Lagrange polynomial is typically used to represent an interpolation curve known as "Newton-Lagrange curve". In Nuntawisuttiwong and Dejdumrong [9] apply Newton-Lagrange polynomial to approximate Hand writing and it is converted to Bézier curve. However, this technique can be applied for simple curves only because Newton-Lagrange is sensitive for ill-conditions and not flexible for complex curves such as loop curve and zigzag curve. It is not flexible for all input cases in order to use as a conversion method.

Let $N^n(t)\ and\ L^n\ (t)$be Newton and Lagrange curve with n +1 control points, $\{\boldsymbol{C_i}\}_{i=0}^{n}$, and a set of knots, $\{t_k\}_{k=0}^{n}$, then Newton-Lagrange polynomial can be represented by

$$N^n(t) = L^n(t) = \sum_{i=0}^{n}\sum_{j=0}^{n} c \cdot f_{i,j} \cdot t^i \quad , \tag{1}$$

Where $f_{i,j}$ is Newton-Lagrange blending function defined by

$$f_{i,j} = \frac{F_{i,n-j}^{n}(0)}{\prod_{i=0,i\neq j}^{n} t_j - t_i}$$

And

$$F_{i,n-j}^{n}(u) = \begin{cases} 1 & , \quad j = 0 \\ \sum_{k=u,k\neq i}^{n-j+1} F_{i,j-1}^{n}(k+1)t_k & , \quad j > 0 \end{cases} \tag{2}$$

2.3 Methods for Curve Representation

After all the prominent curves models were presented, this section describes the algorithms for curve construction in each model.

2.3.1 Basis Function

Typically, curves can be constructed by using their control points and blending functions. In this method, the blending functions are defined in terms of basis functions. A basis function is a simple method to create CAGD curves. The implementation and application in programming are the main issues discussed in this part.

2.3.1.1 Bézier Curve

To create a Bézier curve by its basis, there are two simple components: control points and their blending function. The specific blending function to generate Bézier curve is Bernstein polynomial. The formula is shown below.

A Bézier curve, $B^n(t)$ of degree n with control points, denoted by $\{b\}_{i=o}^{n}$ and can be constructed by [1]

$$B^n(t) = \sum_{i=0}^{n} B_i^n(t) \cdot \boldsymbol{b}_i, \quad 0 \le t \le 1 \tag{3}$$

where $\boldsymbol{b}_i$ is control points , $i = 0,1,2, \dots, n$
$B_i^n(t)$ is the n^{th} degree Bernstein polynomial
expressed by

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \; ; 0 \le t \le 1 \tag{4}$$

and

$$\binom{n}{i} = \frac{n!}{i!\,(n-i)!} \tag{5}$$

2.3.1.2 Said-Ball Curve

The second curve is Said-Ball curve. The blending function of Said curve is improved from ball polynomial. The creation of this cure via programming can be separated into 3 parts. These three conditions belong to the number of control points.

Said-Ball curve, is represented by $S^n(t)$ with the set of control points, $\{V_i\}_{i=o}^{n}$ that can be expressed by [3]

$$S^n(t) = \sum_{i=0}^{n} \boldsymbol{V_i} \cdot S_i^n(t), \quad 0 \le t \le 1 \tag{6}$$

where $\boldsymbol{V}_i$ is control points, $i = 0,1,2, \dots, n$
$S_i^n(t)$ is the n^{th} degree Said polynomial

$$S_i^n(t) = \begin{cases} \binom{\lfloor \frac{n}{2} \rfloor + i}{i} t^i (1-t)^{\lfloor \frac{n}{2} \rfloor + 1} & , \; for \; 0 \le i \le \lceil \frac{n}{2} \rceil, \\ \binom{n}{i} t^i (1-t)^i & , \; if \; i = \frac{n}{2}, \\ S_{n-i}^n (1-t) & , \; for \; \lfloor \frac{n}{2} \rfloor + 1 \le i \le n. \end{cases} \tag{7}$$

2.3.1.3 Wang-Ball Curve

The fundamental of Wang curve is also developed from Ball polynomial. This blending function can be separated into 4 conditions of programing.

Wang-Ball curve, is represented by $A^n(t)$ of degree n with the set of control points, $\{\boldsymbol{p}_i\}_{i=o}^{n}$ that can be expressed by [3]

$$A^n(t) = \sum_{i=0}^{n} \boldsymbol{p_i} \cdot A_i^n(t), \quad 0 \le t \le 1 \tag{8}$$

where $\boldsymbol{p_i}$ is control points, $i = 0,1,2,\dots,n$
$A_i^n(t)$ is the n^{th} degree Wang polynomial

$$A_i^n(t) = \begin{cases} (2t)^i(1-t)^{i+2} & , \ for\ 0 \le i \le \left\lfloor \frac{n}{2} \right\rfloor - 1, \\ (2t)^i(1-t)^{n-i} & , \ if\ i = \left\lfloor \frac{n}{2} \right\rfloor, \\ (2(1-t))^{n-i}t^i & , \ if\ i = \left\lceil \frac{n}{2} \right\rceil, \\ A_{n-i}^n(1-t) & , \ for \left\lceil \frac{n}{2} \right\rceil + 1 \le i \le n. \end{cases} \tag{9}$$

2.3.1.4 DP Curve

DP curve creation by using its basis function is very complex and hard to apply as programming. There are several conditions to coding and implementation.
DP curve is represented by $C^n(t)$ of degree n with the set of control points, $\{\boldsymbol{q}_i\}_{i=o}^{n}$ that can be expressed by [6]

$$C^n(t) = \sum_{i=0}^{n} \boldsymbol{q_i} \cdot C_i^n(t), \quad 0 \le t \le 1 \tag{10}$$

where $\boldsymbol{q_i}$ is control points, $i = 0,1,2,\dots,n$
$C_i^n(t)$ is the n^{th} degree DP polynomial

$$C_i^n(t) = \begin{cases} (1-t)^n & , \quad if\ i = 0, \\ t(1-t)^{n-i} & , \quad for\ 1 \le i \le \left\lfloor \frac{n}{2} \right\rfloor - 1, \\ \left(\left\lceil \frac{n}{2} \right\rceil - \left\lfloor \frac{n}{2} \right\rfloor\right) t(1-t)^{\left\lfloor \frac{n}{2} \right\rfloor + 1} & \\ + \left(\frac{1}{2}\right)^{\left\lceil \frac{n}{2} \right\rceil - \left\lfloor \frac{n}{2} \right\rfloor} \left(1 - t^{\left\lfloor \frac{n}{2} \right\rfloor + 1} - (1-t)^{\left\lfloor \frac{n}{2} \right\rfloor + 1}\right) & , \quad for\ i = \left\lfloor \frac{n}{2} \right\rfloor, \\ \left(\left\lceil \frac{n}{2} \right\rceil - \left\lfloor \frac{n}{2} \right\rfloor\right) (1-t)t^{\left\lfloor \frac{n}{2} \right\rfloor + 1} & \\ + \left(\frac{1}{2}\right)^{\left\lceil \frac{n}{2} \right\rceil - \left\lfloor \frac{n}{2} \right\rfloor} \left(1 - t^{\left\lfloor \frac{n}{2} \right\rfloor + 1} - (1-t)^{\left\lfloor \frac{n}{2} \right\rfloor + 1}\right) & , \quad for\ i = \left\lceil \frac{n}{2} \right\rceil, \\ t^i(1-t) & , \quad for\ \left\lceil \frac{n}{2} \right\rceil + 1 \le i \le n - 1, \\ t^n & , \quad for\ i = n. \end{cases} \tag{11}$$

2.3.1.5 NB1 Curve

The basis form of NB1 is separated into 4 conditions for the implementation. This basis is a simple form used to create NB1 curves but its blending function is complex to be represented in the programming language shown below.

A NB1 Curve, $N^n(t)$ of degree n with control points, $\{\boldsymbol{y}_i\}_{i=o}^{n}$ can be computed by[7][8]

$$N^n(t) = \sum_{i=0}^{n} \boldsymbol{y_i} \cdot N_i^n(t), \quad 0 \le t \le 1 \tag{12}$$

where $\boldsymbol{y_i}$ is control points, $i = 0,1,2,\dots,n$
$N_i^n(t)$ is the n^{th} degree NB1 polynomial

$$N_i^n(t) = \begin{cases} \binom{\lfloor \frac{n}{2} \rfloor - 1 + i}{i} t^i (1-t)^{\lfloor \frac{n}{2} \rfloor} & , \ for\ 0 \le i \le \left\lfloor \frac{n}{2} \right\rfloor - 2, \\ \binom{2\lfloor \frac{n}{2} \rfloor - 2}{\lfloor \frac{n}{2} \rfloor - 1} t^{\lfloor \frac{n}{2} \rfloor - 1} (1-t)^{\lfloor \frac{n}{2} \rfloor + 1} & , \ if\ i = \left\lfloor \frac{n}{2} \right\rfloor - 1, \\ 2\binom{2\lfloor \frac{n}{2} \rfloor - 2}{\lfloor \frac{n}{2} \rfloor - 1} t^{\lfloor \frac{n}{2} \rfloor} (1-t)^{\lceil \frac{n}{2} \rceil} & , \ if\ i = \left\lfloor \frac{n}{2} \right\rfloor, \\ N_{n-i}^n(1-t) & , \ for\ \left\lfloor \frac{n}{2} \right\rfloor + 1 \le i \le n. \end{cases} \tag{13}$$

2.3.1.6 Dejdumrong Curve

Dejdumrong is the last CAGD curve discussed in this thesis. Dejdumrong blending function is the improvement from Wang-Ball model. Thus, this blending function looks like Wang-Ball blending function. The conditions are also separated into 4 cases. Dejdumrong Curve presents degree n with control point n +1 points and the equation of basis as follows:

$$D^n(t) = \sum_{i=0}^{n} \boldsymbol{d_i} \cdot D_i^n(t) \quad ; 0 \le t \le 1 \tag{14}$$

where $\boldsymbol{d_i}$ is control points, $i = 0,1,2,\dots,n$
$D_i^n(t)$ is the n^{th} degree Dejdumrong polynomial

$$D_i^n(t) = \begin{cases} (3t)^i * (1-t)^{i+3} & for\ 0 \le i < \left\lceil \frac{n}{2} \right\rceil - 1 \\ (3t)^i * (1-t)^{n-i} & for\ i = \left\lceil \frac{n}{2} \right\rceil - 1 \\ 2 * 3^i (1-t)^i * t^i & for\ i\ even\ and\ i = \frac{n}{2} \\ D_{n-i}^n(1-t) & for\ \left\lfloor \frac{n}{2} \right\rfloor + 1 < i \le n \end{cases} \tag{15}$$

All basis functions for curve construction are difficult to implement in a computer software because their definition is represented by recursive functions such as Said-Ball blending function. Thus, recurrence Basis is not appropriate for computer programming.

Although the basis form is simple and easy to understand, the implementation of this method is unsuitable to apply in real programming. Several conditions and exceptions in programming are weak points of this method.

2.3.2 Recursive Algorithm

Recursive algorithm is a simple method for software implementation because most programming languages support recursive algorithms. There is a unique formula in each model. For example, Wang-Ball recursive algorithm has a formula to generate Wang-Ball curve. The formula is represented as:

$$A(t) = p_i^r(t) = \begin{cases} p_i, & r = 0, \\ p_i, & 0 \le i \le \left\lfloor \frac{n-r}{2} \right\rfloor, \\ p_{i+r+1}, & \left\lceil \frac{n-r}{2} \right\rceil < i \le n-r, \\ (1-t)p_i^{r-1}(t) + tp_{i+1}^{r-1}(t), & Otherwise. \end{cases} \tag{16}$$

Actually, a recursive algorithm is a simple method for software programming but this method is not appropriate for curve construction. It requires several computational resources and processing time $O(2^n)$. For instance, the high degree curve consumes a lot of memories in the recursive computational process.

2.3.3 Iterative Form of Recursive Algorithm

Iterative form of recursive algorithm is the repetition of a changeless process to generate a curve figure. Each step in the repeating process is called iteration. The mechanism of this technique will interpolate between two adjacent control points then calculate the new starting point of the next iterative point. The process of iteration will end when the last pair of iterative points is calculated and shown as the point of curve figure.

Iterative form of recursive algorithm can be demonstrated based on the below method.

2.3.3.1 Geometric Interpretation Diagrams

This method is represented in graphic showing the step to obtain the point on a curve from given control points. Each model has its own pattern and unique order of figuration.

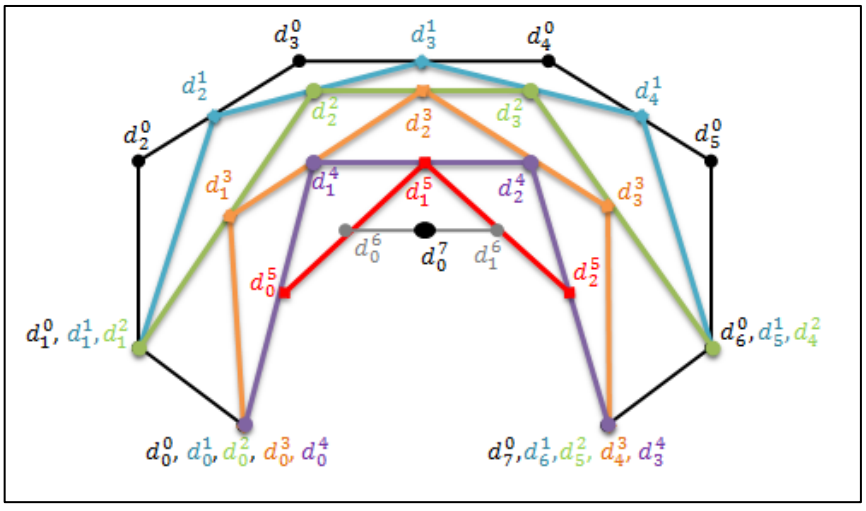


Figure 2.3 Geometric Interpolation Diagram of Dejdumrong Curve

The geometric interpolation diagram in Figure 2.3 shows how each point on the curve in curve construction process is obtained. It is useful for curve representation and curve construction. However, this method is not appropriate to apply to computer software because a higher degree curve makes the diagram complex and confusing to understand.

2.3.3.2 Triangular Scheme

The other method used to represent the curve is a triangular scheme. It is another representation of geometric interpolation diagrams in Figure 2.3 with the upside-down representation. Figure 2.4 demonstrates how points on curve can be obtained from a list of control points. It is obvious that the triangular scheme is easier for a manual calculation.

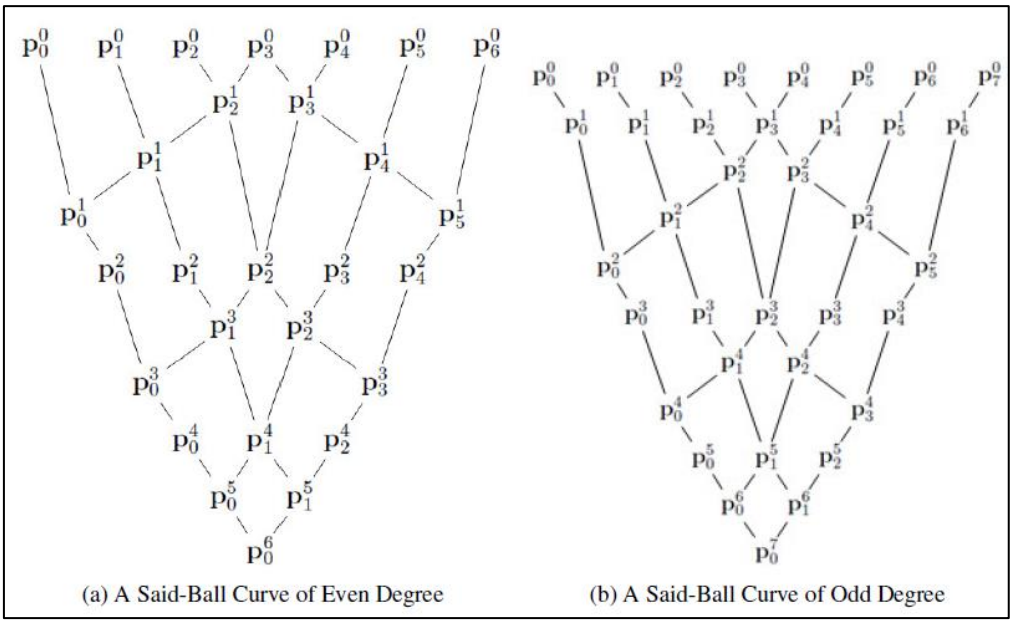


Figure 2.4 Example of Said-Ball Triangular Scheme.

The triangular scheme can be divided into two cases: odd and event degree having some different steps to construct the point. This method is not suitable to apply for software applications but it is useful for a curve representation.

The iterative form is quite simple and easy to understand the step of curve construction. However, it is not appropriate to represent in terms of programming language. This technique does not cover all curve applications such as degree elevation, degree reduction and conversion.

2.3.4 Polar Form Approach

Polar form was introduced by Ramshaw [10]. He presented a method to express the polar forms for polynomial by using differential geometry and elementary symmetric function.

In 1992, polar forms [11] were used to construct Bézier curves and B-spline curves. Polar form approach is simple for Bézier and B-spline curve constructions but it is difficult to understand for other curves. Thus, it is not popular to use in CAGD.

For instance, the polar form representing Bézier Curve degree ***n*** is expressed by

$$f(t_1, \dots, t_n) = \sum_{i=0}^{n} \binom{n}{i} \cdot \sigma_{i,n-i}^{n} \cdot (t_1, \dots, t_n) \tag{17}$$

where, $\sigma_{i,n-i}^{n} \cdot (t_1, \dots, t_n)$ is the mean of all products of form $t_{j1}, \dots, t_{jn} \cdot (1 - t_{j1+1}) \dots (1 - t_{jn})$ Furthermore, $t^i \cdot (1-t)^{n-i}$ in polynomial basis could be represented by , $\sigma_{i,n-i}^{n} \cdot (t_1, \dots, t_n)$

In addition, polar forms are used to find the relationships between Bézier and other curves. For example, it is used to convert the Bézier into Said-Ball [12], Wang-Ball [12], DP [13] can be obtained from using the polar forms. However, the explorations of conversion formulae are difficult to implement and cannot be directly obtained. Thus, the formulas to compute the properties of curve by polar forms are very complex and not practical.

2.3.5 Monomial Forms

Aphirukmatakun and Dejdumrong[12] presented the proposition of Monomial form in 2009. This technique derived from the basis function of approximation curve in the form of matrix multiplication between control points, coefficient of power basis and parameter t at each degree from 0 to n. Cubic Bézier can be represented by (18) and (19):

$$\beta(t) = \mathbf{G} \cdot \mathbf{M} \cdot \mathbf{T} \tag{18}$$

$$\beta(t) = \left[t^2 - 2t + 1\right] \cdot \mathbf{p}_0 + \left[2t^3 - 4t^2 + 2t\right] \cdot \mathbf{p}_1 + \left[-2t^3 + 2t^2\right] \cdot \mathbf{p}_2 + \left[t^2\right] \cdot \mathbf{p}_3 \tag{19}$$

where **G** is control points, i=0,1,2,…,n
M· **T** represents the 3th degree Bernstein polynomial
M is coefficient of power basis in terms of matrix
T is parameter *t*

$$\mathbf{G} = [\mathbf{p_0}, \quad \mathbf{p_1}, \quad \mathbf{p_2}, \quad \mathbf{p_3}] \tag{20}$$

$$\mathbf{M} = \begin{bmatrix} 0 & 1 & -2 & 1 \\ 2 & -4 & 2 & 0 \\ -2 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{21}$$

$$\mathbf{T} = \begin{bmatrix} t^3 \\ t^2 \\ t^1 \\ 1 \end{bmatrix} \tag{22}$$

To derive the monomial matrix, the basis function of Bézier curve will be expanded and grouped by parameter *t* in each degree. Then, the control points are factored out. The coefficient of power basis will be acquired. This group of coefficient is called monomial matrix.

2.4 Curve Properties

This section provides the detail of curve properties which are used in curve modeling such as derivative, degree elevations, degree reduction, and conversions.

2.4.1 Degree Elevation

Degree elevation is used to increase the number of degree and a number of control points. Increasing a number of control points provides the flexibility for shape adjustability. Any curve before and after applying degree elevation is the same curves.

2.4.2 Degree Reduction

Degree reduction is oppositely used to degree elevation in the way that degree reduction is used to reduce the number of degree and a number of control points. Reduction can be an optimizing process for curve construction. For instance, a curve with 50 control points can reduce the number of control points to 30 points but the shape and critical properties do not change. Thus, the process to construction is less than the original curve.

2.4.3 Derivative

Derivative is typically used to compute the tangent vector of curves. This tangent vector represents the slope of curves. In design software derivative is an important property to identify the slope that allows the designer to control the shape of curves.

2.4.4 Conversion

The conversion is used to transform one curve to another curve. This is an essential curve property which is useful to apply in CAD/CAGD software. In curve modeling, we use several models and propose a model conversion. Unfortunately, there is not any method that is perfect to represent the conversion method.

2.5 Curve Conversion

This session will focus on the conversion property which is the main issue for this thesis. From the previous part, there are several algorithms to create a curve line. However, not all methods have the conversion properties. Some algorithms can be applied into conversion method but it is not flexible enough and unusable such as the polar form approach that has all properties but it is highly complex and needs high mathematical knowledge. The traditional conversion between two models used mathematical equalization by equalizing two models and solving resulted control points of the target model. The example traditional conversion between Bézier Curve and Wang-Ball curve is shown below.

Given a sequence of n+1 control points $\boldsymbol{p}_i(i = 0,1,\dots,n)$, the Wang-Ball curve of degree n (Order n+1) defined by these points can be expressed as

$$W(t) = \sum_{i=0}^{n} \boldsymbol{p_i} \cdot A_i^n(t), \quad 0 \le t \le 1 \tag{23}$$

where $\boldsymbol{p_i}$ is control points, $i = 0,1,2,\dots,n$

$A_i^n(t)$ is the n^{th} degree Wang polynomial and can be defined for both odd and even value of n as follows:

$$A_i^n(t) = \begin{cases} (2t)^i(1-t)^{i+2} & , \ for\ 0 \le i \le \left\lfloor \frac{n}{2} \right\rfloor - 1, \\ (2t)^i(1-t)^{n-i} & , \ if\ i = \left\lfloor \frac{n}{2} \right\rfloor, \\ (2(1-t))^{n-i}t^i & , \ if\ i = \left\lceil \frac{n}{2} \right\rceil, \\ A_{n-i}^n(1-t) & , \ for \left\lceil \frac{n}{2} \right\rceil + 1 \le i \le n. \end{cases} \tag{24}$$

The Bézier curve of degree n can be expressed by Wang-Ball control point as follows:

$$W_0 = b_0 \tag{25}$$

$$W_n = b_n \tag{26}$$

$$W_k = \frac{1}{2^k}\left[\binom{n}{k}\boldsymbol{b_k} - \sum_{i=0}^{k-1} 2^i \binom{n-2-2i}{k-i} W_i - \sum_{i=n-k+1}^{n} 2^{n-i}\binom{2i-2-n}{i-k} W_i\right], \quad \text{for } k < \left\lfloor \frac{n}{2} \right\rfloor \tag{27}$$

$$W_k = \frac{1}{2^{n-k}}\left[\binom{n}{k}\boldsymbol{b_k} - \sum_{i=0}^{n-k} 2^i \binom{n-2-2i}{k-i} W_i - \sum_{i=k+1}^{n} 2^{n-i}\binom{2i-2-n}{i-k} W_i\right], \quad \text{for } k > \left\lceil \frac{n}{2} \right\rceil \tag{28}$$

$$W_k = \frac{1}{2^k}\left[\binom{n}{k}\boldsymbol{b_k} - \sum_{i=0}^{k-1} 2^i \binom{n-2-2i}{k-i} W_i - \sum_{i=k+2}^{n} 2^{n-i}\binom{2i-2-n}{i-k} W_i\right], \quad \text{for } k = \left\lfloor \frac{n}{2} \right\rfloor \tag{29}$$

$$W_k = \frac{1}{2^{n-k}}\left[\binom{n}{k}\boldsymbol{b_k} - \sum_{i=0}^{k-2} 2^i \binom{n-2-2i}{k-i} W_i - \sum_{i=k+1}^{n} 2^{n-i}\binom{2i-2-n}{i-k} W_i\right], \quad \text{for } k = \left\lceil \frac{n}{2} \right\rceil \tag{30}$$

After calculating Wang-Ball control points, they will be used to construct a Wang-Ball curve by Wang's algorithm to draw a line curve.

This traditional conversion is a single conversion. There is a conversion for a pair of model with a fixed degree of curve. It is interesting to employ it since we can create a multi-conversion method and improve the computational performance of the conversion process. It will be also useful for a modeler to design more flexible tools.

Thus, the curve conversion method that can represent a multi-conversion, universal conversion concept that is also easy to understand is monomial form conversion.

Monomial form conversion possesses desirable characteristics. It can represent all CAGD curves and support most of the curve applications. It is not a single calculation. Other algorithms are a single process that means it cannot process 2 applications in one process. On another hand, monomial form can convert and continuously process the degree elevation simultaneously by adding a degree elevation matrix. The result is a converted target model with a higher degree than the input curve model. Monomial form is the interesting method to implement in the software usability.

Most research about monomial form is in the field of CAGD curve domain. In the field of interpolation, the control point curve is mentioned only Newton-Lagrange Curves. In the interpolation domain, there are many interesting applications such as forecasting and analyzing programs. Thus, to integrate both categories of curves will increase performances and methods to create a new useful software.

The relationship between the interpolation curve and the approximation curve is the focal issue to investigate a new conversion method. One of many motivating criteria is Chebyshev polynomial.

2.6 Chebyshev Polynomial

Chebyshev polynomial is a popular approximated polynomial. There is a perpendicular property which makes equation stable. Chebyshev polynomial [14] has the period of time between [-1, 1]. The simple form of chebyshev polynomial is the first kind Chebyshev. The Chebyshey polynomial is formally defined as follows:

$$P_n(t) = a_0 + a_1 t + a_{2n}(2t-1) + \cdots + a_n(2tT_{n-1} - T_{n-2}) \tag{31}$$

where

$\emptyset_i = T_i = 2tT_{i-2}$ $when\ i > 1\ and\ T_0 = 1, T_1 = t.$

Chebyshev polynomial is one of the schemes we consider to integrate in our solution. Chebyshev polynomial can be represented in terms of power basis equation (32)

$$T_n(t) = \sum_{i=0}^{\frac{n}{2}} c_i^n t^{n-2i} \tag{32}$$

when

$$C_i^n = (-1)^i 2^{n-2i-1} \cdot \frac{n}{n-i}\binom{n-i}{i} \tag{33}$$

Equation (33) represents a coefficient function for Chebyshev polynomial.

2.7 Overview of Previous Research

In this section, we discuss the previous work that we are going to extend in this thesis.

In 2009, Aphirukmatakun and Dejdumrong presented the concept of monomial form approach for CAGD curve with their applications. This technique represents the polynomial curve in terms of matrix operation. Monomial Matrix [12] presents approximation curve relationship and its useful curve properties such as derivatives, degree elevations, degree reductions and conversions among approximation curves. However, Aphirukmatakun did not provide the proofs for his proposition. The proofs of monomial form for CAGDs curve was published in 2012-2013 by Savetseranee and Dejdumrong [17][18].These papers make monomial form stronger and more interesting. Monomial forms have useful curve utilities and are applicable for computer programming. However, [12] did not provide the application for the interpolation curve. Most methods only support the approximation curve.

Monomial form is a good technique to be applied in the curve conversion algorithm because it can represent the universal conversion concept. Monomial method is a good technique to represent CAGD curve conversion. All curve models can be converted into each other via the monomial method. Increasing the domain of conversion is development to improve monomial conversion. There are some research works that integrate the interpolation curve domain with the CAGD curve such as Newton-Lagrange monomial form. However, Newton-Lagrange does not cover all conversions between the interpolation control point curve and the approximation control point curve.

Thus, this paper will extend the utility of monomial form to the interpolation control point curve. To this end, we will investigate the relation between the approximation control point curve and the interpolation control point curve in terms of curve application. Then, we will propose a model conversion to compliment the shortfall of approximation curve and deliver the seamless conversion to CAGD modeling tool.

CHAPTER 3 METHODOLOGY

From the previous chapter, there are several methods in CAGD curve construction. Different methods are dissimilar properties and abilities in conversion. There are no completely covered all conversion CAGD models. However, monomial form can be used to be a universal conversion. Although monomial form can convert all models in CAGD curves, it does not support all models in conversion between CAGD curves and interpolation control point curves.

This chapter presents the proposed solution of how the conversion between the approximation curve (CAGD curves) and the interpolation curve is performed. This could facilitate the efficient transformation of curves into geometric modeling tools such as CAGD applications. Our conversion model is based on the examination of the relationship between the approximation curve and the interpolation curve. We employ the monomial form to account for the conversion process. The conversion between two kinds of curve represents the relationship between them.

The organization of this chapter begins with the definition of interpolation curve and overall concept of the conversion between interpolation curve and approximation curve. Then, the detailed conversion process is given. Finally, proof of concept of our proposed curve conversion scheme is presented.

3.1 Concept and Definition

This part depicts two major constructs of our proposed solution: interpolation curve representation and monomial form conversion between interpolation and approximation curve.

3.1.1 Interpolation Curve Fitting

Definition 1: The interpolation curve is a curve which generates the figure passing along every control point.

{{7, -2}, {10, 7}, {16, 3}, {25, 0}, {37, 8}}

The curve fitting is shown in Figure 3.1.

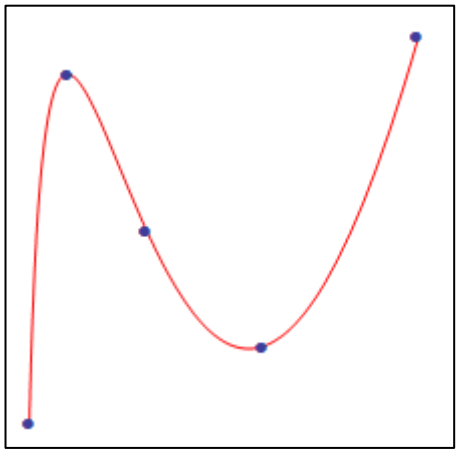


Figure 3.1 The Interpolation Curve with its Control Points

Definition2: The approximation curve is CAGD curve that is composed of popular models used in CAGD curve design, such as Bézier Curve, Wang-Ball curve, Said-Ball Curve, DP curve NB1 curve and Dejdumrong curve.

Definition 3: Curve representation is represented by a set of control points and blending functions. The interpolation curve uses specific blending functions to represent interpolated characteristic called Chebyshev polynomials.

To create interpolation curve, the first step is to initialize a set of control points. Curve direction and simple characteristic are shown by linear interpolation of control points. Second, the polynomial is used to draw the curve figure. Chebyshev is the polynomial used in this research. The below figure shows the interpolation curve generation process. The generated curve figure should walk through every control point. However, the curves may face an ill- conditioned problem [9] where there is a swing of figure from compressing the curve to pass the control point. This problem can be solved by using a knot parameter. The knot parameter can reduce the swing of figuration since it comprises the period of curve construction processes.

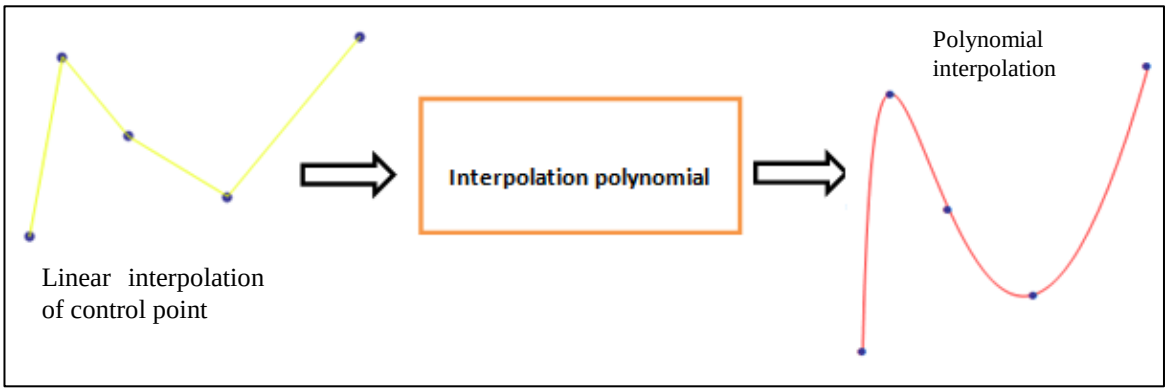


Figure 3.2 Interpolation Curve Constructions

3.1.2 Conversion between Interpolation Curve and Approximation Curve

The conversion process is a core focus of our research. Here, we apply monomial form to represent curves. The conversion concept is to transform ChebyShev into other models by using monomial matrix conversion. Let monomial form of Chebyshev be equal to the monomial form of any approximation model as follows:

$$\mathrm{m} \cdot \mathbf{T}^* \cdot t = \boldsymbol{x} \cdot \mathbf{Y} \cdot t \quad (34)$$

$$\mathrm{m} \cdot \mathbf{T}^* = \boldsymbol{x} \cdot \mathbf{Y} \quad (35)$$

$$\mathrm{m} \cdot \mathbf{T}^* \cdot \mathbf{Y}^{-1} = \boldsymbol{x} \cdot \mathbf{Y} \cdot \mathbf{Y}^{-1} \quad (36)$$

$$\boldsymbol{x} = \mathrm{m} \cdot \mathbf{T}^* \cdot \mathbf{Y}^{-1} \quad (37)$$

Finally, the new control point of approximation curve represented by $\boldsymbol{x}$ can be calculated as (37)

The interpolation curve can be computed by using transformation matrix , $\boldsymbol{\delta}$. A set of interpolation control points (Interpolation curve), $\{m_i\}_{i=0}^{n}$. can be converted to approximation curve, $\{r_i\}_{i=0}^{n}$ as follows:

$$[r_0 \quad r_1 \quad \cdots \quad r_n] = [m_0 \quad m_1 \quad \cdots \quad m_n] \cdot \boldsymbol{\delta}\,, \tag{38}$$

where $\boldsymbol{\delta}$ is a transformation defined in Table 3.1

Table 3.1 Conversion from Interpolation Curve to Approximation Curve

From	To	Transformation Matrix
Interpolation Curve	Bézier	$\boldsymbol{\delta} = \mathbf{T}^* \cdot \mathbf{B}^{-1}$
	Said-Ball	$\boldsymbol{\delta} = \mathbf{T}^* \cdot \mathbf{S}^{-1}$
	Wang-Ball	$\boldsymbol{\delta} = \mathbf{T}^* \cdot \mathbf{A}^{-1}$
	DP	$\boldsymbol{\delta} = \mathbf{T}^* \cdot \mathbf{C}^{-1}$
	NB1	$\boldsymbol{\delta} = \mathbf{T}^* \cdot \mathbf{N}^{-1}$
	Dejdumrong	$\boldsymbol{\delta} = \mathbf{T}^* \cdot \mathbf{D}^{-1}$

In contrast, the transformation inverse from the approximation curve to the interpolation curve is the inverse conversion from Table 3.1. Let $\{\boldsymbol{m}_i\}_{i=0}^{n}$ be the interpolation control point and $\{\boldsymbol{r}_i\}_{i=0}^{n}$ be the approximation control point. Any approximation curve can be converted into the interpolation curve as follows:

$$[m_0 \quad m_1 \quad \cdots \quad m_n] = [r_0 \quad r_1 \quad \cdots \quad r_n] \cdot \boldsymbol{\delta}\,, \tag{39}$$

Where $\boldsymbol{\delta}$ is transformation defined in Table 3.2

Table 3.2 Conversion from Approximation Curve to Interpolation Curve

From	To	Transformation Matrix
Bézier	Interpolation Curve	$\boldsymbol{\delta} = \mathbf{B} \cdot \mathbf{T}^{*-1}$
Said-Ball		$\boldsymbol{\delta} = \mathbf{S} \cdot \mathbf{T}^{*-1}$
Wang-Ball		$\boldsymbol{\delta} = \mathbf{A} \cdot \mathbf{T}^{*-1}$
DP		$\boldsymbol{\delta} = \mathbf{C} \cdot \mathbf{T}^{*-1}$
NB1		$\boldsymbol{\delta} = \mathbf{N} \cdot \mathbf{T}^{*-1}$
Dejdumrong		$\boldsymbol{\delta} = \mathbf{D} \cdot \mathbf{T}^{*-1}$

3.2 Proposed Steps of Conversion

The process to convert an interpolation curve to any approximation curve consists of the following algorithmic processes:

Input: Control points of interpolation curve

Output: Approximation curve: Wang-Ball curve, Said-Ball curve, Dejdumrong curve, and etc.

1. Construct an interpolation curve.
2. Represent the interpolation curve to the Chebyshev curve
3. Convert the interpolation curve control point to the Chebyshev control point
3. Convert the Chebyshev curve to a monomial form.
4. Transfer the control point of Chebyshev curve to the approximation curve control point by the monomial matrix conversion.
5. Construct the approximation curve with the resulted control point.

The conversion method is shown as the figure below.

```
BEGIN
        1.  Convert Interpolation Control Point to Chebyshev Control Point
        chebychevControlPoint =ChebychevCP( pointOnline )
        n = Length( pointOnline )
        i = 0;
        p =     FOR  j from 0 to  n-1
                        IF  n = 3   THEN
                                p =  j / (n - 1)
                        IF  j = 1    THEN
                                p = j / (n - 1)/2
                        IF  j = (n - 2)  THEN
                                p =  1 - ( 1/(n - 1)/2 )
                        ELSE
                                 p = j / (n - 1)
                END FOR
        ChebyshevCurve( n , p )
                        IF  n=1 THEN
                                Cheb = p/p
                        IF  n=2 THEN
                                Cheb = (2*p) – 1
                        ELSE
                                Cheb = ( (2*( (2*p) - 1) * ChebyshevCurve (n - 1, p) )
                                                - ChebyshevCurve (n - 2, p) )
        Return(Cheb)
        Tn( pt ) = Table
                        FOR n from 1 to  Length( pt )
                                ChebyshevCurve( n, p )
                        END FOR
                  End Table
        Inverse (matrix)
                Invs =( 1/det(matrix) ×  matrix )
        Return(Invs)
        T = Tn( pointOnline )
        T((1)) = 1
        Controlp(pt ) = Transpose( pt )  ×  Inverse(T)
        CP = Controlp (pointOnline)
        dstar = TABLE
                        FOR j from1 to n-i
                          FOR k from 1 to 2
                                CP( ( k ) ), ( ( j ) )
                          END FOR
                        END FOR
                END TABLE
        d = Transpose( dstar );
        Return(d );
```

```
2.  Convert Chebyshev Control Point into Approximation Control Point
  ConversionItoA ( chebyshevMonomialForm, approxMonomialForm, chebyshevControlPoint )
  n = Length (chebyshevControlPoint)
  CP = chebyshevControlPoint
  Inverse (matrix)
            Invs =( 1/det(matrix) ×  matrix )
  Return(Invs)
  T* = Transpose (chebyshevMonomialForm )
  B = Inverse (  approxMonomialForm  )
  δ =  T*  × B
   approximationControlPoint = δ  × CP
  Return (approximationControlPoint)
END
```

Figure 3.3 Conversions from Interpolation Curve to Approximation Curve Algorithm

In contrast, the approximation curves can be converted to the interpolation curve by the following algorithm:

Input: Control points of Approximation curve: Wang-Ball curve, Said-Ball curve, Dejdumrong curve and etc.

Output: Interpolation curve with control points

1. Construct an approximation curve.
2. Convert the approximation curve to a monomial form.
3. Transfer the control point of approximation curve to the Chebyshev curve control point by the monomial matrix conversion.
4. Construct the Chebyshev curve with the resulted control point.
5. Convert the Chebyshev control point to the interpolation control points
6. Construct the interpolation curve.

The conversion method in terms of programming shown as the figure below.

```
BEGIN
    1.  Convert Approximation Control Point to Chebyshev Control Point
    chebyshevControlPoint =ConversionAtoI ( approxMonomialForm, chebyshevMonomialForm,
    approximationControlPoint )
      n = Length (approximationControlPoint)
      CP = approximationControlPoint
      Inverse (matrix)
                Invs =( 1/det(matrix) ×  matrix )
      Return(Invs)
      A = Transpose ( approxMonomialForm )
      T*  = Inverse (  chebyshevMonomialForm  )
      δ = A  ×  T*
      chebyshevControlPoint = δ  × CP
      Return (chebyshevControlPoint)
```

```
2.  Convert Chebyshev Control Point to Interpolation Control Point
InterpolationPoint(chebyshevControlPoint)
n = Length(chebyshevControlPoint)
i = 0;
p =       FOR  j from 0 to  n-1
                  IF  n = 3   THEN
                           p =  j / (n - 1)
                  IF  j = 1    THEN
                           p = j / (n - 1)/2
                  IF  j  = (n - 2)  THEN
                           p =  1 - ( 1/(n - 1)/2 )
                  ELSE
                            p = j / (n - 1)
          END FOR
 ChebyshevCurve( n , p )

                  IF  n=1 THEN
                           Cheb = p/p
                  IF  n=2 THEN
                           Cheb = (2*p) – 1
                  ELSE
                           Cheb = ( (2*( (2*p) - 1) * ChebyshevCurve (n - 1, p) )
                                        - ChebyshevCurve (n - 2, p) )
Return(Cheb)
Tn( pt ) =    TABLE
                  FOR n from 1 to  Length( pt )
                           ChebyshevCurve( n, p )
                  END FOR
              END TABLE
Controlp( pt ) = Transpose( pt ) × T
T = Tn(chebyshevControlPoint )
T((1)) = 1
CP = Controlp(chebyshevControlPoint)
Return( Transpose (CP ) )
END
```

Figure 3.4 Conversions from Approximation Curve to Interpolation Curve Algorithm

3.3 Chebyshev Curve

3.3.1 Chebyshev Blending Function in Conversion

In our proposed scheme, the conversion of interpolation curve and approximation curve is based on Chebyshev polynomial. The problem is that the period of Chebyshev polynomial is between [-1, 1] as the equation (40). It is not a normal pattern for curve construction. Thus, period shifting is lifted to [0, 1]

$$\mathrm{T_n}(t^*) = 2t^*\mathrm{T_{n-1}}(t^*) - \mathrm{T_{n-2}}(t^*) \tag{40}$$

To shift the period [-1,1] to [0, 1] by the relation of Chebyshev polynomial [15] .

$$t^* = 2t_\mathrm{i} - 1 \text{ when } 0 \le t_i \le 1 \tag{41}$$

To substitute (41) into (40) can be derived to (42)

$$\mathrm{T_n^*}(t) = 2(2t-1)\mathrm{T_{n-1}}(2t-1) - \mathrm{T_{n-2}}(2t-1) \tag{42}$$

Equation (42) is recursive function to construct Chebyshev polynomial.

There is a set of control points on the interpolation curve. Then, find the coefficient of Chebyshev polynomial. It represents the control point of Chebyshev curve.

$\boldsymbol{c_x}$ and $\boldsymbol{c_y}$ are coordinate values of chebyshev control point. To find the curve figuration point or the point on the line figure can be calculated by each coordinate X and coordinate Y.

$$\boldsymbol{c_x} \cdot [\mathbf{T_n^*}] = \boldsymbol{x} \tag{43}$$

$$\boldsymbol{c_y} \cdot [\mathbf{T_n^*}] = \boldsymbol{y} \tag{44}$$

Equation (43) and (44) show configurations of Chebyshev curve representing the chebyshev control point. Here, $\mathrm{T_n^*}$ is blending function represented by the recursive form as shown in equation (42).
The interpolation curve consists of the control points on the line figure that can be represented by $\boldsymbol{x}$ and $\boldsymbol{y}$.

The first step to convert the interpolation curve is to find each Chevbyshev control point $\boldsymbol{c_x}, \boldsymbol{c_y}$ from interpolation coordinate value by using equation (45) and (46)

$$\boldsymbol{c_x} = [\mathbf{T_n^*}]^{-1} \cdot \boldsymbol{x} \tag{45}$$

$$\boldsymbol{c_y} = [\mathbf{T_n^*}]^{-1} \cdot \boldsymbol{y} \tag{46}$$

Thus,

$\boldsymbol{c_x, c_y}$ represent the control point of Chebyshev curve.

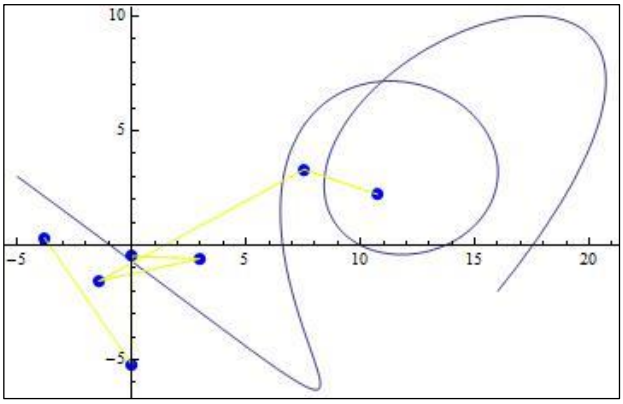


Figure 3.5 The Relationship between Curve Figure and Chebyshev Control Polygon.

3.3.2 Chebyshev Monomial Form

Next, the method used to convert the interpolation curve in terms of Chebyshev polynomial to approximation Curve is the monomial form of Chebyshev. This form is represented by multiplication matrix

$$\text{Chebyshev Curve} = \mathbf{G_c} \cdot \mathbf{M_c} \cdot \mathbf{T} \tag{47}$$

where

$\mathbf{G_c}$ is a set of control point

$\mathbf{M}_c$ is Monomial coefficient matrix for Chebyshev Curve

$\mathbf{T}$ is parameter t

In conversion, $\mathbf{M_c}$ can be represented in a conversion form by shifting the parameter t period from [-1,1] to [0, 1] .

$$\mathrm{T_n}(t) = \sum_{\mathrm{i}=0}^{\frac{\mathrm{n}}{2}} \mathrm{c_i^n} t^{\mathrm{n}-2\mathrm{i}} \tag{48}$$

When M_c[-1,1]

$$\mathrm{C_i^n} = (-1)^{\mathrm{i}} 2^{\mathrm{n}-2\mathrm{i}-1} \cdot \frac{\mathrm{n}}{\mathrm{n}-\mathrm{i}} \binom{\mathrm{n}-\mathrm{i}}{\mathrm{i}} \tag{49}$$

Equation (49) is monomial coefficient matrix in domain [-1,1] while

(48) is the normal basis of ChebyShev polynomial from[16] The relation of $T_n^*(t)$ in domain [0,1] is

$$T_n(t^*) = T_n^*(t) \tag{50}$$

Thus,

$$\begin{aligned} T_{2n}(t) &= \cos 2n\theta \\ &= \cos n(2\theta) \\ &= T_n(\cos 2\theta) \\ &= T_n(2\theta\cos^2\theta - 1) \\ &= T_n(2t^2 - 1) \\ &= T_n(t^{*^2}) \end{aligned} \tag{51}$$

$$T_{2n}\left(\sqrt{t}\right) = T_n(t^*) \tag{52}$$

The equation (48) using normal period domain from [-1 to 1]. It can be derived by substituting equation (52) with equation (53) in terms of parameter *t*

$$T_n(t^*) = \sum_{i=0}^{n} c_i^{2n} t^{n-i} \tag{53}$$

when $\mathbf{M_c}$[0,1]

$$C_i^{2n} = (-1)^i 2^{2n-2i-1} \cdot \frac{2n}{2n-i}\binom{2n-i}{i} \tag{54}$$

Now, the parameter $\mathbf{M_c}$ can be calculated by using equation (54).

3.4 Proof of Concept in the Conversion

This part is a demonstration of conversion algorithm used in this research. The representation of curve is monomial form. It is composed of three parameters in terms of convolution matrix. **G** ·**M**·**T** where **G** is a set of control points, **M** is coefficient matrix depending on type of curve. **T** is paramedic *t*

Definition 3: two curves are equal or are the same curve as the construction equation is equal. Curve conversion is the application to transform a curve model to another model which is the same curve but has different construction parameters.

Control point is the most essential parameter to create a curve. Curve conversion can define a conversion process to calculate and to find a new set of control points or the control points in the expected model.
Thus, conversion from Chebyshev curve to Bézier Curve is described as follows:

$$\text{Chebyshev Curve} \quad = \quad \text{Bézier Curve} \tag{55}$$

$$\mathbf{G_c} \cdot \mathbf{M_c} \cdot \mathbf{T} \quad = \quad \mathbf{G_B} \cdot \mathbf{M_B} \cdot \mathbf{T} \tag{56}$$

where

$\mathbf{G_{C \to B}}$ is a new set of control points (expect model)

$$\mathbf{G_c} \cdot \mathbf{M_c} \cdot \mathbf{T} \quad = \mathbf{G_{C \to B}} \cdot \mathbf{M_B} \cdot \mathbf{T} \tag{57}$$

$$\mathbf{G_c} \cdot \mathbf{M_c} \cdot \mathbf{T} \quad = \mathbf{G_{C \to B}} \cdot \mathbf{M_B} \cdot \mathbf{T} \tag{58}$$

$$\mathbf{G_c} \cdot \mathbf{M_c} \quad = \mathbf{G_{C \to B}} \cdot \mathbf{M_B} \tag{59}$$

$$\mathbf{G_c} \cdot \mathbf{M_c} \cdot [\mathbf{M_B}]^{-1} \quad = \mathbf{G_{C \to B}} \tag{60}$$

Thus, a new set of control points which is Bézier control point is the below equation.

$$\mathbf{G_{C \to B}} \quad = \mathbf{G_c} \cdot \mathbf{M_c} \cdot [\mathbf{M_B}]^{-1} \tag{61}$$

According to definition 3, the result of curve conversion must be the same curve with the original one.

Proof of conversion can be demonstrated by converting the third degree Chebyshev Curve (interpolation curve) into Cubic Bézier curve (Approximation curve) by mathematic calculation as follows.

$$\mathbf{G_c} = [p_0, p_1, p_{2,} p_3] \tag{62}$$

$$\mathbf{M_{c\ (3rd\ degree)}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ 1 & -8 & 8 & 0 \\ -1 & 18 & -48 & 32 \end{bmatrix} \tag{63}$$

$$\mathbf{M_{B(Cubic\ Bezeir)}} = \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{64}$$

$$\mathbf{T} = \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} \tag{65}$$

From equation (61)

$$\mathbf{G_{C\to B}} = \mathbf{G_c}\cdot\mathbf{M_c}\cdot[\mathbf{M_B}]^{-1}$$

$$= [p_0, p_1, p_2, p_3]\cdot\begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & -\frac{1}{3} & -\frac{5}{3} & 5 \\ 1 & \frac{1}{3} & -\frac{5}{3} & -5 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (66)$$

(67) can be derived by substituting (61) into (56) in terms of parameter G_B

$$\mathbf{G_c}\cdot\mathbf{M_c}\cdot\mathbf{T} = [\mathbf{G_c}\cdot\mathbf{M_c}\cdot[\mathbf{M_B}]^{-1}]\cdot\mathbf{M_B}\cdot\mathbf{T} \quad (67)$$

$$[p_0, p_1, p_2, p_3]\cdot\begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ 1 & -8 & 8 & 0 \\ -1 & 18 & -48 & 32 \end{bmatrix}\cdot\begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}$$

$$= [p_0, p_1, p_2, p_3]\cdot\begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & -\frac{1}{3} & -\frac{5}{3} & 5 \\ 1 & \frac{1}{3} & -\frac{5}{3} & -5 \\ 1 & 1 & 1 & 1 \end{bmatrix}\cdot\begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}\cdot\begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} \quad (68)$$

Then, solve the (68) by multiplying GMT parameters with both sides shown in (69) and (70).

$$p_0 + p_1(-1+2t) + p_2(1-8t+18t^2) + p_3(-1+18t-48t^2+32t^3)$$

$$= [p_0, p_1, p_2, p_3]\cdot\begin{bmatrix} 1 & -6 & 12 & -8 \\ 1 & -4 & 0 & 8 \\ 1 & -2 & -4 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\cdot\begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} \quad (69)$$

$$p_0 - p_1 + p_2 - p_3 + 2p_1t - 8p_2t + 18p_3t + 8p_2t^2 - 48p_3t^2 + 32p_3t^3$$
$$=$$
$$p_0 - p_1 + p_2 - p_3 + \left(-3(p_0 - p_1 + p_2 - p_3) + 3\left(p_0 - \frac{p_1}{3} - \frac{5p_2}{3} + 5p_3\right)\right)t$$
$$+\left(3\left(p_0 + \frac{p_1}{3} - \frac{5p_2}{3} - 5p_3\right) + 3(p_0 - p_1 + p_2 - p_3) - 6\left(p_0 - \frac{p_1}{3} - \frac{5p_2}{3} + 5p_3\right)\right)t^2$$
$$+(2p_1 - 3(p_0 + \tfrac{p_1}{3} - \tfrac{5p_2}{3} - 5p_3) + 2p_3 + 3(p_0 - \tfrac{p_1}{3} - \tfrac{5p_2}{3} + 5p_3))t^3 \quad (70)$$

In addition, expand both sides of equation then rewrite the form of equation which displays in equation (71)

$$p_0 - p_1 + p_2 - p_3 + 2p_1t - 8p_2t + 18p_3t + 8p_2t^2 - 48p_3t^2 + 32p_3t^3$$
$$=$$
$$p_0 - p_1 + p_2 - p_3 + 2p_1t - 8p_2t + 18p_3t + 8p_2t^2 - 48p_3t^2 + 32p_3t^3 \quad (71)$$

Thus, both curves are equal. The result form conversion algorithm is the same one with original curve.

In this chapter, we presented the methodology to interpret the relationship between the interpolation curve and the approximation curve into the conversion method. Definitions of technical terms used to define the methodology were provided. The explanation was given to steps and process that occur in conversion algorithm such as interpolation curve fitting by Chebyshev polynomial and monomial form of Chebyshev which is the mechanism to convert control points. In addition, it showed the proof of conversion as mathematic equations.

In the next chapter, experiments and results from this conversion method will be illustrated.

CHAPTER 4 EXPERIMANTAL RESULT

This chapter explains experiments and results of the transformation of interpolation curve to approximation curve. In addition, the result of the conversion of Chebyshev control points to target approximation control point is discussed.

To implement the curve conversion algorithm, we set up a test scenario and evaluation as follows:

- Conversion experimental test
  - Test with 4 standard curves, namely, Monotonic curve , Cusp curve, Loop curve and Sinuate curve
  - Perform the conversion test between interpolation and each approximation curve
  - Investigate the appropriate degree in conversion
  - Collect output, time, round, and curve character.
- Check the correctness of conversion between the converted curve and the original one

4.1 Converting Interpolation Curve to Approximation Curve with Standard Curve

This part describes the conversion from the interpolation curve to the approximation curve. The approximation curve in this test is composed of Bézier curve, Said-Ball curve, Wang-Ball curve, NB1 curve, DP curve, and Dejdumrong curve. The experiment covers four standard samples: monotonic curve, cusp curve, loop curve, and sinuate curve

4.1.1 Test Case: Monotonic Curve

Table 4.1 Detail of Input: Monotonic Curve

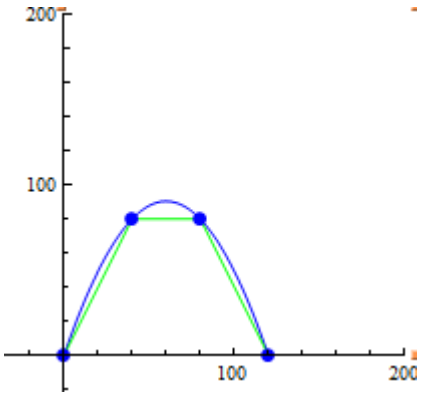
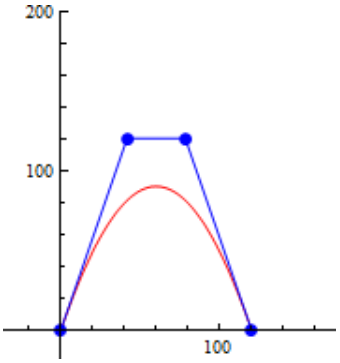
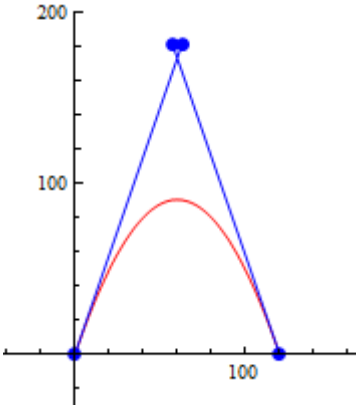
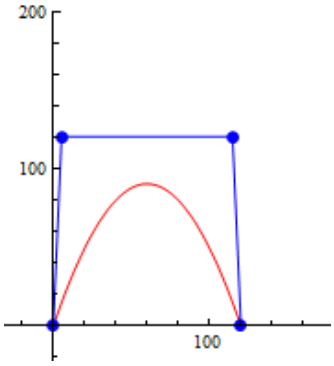
Degree of Interpolation curve	3
Interpolation control point	{{0,0},{40,80},{80,80},{120,0}}
	

Table 4.2 Results of The Conversion from Monotonic Interpolation Curve to Each CAGD Curve

Convert to Model	Conversion Time(ms)	New control point [Approximation control point]	Curve figure
Bézier	2.34002	{{0,0},{41,120},{78,120}, {120,0}}	
Dejdumrong	2.96402		
Wang-Ball	2.65202	{{0,0},{63,181},{57,181}, {120,0}}	
Said-Ball	2.65202		
NB1	3.43202		
DP	3.43202	{{0,0},{5,121},{115,121} ,{120,0}}	

From Table 4.2, there are some sets of new control points which are the results of the conversion and they gave similar point such as Bézier and Dejdumrong curve. This case occurs when input curves have a low degree and are not a complex curve. In a higher degree, the new control points in each CAGD model will be harder to be duplicated.

4.1.2 Test Case: Cusp Curve

Table 4.3 Detail of Input: Cusp Curve

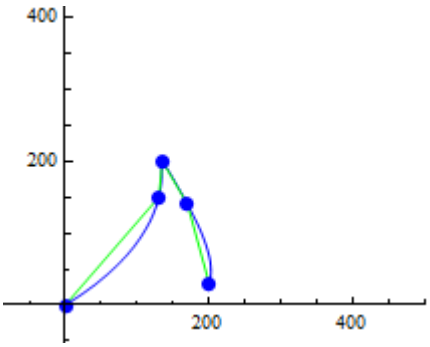
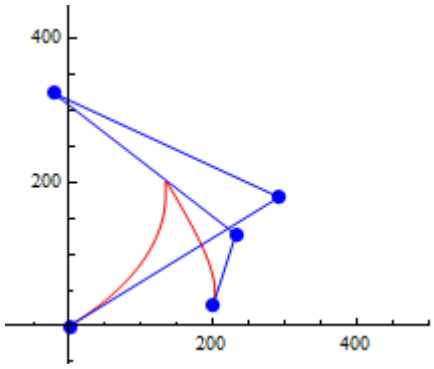
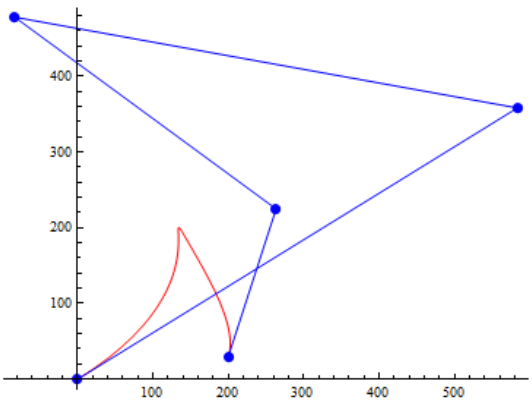
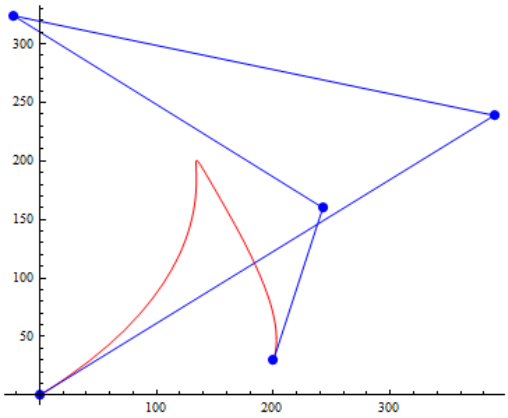
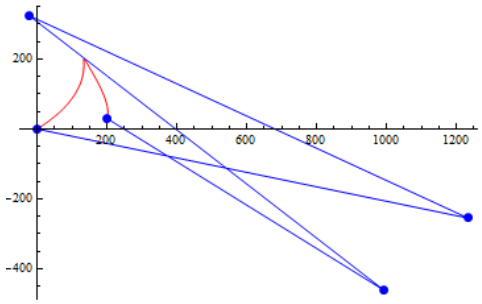
Degree of Interpolation curve	4
Interpolation control point	{{0,0},{130,150},{135,200}, {170,140},{200,30}}
	

Table 4.4 Results of Conversion from Cusp Interpolation Curve to Each CAGD Curve

Convert to Model	Conversion Time(ms)	New control point [Approximation control point]	Curve figure
Bézier	3.90003	{{0,0},{292,179}, {-22,324},{232,128}, {200,30}}	
Wang-Ball	4.68003	{{0,0},{583,358}, {-83,478},{263,225}, {200,30}}	
NB1	4.99203		

Said-Ball	4.68003	{{0,0},{389,239}, {-22,324},{242,160}, {200,30}}	
Dejdumrong	4.52403		
DP	4.83603	{0,0},{1233,-255}, {-22,324}, {993,-462}, {200,30}}	

This simple forth degree cusp curve can be converted into 4 groups of control points. The individual models are Bézier and DP. The second group is Wang-Ball and NB1. The last group is Said-Ball and Dejdumrong curves.

4.1.3 Test Case: Loop Curve

Table 4.5 Detail of Input: Loop Curve

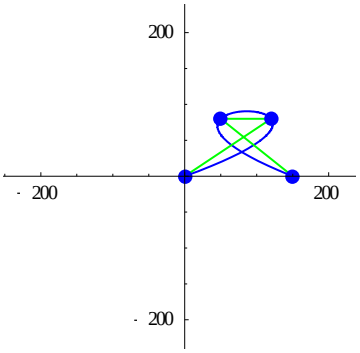
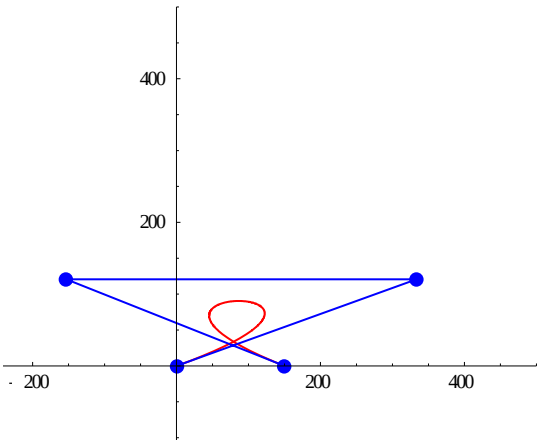
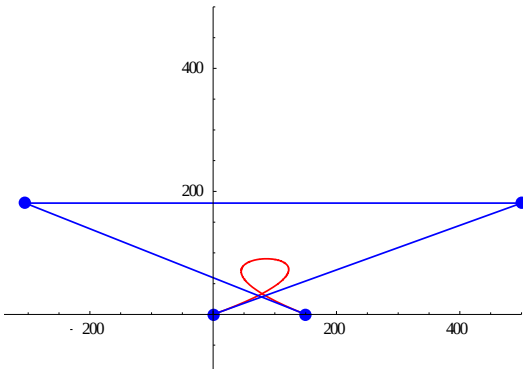
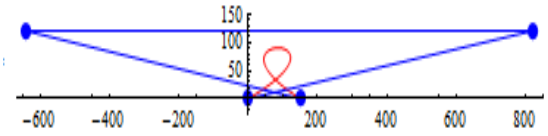
Degree of Interpolation curve	3
Interpolation control point	{{0,0},{120,80},{50,80},{150,0}}
	

Table 4.6 Results of Conversion from Loop Interpolation Curve to Each CAGD Curve

Convert to Model	Conversion Time(ms)	New control point [Approximation control point]	Curve figure
Bézier	2.18401	{{0,0},{333,121}, {-153,121}, {150,0}}	
Dejdumrong	2.65202		
Wang-Ball	2.96402	{{0,0},{500,181}, {-305,181}, {150,0}}	
Said-Ball	2.80802		
NB1	3.12002		

DP	3.43202	{{0,0},{820,121}, {-640,121}, {150,0}}	

This test case is a simple cubic looping curve. There are three resulting groups. Bézier and Dejdumrong have the same control points. The second one is the group of Wang-Ball, Said-Ball, and NB1. DP is a single model as the last group of result.

4.1.4 Test Case: Sinuate Curve

Table 4.7 Detail of Input: Sinuate Curve

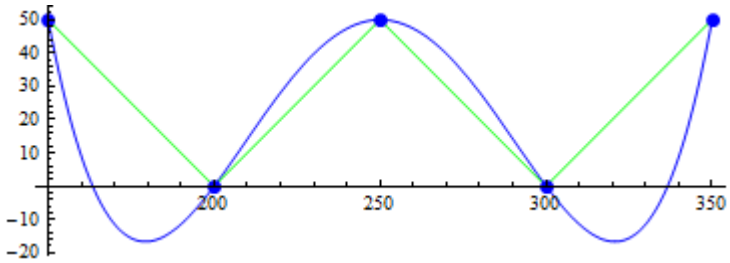
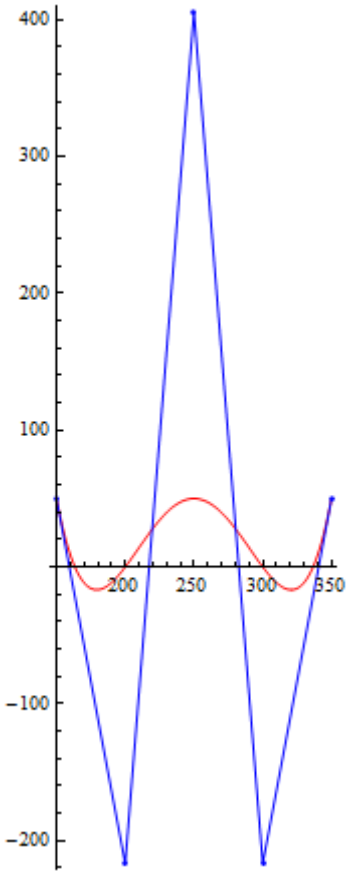
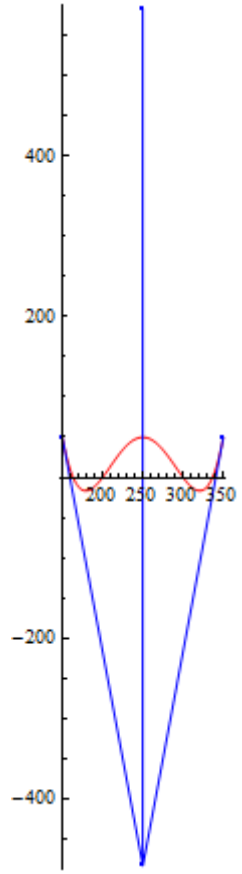
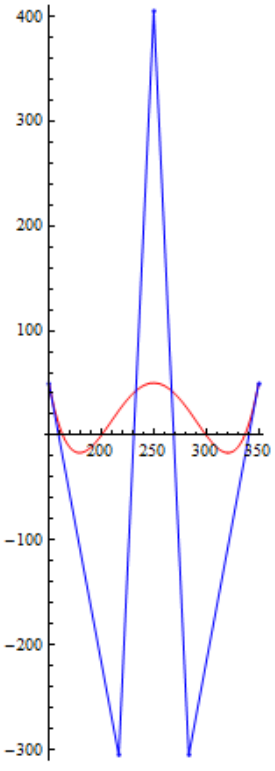
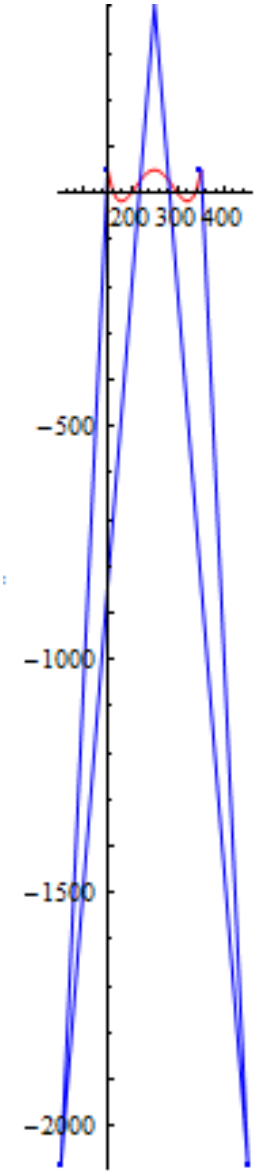
Degree of Interpolation curve	4
Interpolation control point	{{150, 50}, {200, 0}, {250, 50}, {300, 0}, {350, 50}}
	

Table 4.8 Results of Conversion from Sinuate Interpolation Curve to Each CAGD Curve

Convert to Model	Conversion Time(ms)	New control point [Approximation control point]	Curve figure
Bézier	3.58802	{{150,50}, {200,-217}, {250,406}, {300,-217}, {350,50}}	

Wang-Ball	4.68003	{{150,50}, {250,-483}, {250,583}, {250,-483}, {350,50}}	
NB1	4.83603		
Said-Ball	4.36803	{{150,50}, {217,-306}, {250,406}, {283,-306}, {350,50}}	
Dejdumrong	4.68003		

DP	4.83603	{{150, 50.}, {50.,-2083}, {250.,406}, {450,-2083}, {350,50}}	

The sinuate curve is the last conversion test case. It used the forth degree of sinuate curve. There are 4 sets of results which are converted from the interpolation curve. Bézier and DP are individual models. Wang-Ball and NB1 are the second group and Said-Ball and Dejdumrong are the last one.

4.2 Approximation Curve to Interpolation Curve with Standard Curve

This section presents the conversion from the CAGD curve to the interpolation curve with 4 standards. The experimental result reveals that the input and output curve is equal and it is the same curve before and after converting.

Table 4.9 Test Cases: Convert Bézier Curve to Interpolation Curve

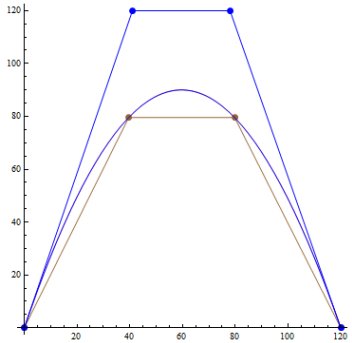
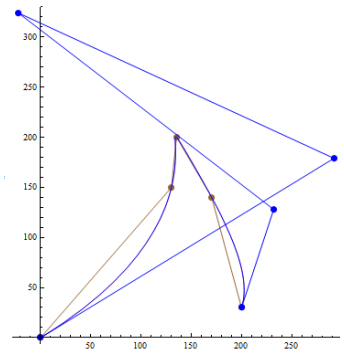
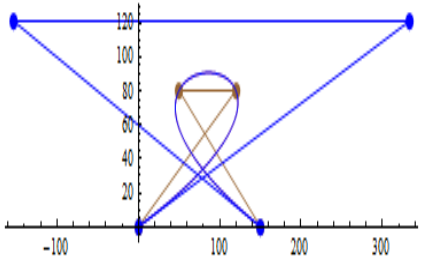
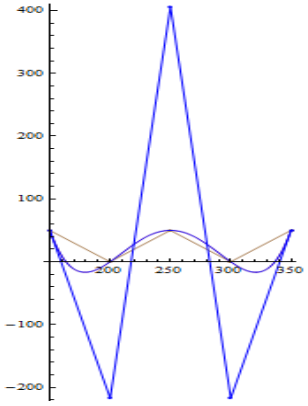
Convert Bézier curve to interpolation curve				
Type of curve	Monotonic Curve	Cusp Curve	Loop Curve	Sinuate Curve
Curve Degree	3	4	3	4
Bézier Control Point	{{0,0},{41,120},{78,120},{120,0}}	{{0,0},{292,179},{-22,324},{232,128},{200,30}}	{{0,0},{333,121},{-153,121},{150,0}}	{{150,50},{200,-217},{250,406},{300,-217},{350,50}}
Bézier Curve figure				
Conversion Time(ms)	1.56001	2.18401	1.56001	2.18401
Interpolation Control Point	{{0,0},{40,80},{80,80},{120,0}}	{{0,0},{130,150},{135,200},{170,140},{200,30}}	{{0,0},{120,80},{50,80},{150,0}}	{{150,50},{200,0},{250,50},{300,0},{350,50}}

Table 4.10 Test Cases: Convert Wang-Ball Curve to Interpolation Curve

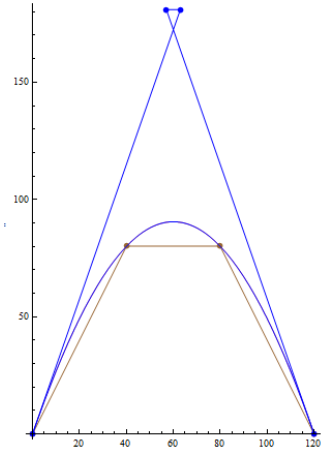
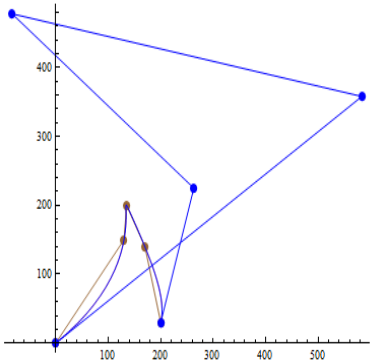
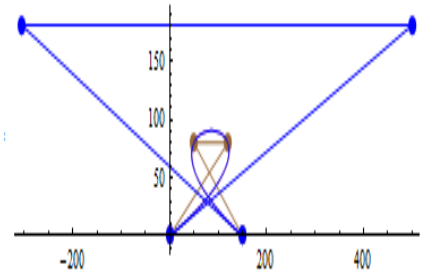
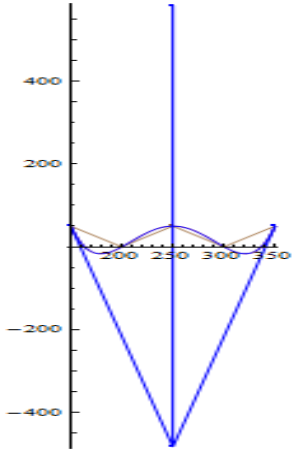
ConvertWang curve to interpolation curve				
Type of curve	Monotonic Curve	Cusp Curve	Loop Curve	Sinuate Curve
Curve Degree	3	4	3	4
Wang Control Point	{{0,0},{63,181},{57,181},{120,0}}	{{0,0},{583,358},{-83,478},{263,225},{200,30}}	{{0,0},{500,181},{-305,181},{150,0}}	{{150,50},{250,-483},{250,583},{250,-483},{350,50}}
Wang Curve figure				
Conversion Time(ms)	2.18401	3.12002	2.18401	3.12002
Interpolation Control Point	{{0,0},{40,80},{80,80},{120,0}}	{{0,0},{130,150},{135,200},{170,140},{200,30}}	{{0,0},{120,80},{50,80},{150,0}}	{{150,50},{200,0},{250,50},{300,0},{350,50}}

Table 4.11 Test Cases: Convert Said-Ball Curve to Interpolation Curve

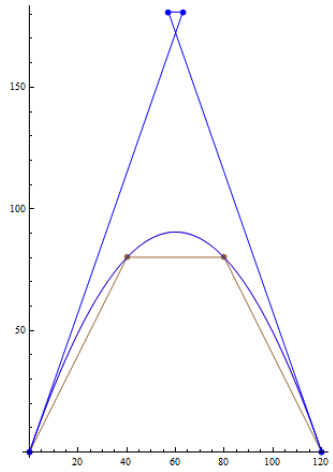
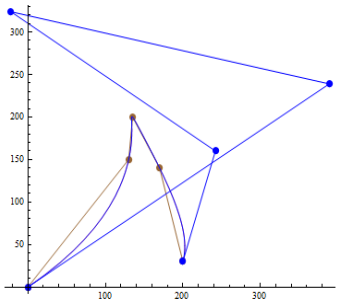
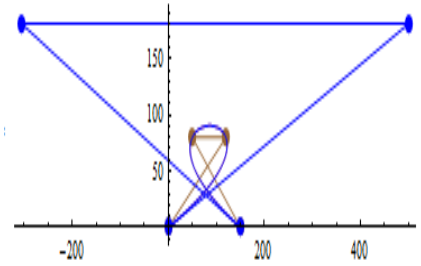
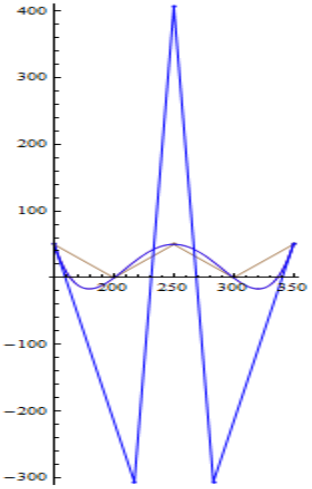
Convert Said curve to interpolation curve				
Type of curve	Monotonic Curve	Cusp Curve	Loop Curve	Sinuate Curve
Curve Degree	3	4	3	4
Said Control Point	{{0,0},{63,181},{57,181},{120,0}}	{{0,0},{389,239},{-22,324},{242,160},{200,30}}	{{0,0},{500,181},{-305,181},{150,0}}	{{150,50},{217,-306},{250,406},{283,-306},{350,50}}
Said Curve figure				
Conversion Time(ms)	2.18401	3.27602	2.18401	3.43202
Interpolation Control Point	{{0,0},{40,80},{80,80},{120,0}}	{{0,0},{130,150},{135,200},{170,140},{200,30}}	{{0,0},{120,80},{50,80},{150,0}}	{{150,50},{200,0},{250,50},{300,0},{350,50}}

Table 4.12 Test Cases: Convert DP Curve to Interpolation Curve

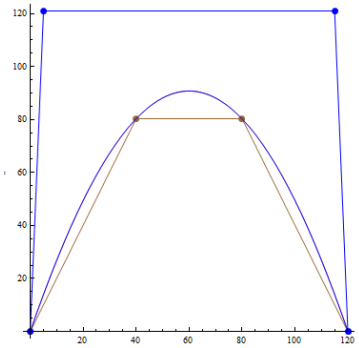
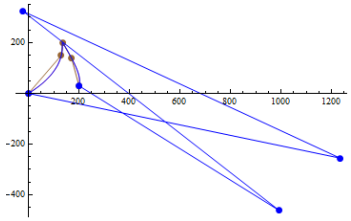
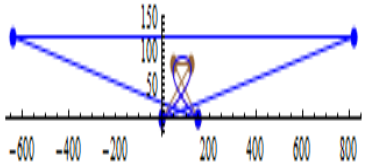
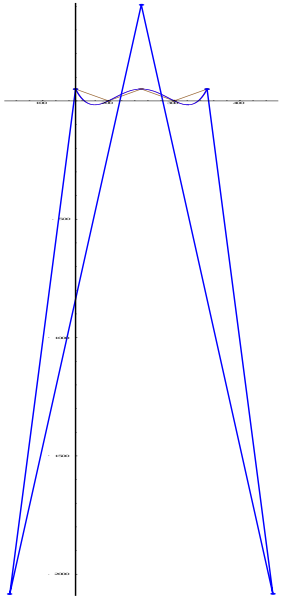
Convert DP curve to interpolation curve				
Type of curve	Monotonic Curve	Cusp Curve	Loop Curve	Sinuate Curve
Curve Degree	3	4	3	4
DP Control Point	{{0,0},{5,121},{115,121}, {120,0}}	{0,0},{1233,-255}, {-22,324},{993,-462}, {200,30}}	{{0,0},{820,121}, {-640,121},{150,0}}	{{150,50},{50,-2083.33},{250,406}, {450,-2083},{350,50}}
DP Curve figure				
Conversion Time(ms)	2.80802	3.27602	2.80802	3.27602
Interpolation Control Point	{{0,0},{40,80},{80,80},{120,0}}	{{0,0},{130,150},{135,200}, {170,140},{200,30}}	{{0,0},{120,80},{50,80}, {150,0}}	{{150,50},{200,0},{250,50}, {300,0},{350,50}}

Table 4.13 Test Cases: Convert NB1 Curve to Interpolation Curve

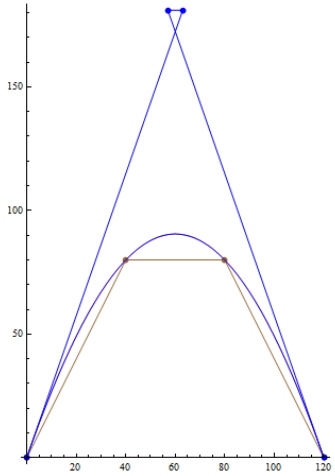
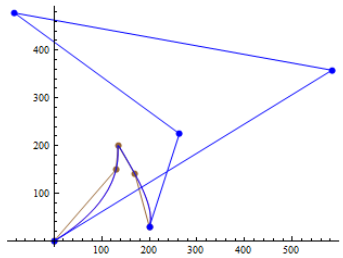
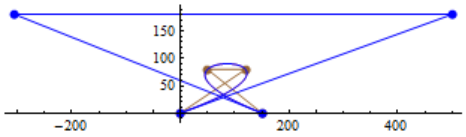
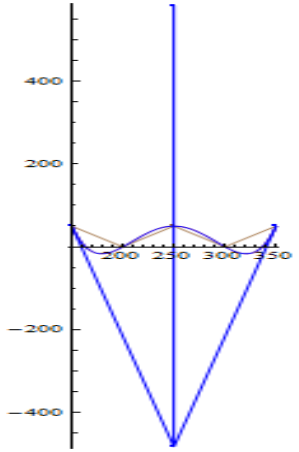
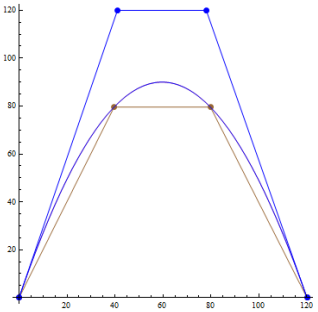
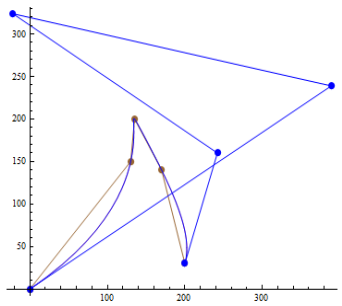
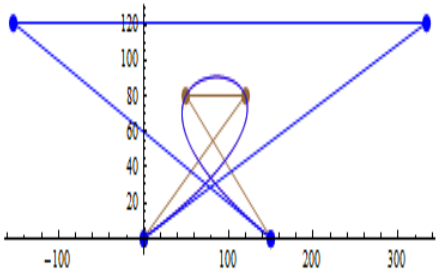
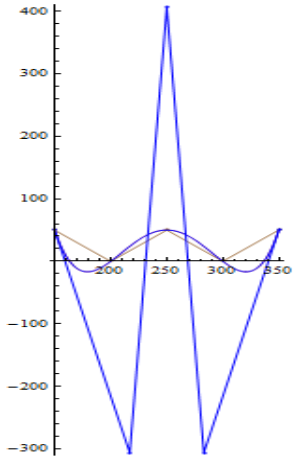
Convert NB1curve to interpolation curve				
Type of curve	Monotonic Curve	Cusp Curve	Loop Curve	Sinuate Curve
Curve Degree	3	4	3	4
NB1 Control Point	{{0,0},{63,181},{57,181}, {120,0}}	{{0,0},{583,358}, {-83,478},{263,225}, {200,30}}	{{0,0},{500,181}, {-305,181},{150,0}}	{{150,50},{250,-483},{250,583}, {250,-483},{350,50}}
NB1 Curve figure				
Conversion Time(ms)	2.80802	3.58802	2.65202	3.58802
Interpolation Control Point	{{0,0},{40,80},{80,80}, {120,0}}	{{0,0},{130,150},{135,200}, {170,140},{200,30}}	{{0,0},{120,80},{50,80},{150,0}}	{{150,50},{200,0},{250,50}, {300,0},{350,50}}

Table 4.14 Test Cases: Convert Dejdumrong Curve to Interpolation Curve

Convert Dejdumrong curve to interpolation curve				
Type of curve	Monotonic Curve	Cusp Curve	Loop Curve	Sinuate Curve
Curve Degree	3	4	3	4
Dejdumrong Control Point	{{0,0},{41,120}, {78,120},{120,0}}	{{0,0},{389,239}, {-22,324},{242,160}, {200,30}}	{{0,0},{333,121}, {-153,121},{150,0}}	{{150,50},{217,-306}, {250,406},{283,-306}, {350,50}}
Dejdumrong Curve figure				
Conversion Time(ms)	2.34002	2.96402	1.87201	3.27602
Interpolation Control Point	{{0,0},{40,80},{80,80}, {120,0}}	{{0,0},{130,150},{135,200}, {170,140},{200,30}}	{{0,0},{120,80},{50,80},{150,0}}	{{150,50},{200,0},{250,50}, {300,0},{350,50}}

4.3 Discussion on the Conversion Test Case

The above tables show the conversion test cases. There are both forward conversions from the interpolation curve into CAGD curves and inverse conversion to the interpolation curve. The purposes of these testings are to obtain the result of conversion by using 4 standard curves and examine curve appearances after the conversion. According to the results, there are two major findings which are significant to the implementation.

The first issue is about new control points set from conversion. When the interpolation curve is converted into CAGD curves with a low degree, lower than 10th, a set of new control points is possible to be duplicated with other models. As shown in Table 4.2, Wang-Ball, Said-Ball and NB1 return a similar set of converted control point as {{0, 0}, {63,181}, {57,181}, {120, 0}}and it constructs same monotonic curves. This issue takes place several times because a lower degree of standard curves is not complex and the small number of control points is not enough to influence the curve representation. A new control point is the result of rounding value after the calculation process. It is not the real value which is decimal in float variable type. However, there are no influences on the correctness of conversion because the computer graphic screen shows the line figure as integer value. The decimal results have to be rounded up as integer numbers.

The second issue is the time used. Timing in conversion can be a simple performance factor. From the test case tables, there are time records in each conversion. These values were calculated from 100 times of conversion then they were all calculated in average by mean. The duration of time that each model spent in the conversion processes was ranked from least to highest, which is presented as follows:

1. Bézier
2. Dejdumrong
3. Wang-Ball
4. Said-Ball
5. DP
6. NB1

The rank was calculated from the average time used to convert each model and related with the result tables above. The result also shows that conversions from any approximation curve (CAGD curve) took less time than those from the interpolation curve to the CAGD curve. The inverse recursive function which calculates the interpolation control point to Chebyshev control point shows the overall time consumption. It required 15% more from the time of conversion.

In addition, the above table shows that the conversion scheme can support and represent the conversion between the interpolation curve and the CAGD curve.

4.4 Newton-Lagrange Curve and Chebyshev Curve Comparison

In this section, we compare results of curve fitting between two models, Newton-Lagrange and Chebyshev curves. Since Newton Lagrange is generally adopted to represent the interpolation curve, we thus take it into our comparative analysis with our employed technique, Chebyshey. These two techniques are compared according to their curve fitting and conversion capability.

Table 4.15 Strength Comparison between Newton-Lagrange and Chebyshev Conversion Method.

Functional Capabilities	Newton-Lagrange	Chebyshev
Curve Fitting		
Simple Curve	Good	Good
Complex Curve	Not Flexible	Flexible*
Support For Real Application	Not Good	Good*
Ill- Conditioned	Sensitive	Lower Sensitive
Conversion		
Processing Steps	Direct Conversion	Indirect Conversion
Accuracy (% of Correctness)	99.95%	99.95%
Time of Processing	2.496×10^{-3} Second	5.304 10^{-3} Second
Computational Complexity	$O(n^3)$ $= O(n^2 \log n + n^3 + n^2)$	$O(n^3)$ $=O(4n^2 - n + n^2 \log n + n^2 \log n + n^3 + n^2)$

*Jaroensawad,R.,[16] mention in the Paper

Tables 4.15 presents the comparative features between Newton-Lagrange and our proposed method. Regarding the curve fitting capability, it is able to determine the suitability of interpolation curve representation. The table shows that Chebyshev is flexible to be applied with complex curve and real applications. This information is based on the work conducted by Jaroensawad, R.[16] . She also provided the reason to support that Chebyshev is better than Newton-Lagrange for their applied work on the approximated digital hand drawing curve. Furthermore, Chebyshev is more robust to the ill-conditioned curves.

Newton-Lagrange is a polynomial that represents the interpolation curve and can be applied to use with a monomial form in several curve applications such as degree elevation, degree deduction and conversion. In this work, conversion is our focused application. Before converting curves, curve fitting is the first process that is undertaken.

Newton-Lagrange and new proposed method Chebyshev are both interpolation curve fitting representations. However, the Newton-Lagrange curve is sensitive for the ill-conditioned curves especially the first and last control points. On the other hand, Chebyshev is more suitable to work with complex curves and shapes.

The ill-conditioned in Newton-Lagrange curve appears with a high degree of curve and some shape such as, a zigzag curve or a sinuate curve. The followings are the examples of the ill-conditioned of Newton-Lagrange curve compared with curve which is represented by Chebyshev polynomial.

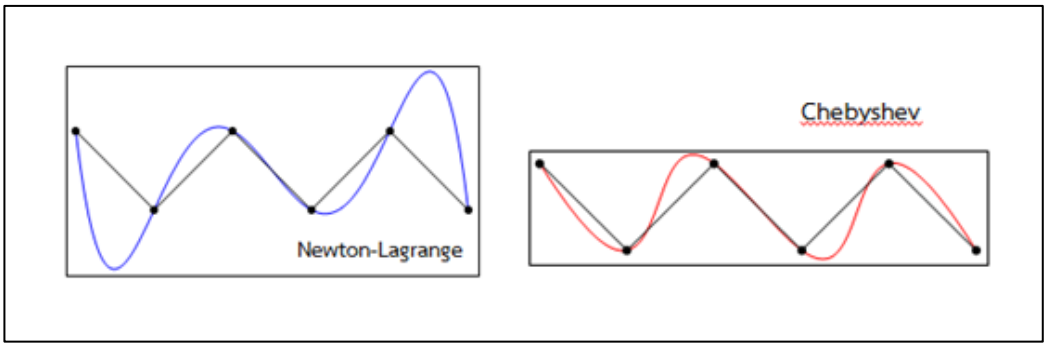


Figure 4.1 Sinuate Ill-Conditioned Case in Newton-Lagrange Curve Compared with Chebyshev Curve

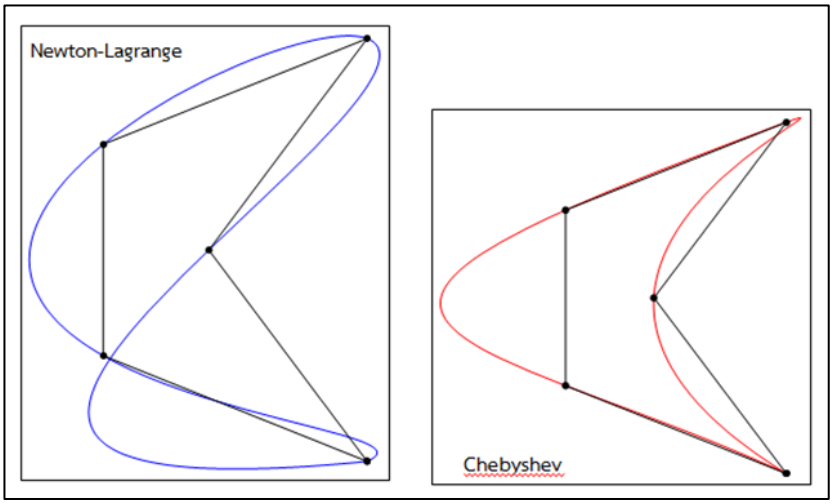


Figure 4.2 Complex Shape Ill-Conditions in Newton-Lagrange Curve Compared with Chebyshev Curve

Figure 4.2 shows the complex shape like a boomerang. The result of interpolation curve fitting shows that with the same control points, Chebyshev curve is more realistic than Newton-Lagrange curve. The blue line figure in the last pair of control point swings out of the control polygon. The shape of blue one is not correct as expected.

There are some other cases of the ill-conditioned of Newton-Lagrange polynomial in cusp curve model and loop model as shown in figure 4.3

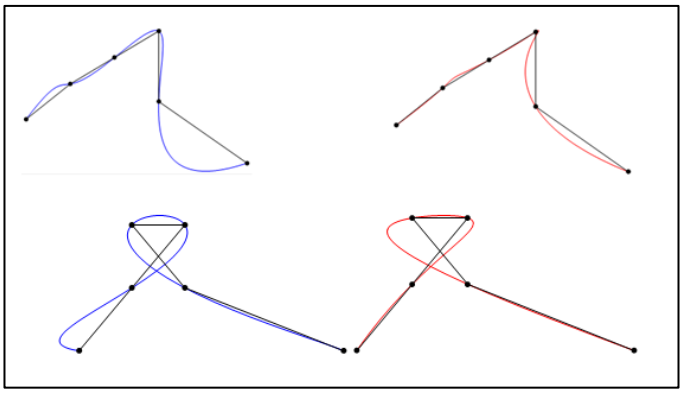


Figure 4.3 Newton-Lagrange Ill-Conditioned Case in Loop and Crust Curve (Blue) Compare with Chebyshev (Red)

Lower the ill-conditioned cases are a strong point applied to the proposed method Chebyshev to fit the interpolation model.

Another functional capability is the conversion issues. The comparison of the conversion result between two methods is composed of steps of conversion, the correctness after converting, computational time and computational complexity.

Chebyshev conversion cannot directly use the interpolation control point. It has to convert the interpolation control point to the Chebyshev control point before converting to the approximation model. Thus, Chebyshev conversion has two steps in its conversion process.

The conversion results from both Newton-Lagrange and Chebyshev demonstrate the same percentage of correctness at 99.95% because the polynomial equation of original curve and conversion output curve is equal and has the same polynomial. Thus, the accuracy of both conversion methods is the same.

Chebyshev consumes more time of computation when compared with Newton-Lagrange. It is twice of Newton-Lagrange time processing. It relates to the computational complexity as Newton-Lagrange provides lower complexity. However, the worst case of both methods are $O(n^3)$.

There are 2 steps of the conversion by Chebyshev polynomial. The first step is the calculation from interpolation control points to Chebyshev control points or from Chebyshev control points to interpolation control points. This step is based on the recursive algorithm with is represented by Chebyshev polynomial. Chebyshev control point is the result between interpolation control point and inverse of Chebyshev polynomial.

$$\{\boldsymbol{c_x}, \boldsymbol{c_y}\} = \{\boldsymbol{x}, \boldsymbol{y}\} \cdot [T_n]^{-1} \tag{72}$$

where

$\{\boldsymbol{c_x}, \boldsymbol{c_y}\}$ is Chebyshev Control Points
$\{\boldsymbol{x}, \boldsymbol{y}\}$ is interpolation Control Points
T_n is Chebyshev Polynomial

According to equation 72, there are multiple inverse matrices with control points and therefore the computational complexity is $O(n^2 \log n)$. It is calculated from $O(4n^2 - n + n^2 \log n)$. The complexity of inverse matrix is $O(n^2 \log n)$ which is the *BigO* in this process.
Thus, the inverse calculation from Chebyshev control points to interpolation control points is the result of Chebyshev control points that are substituted into Chebyshev polynomial.

$$\{\mathbf{x}, \mathbf{y}\} = \{\boldsymbol{c_x}, \boldsymbol{c_y}\} \cdot T_n \tag{73}$$

The computational complexity in calculation interpolation control point is $O(n^2)$ = $O(2n^2 + 2n^2 - n)$. There are multiplication and addition only from matrix multiplication process. Multiplication *is* $O(2n^2)$ and addition is $O(2n^2 - n)$

The second step is the conversion from the Chebyshev model to the approximation model or any approximation to the Chebyshev model. They have the same computational complexity calculation.

$$\{\boldsymbol{a_x}, \boldsymbol{a_y}\} = \{\boldsymbol{c_x}, \boldsymbol{c_y}\} \cdot [\mathbf{M_c}]^{-1} \cdot \mathbf{M_{Approximation}} \tag{74}$$

$$\{\boldsymbol{c_x}, \boldsymbol{c_y}\} = \{\boldsymbol{a_x}, \boldsymbol{a_y}\} \cdot [\mathbf{M_{Approximation}}]^{-1} \cdot \mathbf{M_C} \tag{75}$$

where

$\{\boldsymbol{a_x}, \boldsymbol{a_y}\}$ is Approximation Control Points
$\mathbf{M_C}$ is Monomial coefficient matrix for Chebyshev Curve
$\mathbf{M_A}$ is Monomial coefficient matrix for Approximation Curve

The complexity of equation 74 and equation 75 is equal to the multiplication between inverse matrix and coefficient matrix which are then multiplied with control points.
Then, the complexity is $O(n^3)$. The result is from $O(n^2 \log n + n^3 + n^2)$. Inverse matrix is $n^2 \log n$ then n^3 is from coefficient matrix multiplication and finally the n^2 is from the multiplication between control points and the result of coefficient matrix multiplication.

Lastly, the computational complexity of Chebyshev conversion method is calculated as follows. This is based on the conversion of interpolation curve to approximation curve that comprises interpolation control point calculation and conversion from the Chebyshev control points.

$$O(4n^2 - n + n^2 \log n) + O(n^2 \log n + n^3 + n^2)$$
$$= O(4n^2 - n + n^2 \log n + n^2 \log n + n^3 + n^2)$$
$$= O(n^3)$$

From the approximation curve to the interpolation control point.

$$O(2n^2 - n) + O(n^2 \log n + n^3 + n^2)$$
$$= O(2n^2 - n + \log n + n^3 + n^2)$$
$$= O(n^3)$$

This computational complexity entails the reason why the conversion from interpolation into approximation is more complex and needs more computational time than the inverse conversion.

Next is the computational complexity of Newton-Lagrange conversion. This technique directly converts the interpolation control point to the approximation control point. Thus, the computational complexity of this technique is $O(n^3)$ because all computational processes are the same as equation 74 and 75.

4.5 Evaluation Conversion Results

This section gives the comparative analysis between two curves, input curve and result after the curve conversion. To obtain the physical representation of curves, there is an algorithm used to compare two curves from the starting point to the end point with 1000 sampling point of time. Then, the algorithm also compares both corresponding values between input and output curves by Euclidean distance.

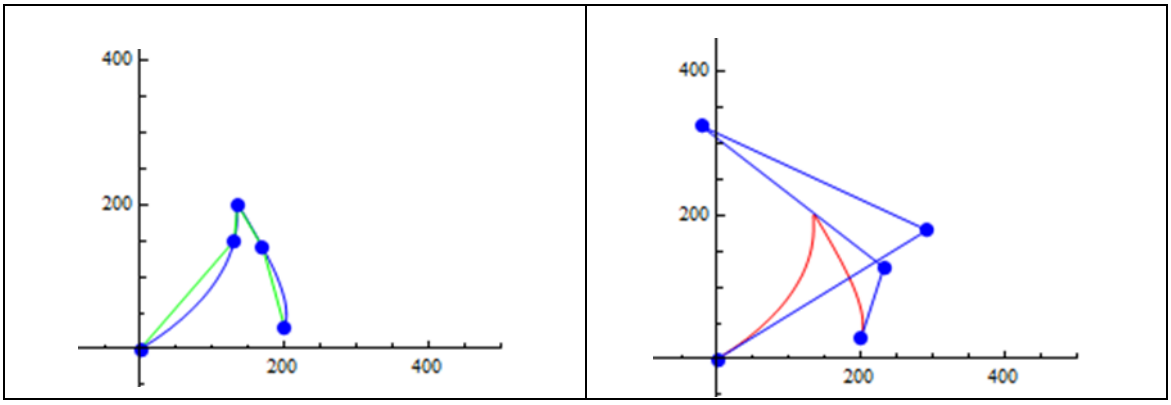


Figure 4.4 The Instance of Comparison Result between Input and Output Curve

Table 4.16 Random Sampling Point to Evaluate Input and Output Curve for Example.

Random time	Input curve (interpolation)	Output curve (Bézier)
$t = 0.05$	{12.9046,14.7739}	{12.9046,14.7739}
$t = 0.35$	{86.9146,81.7189}	{86.9146,81.7189}
$t = 0.5$	{121.25,93.75}	{121.25,93.75}
$t = 0.75$	{170.391,78.0469}	{170.391,78.0469}
$t = 1$	{200,30}	{200,30}

Table 4.16 is the example of comparison curves having random time between 0 and 1. The point is on the curve figure with time ***t***. The result from comparison input and all output curves shows that all output curves from the conversion method are 100 percent similar to the input curves as in Figure 4.4 because both input and output are the same polynomial. However, the rounding of value of point takes place from the calculation process; therefore, the accuracy of conversion is 99.95%. There is 0.05 percent of error.

In general CAGD curve, there are several degrees of curve. This depends on the utility or the application. This conversion method can convert multiple degrees of curve. However, a higher degree of input curve has a direct influence on the complexity and time of calculation. For this reason, , in the next section, the analysis of the appropriate degree for this conversion method will be discussed.

4.6 Influence of Curve Degree to Conversion Performance

This section provides statistical information of time used in the conversion. This is an average processing time. To forecast the required time of conversion in each input curve degree, measurement is required to appropriately shape the soundness of the conversion.

Table 4.17 Record of Average Time in Conversion

.

Conversion	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Interpolation to CAGDs	1.43	2.18	3.41	5.54	8.61	13.42	20.57
CAGDs to Interpolation	1.14	2.13	2.73	3.80	4.91	7.10	10.45
Conversion	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Interpolation to CAGDs	30.34	45.47	66.04	97.06	148.33	213.36	366.11
CAGDs to Interpolation	16.17	23.32	36.43	54.89	85.88	137.07	238.58

Time Unit is Millisecond (ms)

Table 4.17 shows a record of average value of processing time from both directions of conversion. The processing time in each degree of conversion is to find the mean from conversion between each CAGD curve and interpolation curve by the conversion method. Then, it uses the record to build model and predict the processing time that occurs when input curve has a higher degree. The record is plotted as Figure 4.5.

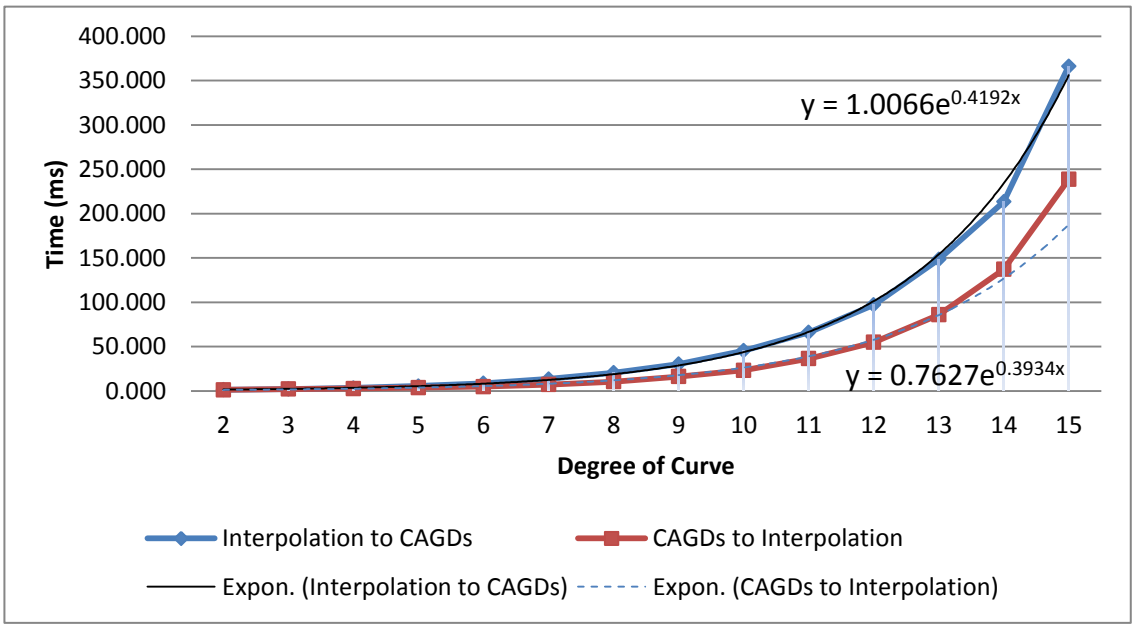


Figure 4.5 Time Performance and Influence of Curve Degree.

From Figure 4.5, there are two series of data. The first one is processing time from converting the interpolation to CAGDs curve which has more slopes with exponential equation $y = 1.0066e^{0.4192x}$. The second one is processing time of conversion back from CAGDs to the interpolation curve. The second series is represented by $y = 0.7627e^{0.3934x}$. Both exponential equations can define the relation of input curve degree and processing time.

The statistical information from Figure 4.5 and both exponential equations can be interpreted that the high degree of input curve is not appropriate to this method because of more processing time required. The appropriate degree for this conversion method is between (3, 10). The gap between conversion from interpolation to CAGDs and conversion back is larger when the input curve is higher. The cause of differential time of process comes from an inverse function that calculates Chebyshev control points to interpolation control points on the figure line and the computational complexity as mentioned in the previous part that conversion from interpolation to approximation is more complex than converse conversion. As a result, the appropriate input for this conversion method is presented and described as statistical data.

The next part shows how to increase the performance and flexibility of the method.
There are some techniques that can decrease processing time for instance, generating the coefficient matrix before reading from files. It can help optimize the calculation step. The performance of conversion can increase flexibility in curve fitting by using a knot parameter.

4.7 Increase Performance by Knot Parameter

Knot parameter can be defined as the weight of time between each pair of control point. To draw a curve line, it uses time domain 0 to 1. Curve is stated at $t = 0$ then ends at $t = 1$. A curve can be separated by a piece of curve segment between each pair of control point. In a normal case, the time in each segment is distributed as equal weight and summation of all segments has to be equal to 1. For example, there are 5 control points to draw a curve. There are 4 segments of time between each control point. The weight of each segment is equal {0.25, 0.25, 0.25, 0.25}. Each 0.25 is a parameter called knot.

Assigning a knot parameter with different weight directly changes curve characteristics. This technique can be employed to increase the flexibility and the efficiency in the conversion algorithm. Chebyshev curve representing the interpolation curve can integrate the knot value to upgrade the curve fitting ability. More flexible shapes in design could help a designer and a user create an aspiring shape.

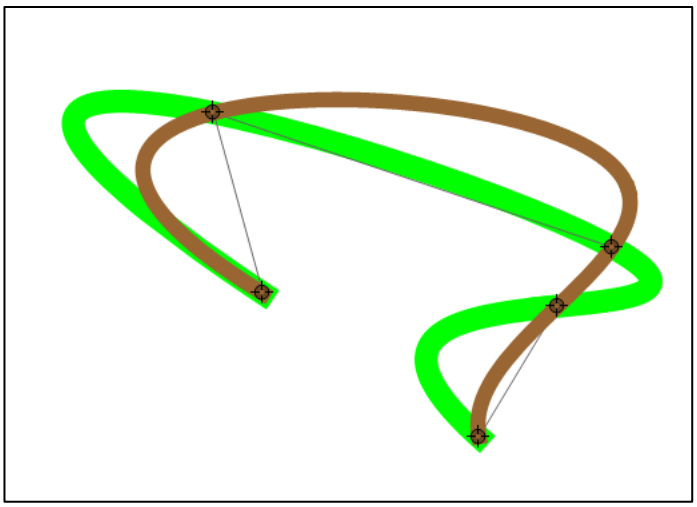


Figure 4.6 Applied Knot Parameter by Chord Length Parameterize (brown) and Normal Weight Distribution of knot (green)

Figure 4.6 shows the different estimated technique to assign knot parameters with the same set of control points and blending function. The brown line is applied by Chord Lengths Parameterize. Its shape is more smooth and realistic when it is compared with the green line. The green curve is assigned with all the weight as equal value. The shape of the green line is slightly hard to control when there is a change of position of control points.

This conversion algorithm can be applied with many techniques to increase capability in the curve design. Figure 4.6 is an example of the integration with the chord length method. An approximation of the arc-length parameterization method can solve the problem about length between each control points and fix the line curve to be smoother.

In conclusion, this chapter provides the details of experiment of interpolation and approximation curve conversion. There are both directions of conversion, interpolation curve to CAGDs curve and backward conversion. The results from the experiments are considered as critical issues including processing time, curve characteristics, and control points. We evaluated the methods by the correctness of output from the conversion. Comparison between the input curve and the output curve guarantees a precise output from this method. Then, regarding an efficacy factor that is influence to the conversion method, the input curve degree was investigated and shown as statistical information. It is possible to find an appropriate degree and request time by using predictable models. At the end of this chapter, some experimental findings that suggest a way to increase the efficiency of the conversion by applying the knot parameter and some techniques which have direct effects on method performances are also discussed.

Commercial test cases are included in Appendix C. They used the real figure from free resources to convert from the interpolation model into the approximation model and vice versa. From the commercial figure result, it shows that real application does not require a high degree of conversion but the curve figure is separated into pieces of curve then converted to other models. It supports the use of the proposed method in commercial software.

In the next chapter, we will sum up our proposed idea and provide suggestions for future work.

CHAPTER 5 CONCLUSION

Our research focuses on the curve conversion by examining the relationship between the interpolation curve and the approximation curve, developing the curve conversion scheme, and conducting the experiments to validate our proposed solution. In addition, we also analyzed the curve properties and demonstrated how the converted curves could be applied in CAGD applications. Both types of curves were applied in different usability. According to our implementation, the curve conversion cannot be done only by basis functions whereas the monomial matrix is appropriate to transform the model to another curve model.

5.1 Conclusion

Basically, this thesis is an extended monomial form method in curve applications. The monomial form is used to convert CAGD curves but there had been no efficient methods to convert the curves into the interpolation model. To improve the monomial form to efficiently support more applications, Chebyshev polynomial was considered the sound solution to be used to represent the relationship between the interpolation curve and the approximation curve (CAGD curve).

According to the experiments in the previous chapter, they revealed output curve characteristics from the conversion algorithms. The results were the correct shape and the characters of input curves. It is clear that there was no false output from the conversion process or the rounding of calculation values.
Conversion time was a factor also measured in the experiments. Conversion time ranking and limitations of conversion methods were also taken into account. The complex shape and high degree of input curves were an important factor to determine the conversion effectiveness. The proposed conversion scheme was appropriate for curves lower than 10 degrees. However, if the input curve were a high degree curve, the curve would be required to be separated into several pieces before being converted with our conversion model.

In addition, our conversion model can be applied to the conversion between interpolation control point curves and all CAGD curves. This conforms to the concept of universal conversion method. The method is possibly applicable to real-world software. The universal conversion method can satisfy the requirements of programmers in CAGD software development. The incompleteness of several conversion methods in CAGD is still a common problem as it cannot support every model to represent conversion applications. Thus, the monomial form integrated with this interpolation and approximation curve conversion is usable for both designer and programmer. They can use the monomial form conversion with any work such as digital hand writing conversion, model converter for architecture and etc.

5.2 Recommendations & Suggestions for Future Development

The improvement of this conversion method was discussed in the previous chapter. We again summarize the key improvements as follows.

.

5.2.1 Applying Knot Parameter and Weight of Control Points

To increase the flexibility of curve fitting, the knot parameter or weight of control points should be included. In fitting the interpolation control point curve, the blending function represents the curve figure. Fixed values of knot parameter in blending function are not appropriate for all interpolation curve designs. The dynamic knot which is applied with any approximation technique allows the curve to be captured effectively. This dynamic knot is a property in Chebyshev polynomial. Therefore, it was employed to be the main mechanism for our conversion method. This knot or weight value can be applied to enhance any approximation methods for or instance, Chord Length Parameterize as mentioned in the example. An approximation of control point positions will take the average of the weight of knot based on the geometry on the graphic screen.

5.2.2 Improvement by Programming Technique

The previous issue is the improvement by changing the parameter in the blending function, which is the inner improvement. Next is an improvement technique by programming. File read-write is a coding strategy to decrease the processing time. There are several calculations that consume the resource in processing. They can be grouped and to be run only once by writing the result of the process as files. To process in the subsequent time, the software application can read the results in the file to process continuously. The example illustrated in this thesis is the generation of coefficient matrix for each model.

5.3 Future Work

For future work, three major issues are going to be worked on more. First, more experiments are required with several modeling software in order to evaluate the generalization and applicability of our conversion model. For example, the interpolation and approximation conversion can increase the domain in the model conversion and apply with any software for touch screen devices.

Second, even though we adopted the dynamic knot in interpolation curve fitting, it was not fully generated. For the future work, we will perform an in-depth investigation on the calculation of knot parameters to dynamically and automatically determine the appropriate weight of knot. The result of curve will be more flexible to fit the designer’s idea. Finally, it is a real challenge to apply our model in representing hand writing to any curve forms.

REFERENCES

1. Farin, G., 2002, **Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide, Academic Press**, Morgan Kaufman Publishers, San Francisco, 5th edition.

2. Said, H. B., 1989, "Generalized Ball Curve and its Recursive Algorithm", **ACM Transactions on Graphics**, Vol. 8, pp. 360–371.

3. Wang, G.Z., Hu, S.M. and Jin, T.G.,1996, "Properties of Two Types of Generalized Ball Curves", **Computer-Aided Design**, Vol. 28, No. 2, pp. 125–133.

4. Wang, G. J., 1987, "Ball Curve of High Degree and its Geometric Properties", **Appl. Math.: A Journal of Chinese Universities**, Vol. 2, pp. 126–140.

5. Dejdumrong, N., 2008, "Efficient Algorithms for Non-Rational and Rational Bézier Curves", **The 5th International Conference Computer Graphics, Imaging and Visualization (CGIV'08)**, August 26- 28, Penang, Malaysia, pp.109-114.

6. Delgado,J. and Pe˜na, J. M., 2003, "**A Shape Preserving Representation with an Evaluation Algorithm of Linear Complexity**", Computer Aided Geometric Design, Vol. 20, No.1, pp. 1–20, [March 2003].

7. Hongyi, Wu., 2000 ,"Unifying Representation of BézierCurve And Genaralized Ball Curves", **Appl. Math. J. Chinese Univ. Ser. B**, Vol. 5, No. 1, pp. 109–121.

8. Xinmeng, Chen. and Dan, Yu., 2007 ,"Another Type Of Generalized Ball Curves And Surfaces",**Acta Mathematica Scientia**, Vol. 27B, No. 4, pp. 897-907.

9. Nunthawisuttiwong,T. and Dejdumrong,N.,2012, "Approximating Online Handwritten Image by BézierCurve", **The 9th International Conference Computer Graphics, Imaging and Visualization (CGIV'12)**, July 25- 27, Hsinchu, Taiwan, pp.17-22.

10. Ramshaw, L., 1989, "Blossom Are Polar Forms", **CAGD**, Vol. 6, pp. 323–358.

11. Tang van,To., 1992, **Polar Form Approach to Geometric Modeling, Ph.D. thesis**, Asian Institute of Technology, Bangkok, Thailand.

12. Aphirukmatakun, C. and Dejdumrong, N., 2009, "Monomial Forms for Curves in CAGD with Their Applications", **The 6th International Conference Computer Graphics, Imaging and Visualization (CGIV'09)**, August 11- 14, Tianjin University, Tianjin, China, pp. 211–216.

13. Itsariyawanich,K. and Dejdumrong,N.,2010, "Erratum to: "Conversion and Evaluation for Two Types of Parametric Surface Constructed by NTP Bases", **Computers and Mathematics with Applications**, Vol. 59, No.1, pp. 595–601.

14. Snyder, M., **Chebyshev Methods in Numerical Approximation**, Prentice-Hall Series in Automatic Computation, Englewood Cliffs, Prentice-Hall [1966].

15. Rababah,A.,2003,"Tranformation of Chebyshev–Bernstein Polynomial Basis" , **Computational Methods in Applied Mathematics**, Vol.3, No.4, pp.608–622

16. Jaroensawad,R., and and Dejdumrong,N.,2013, "Conversion from an Arbitrary Curve into a Bézier Curve with the Least Degree Using Constrained Economization of Chebyshev Polynomials", **10th International Joint Conference on Computer Science and Software Engineering (JCSSE),** 29-31 May, Maha Sarakham,Thailand. pp. 229 - 234

17. Savetseranee, D. and Dejdumrong, N., 2013, "New Monomial Forms Approach for DP and NB1 Curves with Their Proofs**", The 10th International Conference Computer Graphics, Imaging and Visualization (CGIV'13) Computer Graphics**, pp. 42 - 45

18. Savetseranee, D. and Dejdumrong,N., 2013, "Monomial Forms of Two Generalized Ball Curves and their Proofs", **Computer Science and Software Engineering (JCSSE), 2013 10th International Joint Conference**, pp. 235- 239

APPENDIX A
Conversion Program by Mathematica Software

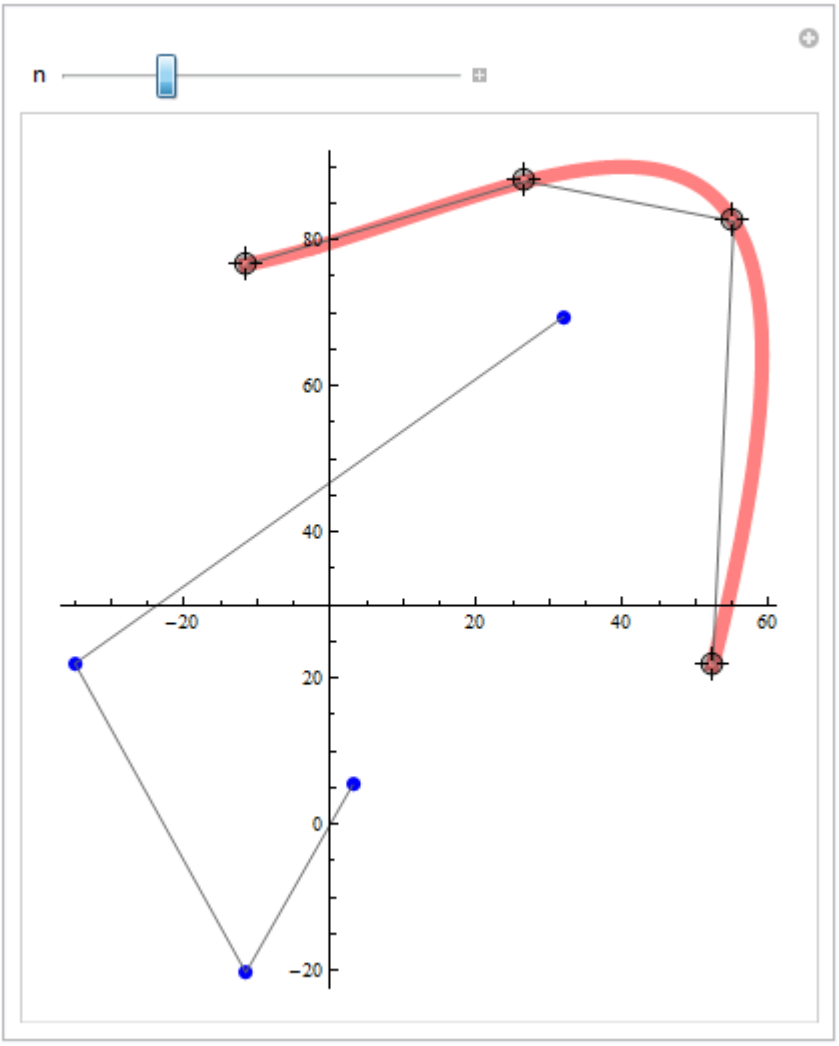
Program1: The Relationship between Curve Shape and Chebyshev Control Point

```
uKnot[cp_] := Table[i / (Length[cp] - 1), {i, 0, Length[cp] - 1}]
MChebyshev = ReadList["/Users/disave/Desktop/MonoDB/ChebyShev.txt"];
ChebyT = ReadList["/Users/disave/Desktop/MonoDB/ChebyShevT.txt"];

ChebyshevCP[pt_] := Module[
   {n, cp},
   n = Length[pt];
   cp = Transpose[Transpose[pt].GenT[n]];
   Return[N[cp]];
  ];
T[n_] := Table[t^i, {i, 0, n}];

genCP[num_, min_, max_] := Module[{cp, i, dist, chMax, chMin}, cp = {};
   dist = (max - min) / num;
   For[i = 0, i < num, i++,
    (*cp=
      Join[cp,{{RandomInteger[{dist*(i-1),dist*i}],
         RandomInteger[{dist*(i-1),dist*i}]}}];*)
    cp =
      Join[cp, {{RandomInteger[{min, max}],
         RandomInteger[{min, max}]}}];];
   Return[cp];];

Manipulate[
 pts = PadRight[pts, n, RandomReal[{0, 100}, {5, 2}]];
 plot2 = ListPlot[pts, Joined → True,
   PlotStyle → {PointSize[0.02], GrayLevel[.4]}];
 plot3 = ParametricPlot[MChebyshev[[Length[pts] - 1]].
    T[Length[pts] - 1].
    Transpose[Transpose[pts].ChebyT[[Length[pts]]]], {t, 0, 1},
   PlotStyle → {Pink, Thickness[0.02]}];
```

```
plot4 = ListPlot[Transpose[Transpose[pts].ChebyT[[Length[pts]]]],
  Joined → True, PlotStyle → {PointSize[0.02], GrayLevel[.4]}];
plot5 := ListPlot[Transpose[Transpose[pts].ChebyT[[Length[pts]]]],
  PlotStyle → {PointSize[0.02], Blue}];
Show[plot3, plot2, plot5, plot4, Axes → True, PlotRange → All]
, {n, 2, 10, 1}, {{pts, {{0, 0}}}, Locator}]
```

Results1 :

Out[100]=



Program2: Interpolation & Approximation Curve Conversion Program

In[108]:=
```
MBezier = ReadList["/Users/disave/Desktop/MonoDB/Bezier.txt"];
MWang = ReadList["/Users/disave/Desktop/MonoDB/Wang.txt"];
MSaid = ReadList["/Users/disave/Desktop/MonoDB/Said.txt"];
MDP = ReadList["/Users/disave/Desktop/MonoDB/DP.txt"];
MDejdumrong = ReadList["/Users/disave/Desktop/MonoDB/Dejdumrong.txt"];
MNB1 = ReadList["/Users/disave/Desktop/MonoDB/NB.txt"];
MChebyshev = ReadList["/Users/disave/Desktop/MonoDB/ChebyShev.txt"];
```

In[115]:=
```
(* Monomial Bezeir Form*)
monoBezier[n_, i_, j_] := (-1)^(j - i) * Binomial[n, j] * Binomial[j, i];
monoMatrixBezier[n_] := Table[monoBezier[n, i, j], {i, 0, n}, {j, 0, n}]
(*------------------------------------------------------------*)
(* Monomial Wang-Ball Form*)
monoWang[n_, i_, j_] := (-1)^(i + j) 2^i Binomial[i + 2, j - i] /;
  0 ≤ i && i ≤ Floor[n / 2] - 1
monoWang[n_, i_, j_] := (-1)^(i + j) 2^Floor[n/2] Binomial[Ceiling[n/2], j - Floor[n/2]] /;
  i == Floor[n / 2]
monoWang[n_, i_, j_] := (-1)^(i + j) 2^Floor[n/2] Binomial[Floor[n/2], j - Ceiling[n/2]] /;
  i == Ceiling[n / 2]
monoWang[n_, i_, j_] := (-1)^(n - i + j) 2^(n-i) Binomial[n - i, j - ((n - i) + 2)] /;
  Ceiling[n / 2] + 1 ≤ i && i ≤ n
monoMatrixWang[n_] := Table[monoWang[n, i, j], {i, 0, n}, {j, 0, n}]
(*------------------------------------------------------------*)
```

```
(* Monomial Said Form*)
monoSaid[n_, i_, j_] :=
 (-1)^(i + j) Binomial[i + Floor[n/2], i] * Binomial[Floor[(n)/2] + 1, j - i] /;
  0 ≤ i ≤ Ceiling[n/2] - 1
monoSaid[n_, i_, j_] := (-1)^(i + j) Binomial[n, n/2] Binomial[n/2, j - i] /;
  i == n/2 && EvenQ[n]
monoSaid[n_, i_, j_] :=
 (-1)^(Floor[n/2] + j + 1) Binomial[-i + n + Floor[n/2], -i + n]
   Binomial[n - i, j - (Floor[n/2] + 1)] /; Floor[n/2] + 1 ≤ i && i ≤ n
monoMatrixSaid[n_] := Table[monoSaid[n, i, j], {i, 0, n}, {j, 0, n}]
(*-------------------------------------------------------------*)
(* Monomial DP Form*)
monoDP[n_, i_, j_] := (-1)^(i + j) Binomial[n, j] /; i == 0
monoDP[n_, i_, j_] := (-1)^(j + 1) Binomial[n - i, j - 1] /; 0 < i <= Floor[n/2] - 1
monoDP[n_, i_, j_] :=
 (Ceiling[n/2] - Floor[n/2]) ((-1)^(j + 1)) Binomial[(Floor[n/2] + 1), j - 1] +
   (1/2)^(Ceiling[n/2] - Floor[n/2])
    (Binomial[0, j] - Binomial[0, j - (Floor[n/2] + 1)] -
      ((-1)^j) Binomial[(Floor[n/2] + 1), j]) /; i == Floor[n/2]
monoDP[n_, i_, j_] :=
 (Ceiling[n/2] - Floor[n/2]) ((-1)^(j + (Floor[n/2] + 1)))
    Binomial[1, j - (Floor[n/2] + 1)] +
   (1/2)^(Ceiling[n/2] - Floor[n/2])
    (Binomial[0, j] - Binomial[0, j - (Floor[n/2] + 1)] -
      ((-1)^j) Binomial[(Floor[n/2] + 1), j]) /; i == Ceiling[n/2]
monoDP[n_, i_, j_] := (-1)^(i + j) Binomial[1, j - i]   /;
  Ceiling[n/2] + 1 <= i && i ≤ n - 1
monoDP[n_, i_, j_] := (-1)^(i + j) Binomial[0, j - i]     /; i == n
monoMatrixDP[n_] := Table[monoDP[n, i, j], {i, 0, n}, {j, 0, n}]
(*-------------------------------------------------------------*)
```

```
(* Monomial Dejdumrong Form*)
monoDejdumrong[n_, i_, j_] :=
 (3^i) * ((-1) ^ (i + j)) *Binomial[i + 3, j - i]  /; 0 ≤ i < Ceiling[n / 2] - 1
monoDejdumrong[n_, i_, j_] :=
 (3^i) * ((-1) ^ (i + j)) *Binomial[n - i, j - i] /; i == Ceiling[n / 2] - 1
monoDejdumrong[n_, i_, j_] :=
 (2 * (3^ (i - 1))) * ((-1) ^ (i + j)) *Binomial[i, j - i] /; (i == (n / 2)) && EvenQ[n]
monoDejdumrong[n_, i_, j_] :=
 (3^ (n - i)) * ((-1) ^ (i + j)) *Binomial[n - i, j - i] /; i == Floor[n / 2] + 1
monoDejdumrong[n_, i_, j_] :=
 (3^ (n - i)) * ((-1) ^ (j + 1 + (n - i))) *Binomial[(n - i), j - (n - i) - 3]  /;
  Floor[n / 2] + 1 < i ≤ n
monoMatrixDejdumrong[n_] := Table[monoDejdumrong[n, i, j], {i, 0, n}, {j, 0, n}]
(*-------------------------------------------------------------*)

(*Monomial NB1 Form*)
monoNB1[n_, i_, j_] :=
 (-1) ^ (i + j) *Binomial[-1 + i + Floor[n/2], i] *Binomial[Floor[n/2], j - i] /;
  0 ≤ i ≤ Floor[n / 2] - 2
monoNB1[n_, i_, j_] :=
 (-1) ^ (i + j) *Binomial[2 *Floor[n / 2] - 2, Floor[n / 2] - 1] *
   Binomial[Floor[n / 2] + 1, j - (Floor[n / 2] - 1)] /; i == Floor[n / 2] - 1
monoNB1[n_, i_, j_] :=
 (-1) ^ (i + j) 2 *Binomial[2 *Floor[n / 2] - 2, Floor[n / 2] - 1] *
   Binomial[Ceiling[n / 2], j - Floor[n / 2]] /; i == Floor[n / 2]

monoNB1[n_, i_, j_] :=
 (-1) ^ (j + Floor[n / 2] + 1) *2^ ((n - i) - (Floor[n / 2] - 1)) *
   Binomial[2 *Floor[n / 2] - 2, Floor[n / 2] - 1] *
   Binomial[n - i, j - (Floor[n / 2] + 1)] /;
  (Floor[n / 2] + 1) <= i <= (Ceiling[n / 2] + 1)
monoNB1[n_, i_, j_] :=
 (-1) ^ (j + Floor[n / 2]) *Binomial[-1 + n - i + Floor[n/2], n - i] *
   Binomial[n - i, j - Floor[n/2]] /; Ceiling[n / 2] + 1 < i ≤ n
monoMatrixNB1[n_] := Table[monoNB1[n, i, j], {i, 0, n}, {j, 0, n}]
(*-------------------------------------------------------------*)
```

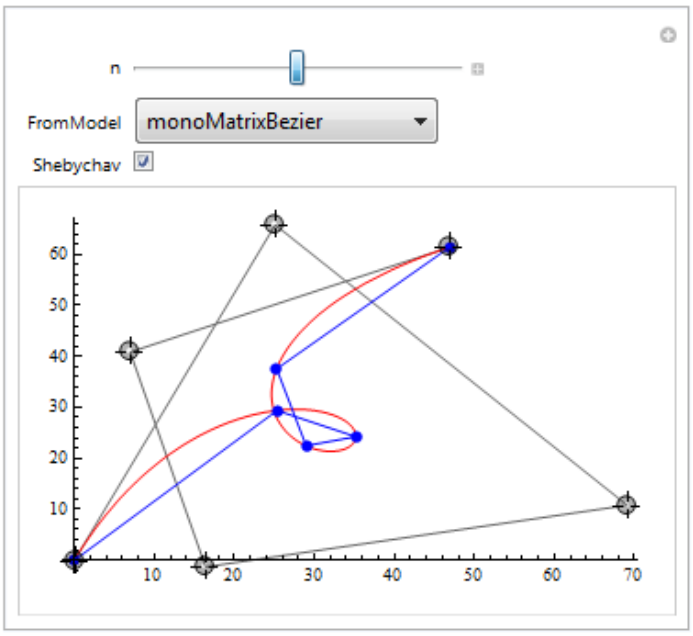
```
ChevbyP[l_, n_] := {1} /; l == 0;
ChevbyP[l_, n_] :=
  Table[((-1)^m) * (2^((2 * l) - (2 * m) - 1)) * ((2 * l) / ((2 * l) - m)) *
    Binomial[(2 * l) - m, m], {m, l, 0, -1}];
monoMatrixShebyChav[n_] := Table[PadRight[ChevbyP[i, n], n + 1], {i, 0, n}];
ShebychavCP[PointOnline_] :=
  Module[{n, i, k, j, Tn, T, p, dstar, CP, d, RecursiveT},
   RecursiveT[n_, p_] := p / p /; n == 1;
   RecursiveT[n_, p_] := (2 * p) - 1 /; n == 2;
   RecursiveT[n_, p_] :=
    Expand[((2 * ((2 * p) - 1) * RecursiveT[n - 1, p]) - RecursiveT[n - 2, p])];
   n = Length[PointOnline];
   Tn[pt_] := Table[RecursiveT[i, p], {i, 1, Length[pt]}];
   Controlp[pt_] := Transpose[pt].Inverse[T];
   T = Tn[PointOnline];
   p = N[Table[j / (n - 1), {j, 0, n - 1}]];
   p = N[Round[p * 100] / 100];

   T[[1]] := Table[1, {j, 1, Length[PointOnline]}];
   CP := Controlp[PointOnline];
   i = 0;
   n = Length[PointOnline];
   dstar := Table[CP[[k]][[j]], {k, 1, 2}, {j, 1, n - i}];
   d := Transpose[dstar];
   Return[d];
  ];
```

```
InterpolationPoint[dstar_] := Module[{RecursiveT, Tn, T, n, p, CP},
   RecursiveT[n_, p_] := p / p /; n == 1;
   RecursiveT[n_, p_] := (2 * p) - 1 /; n == 2;
   RecursiveT[n_, p_] :=
    Expand[((2 * ((2 * p) - 1) * RecursiveT[n - 1, p]) - RecursiveT[n - 2, p])];
   Tn[pt_] := Table[RecursiveT[i, p], {i, 1, Length[pt]}];
   Controlp[pt_] := Transpose[pt].T;
   T = Tn[dstar];
   n = Length[dstar];
   p = N[Table[j / (n - 1), {j, 0, n - 1}]];
   p = N[Round[p * 100] / 100];

   T[[1]] := Table[1, {j, 1, Length[dstar]}];
   CP := Controlp[dstar];
   Return[Transpose[CP]];
  ];

(* Monomial Curve Construction*)
M[n_] := Table[PadRight[ChevbyP[i, n], n + 1], {i, 0, n}];
T[n_] := Table[t^i, {i, 0, n}];


genCP[num_, min_, max_] := Module[{cp, i, dist, chMax, chMin}, cp = {};
   dist = (max - min) / num;
   For[i = 0, i < num, i++,
    cp = Join[cp, {{RandomInteger[{min, max}], RandomInteger[{min, max}]}}];];
   Return[cp];];
MonoBZ[cp_] := Monomial[[Length[cp] - 1]].T[Length[cp] - 1].cp;
```

```
Manipulate[

 pts = PadRight[pts, n, RandomReal[{0, 100}, {5, 2}]];
 Origi[cp_] := FromModel[Length[cp] - 1].T[Length[cp] - 1].cp;
 conV[p1_] :=
  Transpose[FromModel[Length[p1] - 1].
     Inverse[monoMatrixShebyChav[Length[p1] - 1]]].p1;
 FCP = ListPlot[pts, Joined → True,
   PlotStyle → {PointSize[0.02], GrayLevel[.4]}];
 plotF = ParametricPlot[Origi[pts], {t, 0, 1}, PlotStyle → {Red},
   DisplayFunction → Identity, PlotRange → 200];
 TCP = ListPlot[InterpolationPoint[conV[pts]], Joined → {True, False},
   PlotStyle → {Blue, PointSize[0.03]}];
 plotT := ListPlot[InterpolationPoint[conV[pts]],
   PlotStyle → {PointSize[0.02], Blue}, Axes → {True, True}];

 Show[If[Shebychav, {FCP, plotF, TCP, plotT}, {FCP, plotF}], Axes → True,
  PlotRange → All]
 , {n, 2, 10, 1}, {{pts, {{0, 0}}}, Locator},
 {FromModel, {monoMatrixBezier, monoMatrixWang, monoMatrixSaid,
   monoMatrixDP, monoMatrixDejdumrong, monoMatrixNB1}, ControlType → PopupMenu}

 , {Shebychav, {True, False}}
]
```

Result2:

Out[154]=

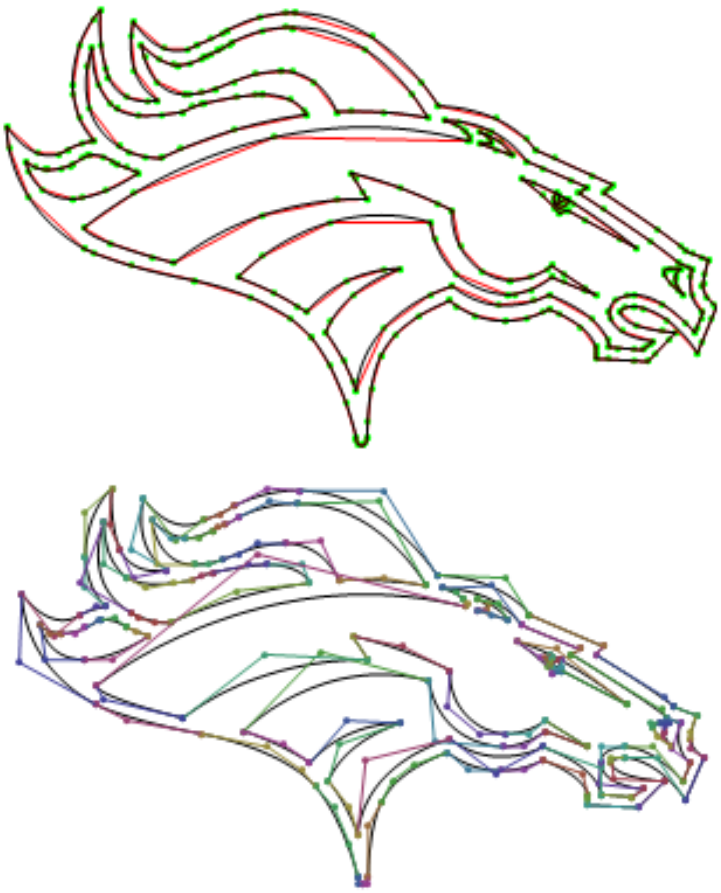


Program3: Commercial Figure Conversion Program

```
data := Import["/Users/disave/Desktop/EPS/L2.eps", "Table"]
Bernstien[n_, i_, t_] := Binomial[n, i] * (t^i) * (1 - t) ^ (n - i)
BezierII[p_List] := Sum[p[[i]] * Bernstien[Length[p] - 1, i - 1, t], {i, 1, Length[p]}]
Bezier[V_, t_] := Module[
   {i, j, Q, Vtemp, degree},
   degree = Length[V] - 1;|
   Vtemp = V;
   For[i = 1, i ≤ degree, i++,
    For[j = 1, j ≤ degree - i + 1, j++,
      Vtemp[[j]] = (1 - t) * Vtemp[[j]] + t * Vtemp[[j + 1]];
     ];
   ];
   Q = Vtemp[[1]];
   Return[Q];
  ];

ManData[d_] := Module[
   {n, i, plot, curve, last, m, setPt, flag, curveSet},
   n = Length[d];
   plot = {};
   curveSet = {};
   flag = 0;
   For[i = 1, i ≤ n, i++,
    m = Length[d[[i]]];
    If[d[[i, m]] == "m",
     setPt = {{d[[i, 1]], d[[i, 2]]}};
     last = setPt;
     flag = 1;
    ];
    If[d[[i, m]] == "L" || d[[i, m]] == "l",
     setPt = {last, last, {d[[i, 1]], d[[i, 2]]}, {d[[i, 1]], d[[i, 2]]}};
     flag = 1;
    ];
    If[d[[i, m]] == "c" || d[[i, m]] == "C",
     setPt = {last, {d[[i, 1]], d[[i, 2]]}, {d[[i, 3]], d[[i, 4]]}, {d[[i, 5]], d[[i, 6]]}};
     flag = 1;
    ];
    If[flag == 1 && NumberQ[setPt[[1, 1]]],
     last = setPt[[Length[setPt]]];
     If[Length[setPt] > 1,
      curveSet = Join[curveSet, {setPt}];
     ];
    ];
    flag = 0;
   ];
   Return[curveSet];
  ];
```

```
DrawVectorInter[curveSet_] := Module[
   {n, i, curve, plot},
   plot = {};
   n = Length[curveSet];
   For[i = 1, i ≤ n, i++,
    curve = ParametricPlot[Inter2[Bezeir2I[curveSet[[i]]]], {t, 0, 1}, PlotStyle → {Black}];
    If[Length[plot] == 0,
     plot = {curve},
     plot = Join[plot, {curve}];
    ];
   ];
   Return[plot];
  ];
LCPVectorInter[curveSet_] := Module[
   {n, i, curve, plot},
   plot = {};
   n = Length[curveSet];
   For[i = 1, i ≤ n, i++,
    curveLP = ListPlot[InterpolationPoint[Bezeir2I[Cat[[i]]]], Joined → {True, False},
      PlotStyle → {Red, PointSize[0.05]}]; If[Length[plot] == 0,
     plot = {curveLP},
     plot = Join[plot, {curveLP}];
    ];
   ];
   Return[plot];
  ];
CPVectorInter[curveSet_] := Module[
   {n, i, curve, plot},
   plot = {};
   n = Length[curveSet];
   For[i = 1, i ≤ n, i++,
    curveP = ListPlot[InterpolationPoint[Bezeir2I[Cat[[i]]]], PlotStyle → {PointSize[0.01],
        Green}, Axes → False]; If[Length[plot] == 0,
     plot = {curveP},
     plot = Join[plot, {curveP}];
    ];
   ];
   Return[plot];
  ];

fig = Round[ManData[data]];
plot1 := CPVectorInter[fig]
plot2 := LCPVectorInter[fig]
plot4 := DrawVectorInter[fig]
plot3 := DrawVector[fig]
plot5 := ListPlot[fig, PlotStyle → {PointSize[0.01]}]
plot6 := ListPlot[fig, Joined → True]
Show[plot1, plot2, plot4, PlotRange → All, Axes → False]
```

Result3:



APPENDIX B

Related Papers

2014 11th International Conference on Computer Graphics, Imaging and Visualization

RELATIONSHIP OF INTERPOLATION & APPROXIMATION CURVE CONVERSION BASED ON MONOMIAL FORM

Dilokvith Savetseranee
Department of Computer Engineering
King Mongkut's University of Technology Thonburi
Bangkok,Thailand
Email:ds@kmutt.net

Natasha Dejdumrong
Department of Computer Engineering
King Mongkut's University of Technology Thonburi
Bangkok,Thailand
Email:natasha@cpe.kmutt.ac.th

***Abstract*—In Computer Aided Geometric Design (CAGD), the utilization of using curves for geometric design and modeling is undertaken by most research works. Due to the high degree of shape preservation of the approximation curve and ease of configuration of the interpolation curve, they are commonly used in CAD and CAM application. However, there are very few works that pay attention to the integration and transformation of those types of curve in graphical modeling tools. In 2010, Aphirukmatakun et al [1] proposed the transformation scheme based on Monomial Form technique to transform the CAGD curve into Newton-Lagrange curve and vice versa. However, Newton-Lagrange has a limitation in dealing with loops curve and zigzag. In this paper, we propose an interpolation and approximation curve conversion scheme to provide more flexible and efficient conversion between them. To this end, we combine Chebyshave polynomial representation and monomial matrix conversion to develop a set of conversion algorithms. The paper presents how the curves are converted and it is considered to be feasible and sound to implement in CAGD application.**

***Keywords*-Monomial form; Monomial Matrix; Curve Conversion;**

I. Introduction

Curve can be separated based on their construction and characteristics into two major groups namely, approximation curve and interpolation curve. Approximation curves are composed of control points in the curve line. These control points are used to set the direction and to generate their blending function to construct a curve. The examples of these models are Bézier Curve, Wang-Ball Curve, Said Ball Curve, etc. This kind of curve can preserve the shape of curve efficiency properties and popular in CAGD field. For the Interpolation curve, it is a curve that line must pass along all control points. In many applications, interpolation curves are typically used to sketch and design process because of its control points that are on the curve is easy to configure the shape.

Besides, interpolation curve is attractive in utilities and complex applications. However, the complex blending functions and complicated method require much more computation time and resource consumption. Thus, both curve forms have their own strength and weakness.

In this research, researchers aimed to improve geometric modeling in CAGD application. Due to high degree of available complex shape of curve along interpolated curve scheme, it is integrated with approximated scheme to be a foundation approach for CAGD application. Moreover, the relevance of approximation and interpolation scheme can be examined in this proposed work.

In 2010, Aphirukmatakun and Dejdumrong [1] presented the solution of various designed methodologies and algorithms. This technique is called Monomial Form. Besides the CAGD curve, monomial form approach can be applied in interpolated curve especially in Newton and Lagrange polynomials.

Additionally, conversion between interpolation scheme and approximation scheme is limited due to the shape preserving problem. This problem occurs when high degree of curve is constructed. In high degree, the curve is wiggle at both ends of curve. This problem is called ill-condition problem. In this research, ill-condition problem can be solved by monomial form.

II. Related theory

A. Approximation curve

Approximation curve is a curve that all control points does not need to locate on the created curve. This kind of curve is widely adopted because it has a high degree of shaper preservation as well as ease of use.

1) Bézier: Bézier curve can be constructed by using either de Casteljau algorithm in 1957 or a model base on Bernstein polynomials provide by P.Bzier in 1962 [3]. Bézieris popular because there is clear Bernstein polynomial form and the influence of control point is equal every points. Although there are some research presented the better algorithm to draw curve. Its Said-Ball algorithm.

2) Said-Ball Curve: In 1974, Ball defined a set of basis function for cubic curve and bicubic surfaces. Said [4] investigated Said-Ball curves by generalizing the Ball model to higher degree. After that Hu et. al.(1996) developed the recursive algorithm for Said-Ball curves. It makes the simplicity for Said-Ball curve construction. Unfortunately,

978-1-4799-5720-0/14 $31.00 © 2014 IEEE
DOI 10.1109/CGiV.2014.12

Said-Ball is more complex to use when comparing with Wang-Ball algorithm which were publicized in same paper by Hu et. al.

3) Wang-Ball Curve: Wang-Ball curve were investigated by Wang [5] in 1987 but publicized later in 1996 by Hu et.al [6] with its degree elevation and degree reduction. A Wang-Ball curve can be computed more efficient than Bzier and Said-Ball curve because it requires linear time complexity (n).

4) DP Curves: In 2003, Delgado and Pena [11] developed DP curves which represent a curve with linear computational complexity. They proved that blending basis of DP curves is a normalized totally positive basis which claims that DP curves have shape preserving property.

5) NB1 Curves: NB1 and NB2 were first investigated by Wu [12] in 2000. Later, in 2007, Yu and Chen [13] introduced the basis functions and their properties of NB1 curves (NB2 curve in Wu).

6) Dejdumrong Curves: In 2008, a new algorithm of plotting curve was presented by Dejdumrong[10] . The curve recursive algorithm is a linear computation form similar to Wang-Ball and DP curves. Dejdumrong curve calculation was faster than the other curves except for Wang-Ball and DP curves. However, the shape of Dejdumrong curve was closest to the Bzier curve. Furthermore, Dejdumrong curve can be plotted faster than Bzier curve.

B. Interpolation curve

Newton and Lagrange polynomial is typically used to represent an interpolation curve known as Newton-Lagrange curve. In [10] T. Nuntawisuttiwong and N. Dejdumrong, apply Newton-Lagrange polynomial to approximate Hand writing and it is converted to Bzier curve. But this technique can be applied for simple curve only because Newton-Lagrange is sensitive for ill-condition and not flexible for complex curves such as loop curve and zigzag curve.

Thus, method which flexible used to apply in monomial form is investigated from the relationship of curves.

C. Monomial Matrices

Aphirukmatakun and Dejdumrong [2] presented the proposition of Monomial form in 2009. This technique showed that polynomial basis can be representative by the group of coefficient matrix-multiplication but they specify the coefficient matrix as the monomial coefficient. This technique derived from the basis function of approximation curve in form of matrix multiplication between control points, coefficient of power basis and parameter t at each degree from 0 to n. Cubic Bézier can represent by (1) and (2):

$$\beta(t) = \mathbf{G} \cdot \mathbf{M} \cdot \mathbf{T} \tag{1}$$

$$\begin{aligned}\beta(t) \quad = \quad & \left[t^2 - 2t + 1\right] \cdot \mathbf{p}_0 + \left[2t^3 - 4t^2 + 2t\right] \cdot \mathbf{p}_1 \\ & + \left[-2t^3 + 2t^2\right] \cdot \mathbf{p}_2 + \left[t^2\right] \cdot \mathbf{p}_3\end{aligned} \tag{2}$$

D. Vandermonde Matrix

It is an alternative approach to interpolate the control points. This technique is not represented the function in term of polynomial. It performs coefficient matrix which is better benefit to represent the curve more than polynomial. The Vandermonde is defined as follows:

$$M^n = \begin{bmatrix} m_{0,0} & m_{0,1} & \dots & m_{0,n} \\ m_{1,0} & m_{1,1} & \dots & m_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,0} & m_{n,1} & \dots & m_{n,n} \end{bmatrix} \tag{3}$$

where

$$m_{i,j} = \begin{cases} 1 & , \quad for\ i = 0 \\ t_j^i & , \quad for\ i > 0 \end{cases} \tag{4}$$

And $\{t_j\}_{j=0}^n$ are a set of knots

E. Chebyshev Polynomial

Chebyshev polynomial is a popular approximate polynomial. There is a perpendicular property which makes equation is stable. Chebyshev polynomial [8] has the period of time between $[-1, 1]$. The simple form of Chebyshev polynomial is the first kind Chebyshev. It is formally defined as follow:

$$P_n(t) = a_0 + a_1 t + a_{2n}(2t - 1) + \ldots + a_n(2tT_{n-1} - T_{n-2}) \tag{5}$$

where

$\phi_i = T_i = 2tT_{i-2}$ for $i > 1$.

and

$T_0 = 1$.

$T_1 = t$.

Chebyshev polynomial is one of the schemes we consider to integrate in our solution. It can be represented in term of power basis (6)

$$T_n(t) = \sum_{i=0}^{\frac{n}{2}} c_i^n t^{n-2i} \tag{6}$$

when

$$C_i^n = (-1)^i 2^{n-2i-1} \cdot \frac{n}{n-i} \begin{pmatrix} n-i \\ i \end{pmatrix} \tag{7}$$

(7) represents an coefficient function for Chebyshev polynomial.

The interpolation and approximation curve which are mention in this part. They are profitably used to apply in several applications. However, the method that used to

convert between both types is inflexible to transform the model. Thus, to study and develop the relationship of this conversion method will increase the utility of monomial form and several applied application in CAGD.

III. Methodology

This session consists of two parts: interpolation curve fitting with ChebyShev polynomial and conversion of Chebyshev monomial form to approximation curve.

A. Chebyshev Curve

Conversion between Interpolation curve and Approximation curve cannot be done by using Chebyshev polynomial directly. This is because the range of Chebyshev polynomial is between $[-1, 1]$ which is not applicable for normal curve construction as the equation (8). it not normal for curve construction. Thus, period shifting is required to $[0, 1]$

$$T_n(t^*) = 2t^* T_{n-1}(t^*) - T_{n-2}(t^*) \tag{8}$$

To shift the period $[-1, 1]$ to $[0, 1]$ by the relation of Chebyshev polynomial [9].

$$t^* = 2t_i - 1, \quad 0 \leq t_i \leq 1 \tag{9}$$

To substitution (9) into (8) can be derive to (10) as follows:

$$T_n^*(t) = 2(2t-1)T_{n-1}(2t-1) - T_{n-2}(2t-1) \tag{10}$$

(10) is recursive function for construct Chebyshev polynomial

There is a set of control point on the interpolation curve. Then, find coefficient of Chebyshev polynomial. It represented the control point of Chebyshev curve.

$$C_y = [T_n^*]^{-1} \cdot y \tag{11}$$

$$C_x = [T_n^*]^{-1} \cdot x \tag{12}$$

Thus,C_x and C_y are represented as the control point of Chebyshev curve.

B. Chebyshev Monomial Form and Conversion

1) Chebyshev Monomial Form: Next, the method used to convert interpolation curve in term of Chebyshev polynomial to approximation Curve is monomial form of Chebyshev.

$$T_n(t) = \sum_{i=0}^{\frac{n}{2}} c_i^n t^{n-2i} \tag{13}$$

(13) is the normal basis of Chebyshev polynomial form without shifting period to$[0, 1]$. Thus, the shifted formula for construction process is(15). It can be derived by substitution (14) into (13) in term of parameter t [8].

Table I
Conversion from Interpolation curve to Approximation curve

Input	Output	Transformation Matrix
Interpolation Curve	Bézier	$\delta = T^* \cdot B^{-1}$
	Said-Ball	$\delta = T^* \cdot S^{-1}$
	Wang-Ball	$\delta = T^* \cdot A^{-1}$
	DP	$\delta = T^* \cdot C^{-1}$
	NB1	$\delta = T^* \cdot N^{-1}$
	Dejdumrong	$\delta = T^* \cdot D^{-1}$

$$T_{2n}(\sqrt{t}) = T_n(t^*) \tag{14}$$

$$T_n(t^*) = T_n^*(t) = \sum_{i=0}^{\frac{n}{2}} c_i^{2n} t^{n-i} \tag{15}$$

when

$$C_i^{2n} = (-1)^i 2^{2n-2i-1} \cdot \frac{2n}{2n-i} \begin{pmatrix} 2n-i \\ i \end{pmatrix} \tag{16}$$

2) Conversion Model: The Conversion concept is to transform Chebyshev into other models by using monomial matrix conversion. Let monomial form of Chebyshev is equal to the monomial form of any approximation model as follow:

$$m \cdot T^* \cdot t = x \cdot Y \cdot t \tag{17}$$

$$m \cdot T^* = x \cdot Y \tag{18}$$

$$m \cdot T^* \cdot Y^{-1} = x \cdot Y \cdot Y^{-1} \tag{19}$$

$$x = m \cdot T^* \cdot Y^{-1} \tag{20}$$

Finally, the new control point of approximation curve which represent by x can calculate as (20).

The Interpolation curve can be computed by using transformation matrix, δ. A set of interpolation control points (Interpolation curve), $\{m_i\}_{i=0}^n$. can be converted to approximation curve, $\{r_i\}_{i=0}^n$ as follows:

$$[r_0 \;\; r_1 \;\; \dots \;\; r_n] = [m_0 \;\; m_1 \;\; \dots \;\; m_n] \cdot \delta \tag{21}$$

where δ is transformation defined in Table 1

The invert transformation to convert approximation curve to interpolation curve is the invert conversion from Table 3.1. Let $\{m_i\}_{i=0}^n$ is the interpolation control point and $\{r_i\}_{i=0}^n$ is approximation control point. Any approximation curve can be converting into interpolation curve as follows:

$$[m_0 \;\; m_1 \;\; \dots \;\; m_n] = [r_0 \;\; r_1 \;\; \dots \;\; r_n] \cdot \delta \tag{22}$$

where δ is transformation defined in Table 2.

Table II
CONVERSION FROM APPROXIMATION CURVE TO INTERPOLATION CURVE

Input	Output	Transformation Matrix
Bézier	Interpolation Curve	$\delta = T^* \cdot B^{-1}$
Said-Ball		$\delta = T^* \cdot S^{-1}$
Wang-Ball		$\delta = T^* \cdot A^{-1}$
DP		$\delta = T^* \cdot C^{-1}$
NB1		$\delta = T^* \cdot N^{-1}$
Dejdumrong		$\delta = T^* \cdot D^{-1}$

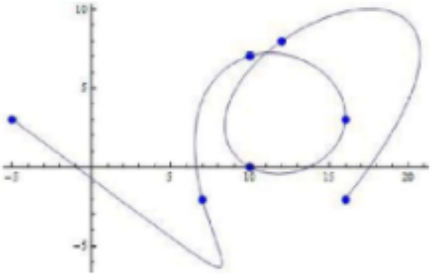


Figure 1. Example1 Chebyshev interpolated curve

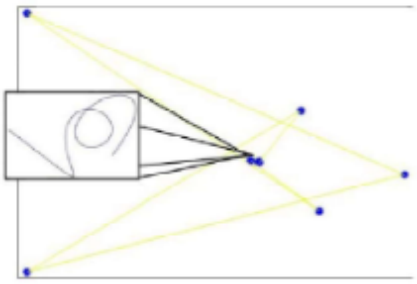


Figure 2. Result from convert into Wang-Ball curve

IV. EMPIRICAL RESULTS

In general, to transform interpolation curve to approximation curve have to represent curve in terms of their basis or monomial form. In this experiment we convert Chebyshev interpolated curve to Wang-Ball by using monomial matrix. The conversion transforms Chebyshev control points to Wang-Ball control point. Figure.1 represents original Chebyshev interpolated curve. Figure.2 is Wang-Ball curve with a new control point from the conversion.

The process of conversion from Chebyshev interpolated curve to Wang-Ball Curve is described as follows:

$$C_{Wang} = \delta \cdot C_{Chebyshev} \tag{23}$$

where

δ is the conversion matrix which is defined as follow:

$$\delta = t^* \cdot A^{-1} \tag{24}$$

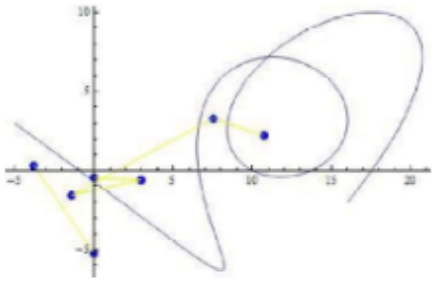


Figure 3. The result of interpolation curve with Chebyshev Control polygon.

C_{Wang} is the set of Wang-Ball control point.
$C_{Chebyshev}$ is Chebyshev control point.

V. CONCLUSION

Our research focuses on the curve conversion by examining the relationship between interpolation curves and approximation curves into terms of curve properties and applications. The recursive form of the polynomial curve can be simply constructed using monomial approach. A monomial form also provides curve conversion using inverse matrix. The degree elevation and degree reduction can also be applied. Functional complexity *O(n)* can also be avoided by converting the curve type into the basis which has lower complexity.

For the future improvements, the analysis of conversion and approximation errors will be investigated.

REFERENCES

[1] C. Aphirukmatakun and N. Dejdumrong, *Relationships between Newton-Lagrange and CAGD Curves using Monomial Form Approach*, in Proc. The 7th International Joint Conference on Computer Science and Software Engineering , ,pp.265-270,2010.

[2] C. Aphirukmatakun and N. Dejdumrong, *Monomial Form for Curves in CAGD with Their Applications*, The 6th International Conference Computer Graphics, Imaging and Visualization(CGIV09), Tianjin University, Tianjin, China August 11-14,pp.211-216,2009.

[3] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design: A practical guide* , Academic Press, Morgan Kaufman Publishers,San Francisco, 5th edition,2002.

[4] H. B. Said, *Generalized Ball Curve and Its Recursive Algorithm*, ACM Transactions on Graphics, vol. 8, pp. 360-371, 1989.

[5] G. J.Wang, *Ball Curve of High Degree and Its Geometric Properties*, Appl. Math.:A Journal of Chinese Universities, vol. 2, pp. 126-140, 1987.

[6] G.Z. Wang S.M. Hu and T.G. Jin, *Properties of Two Types of Generalized Ball Curves*, Computer-Aided Design, vol. 28, no. 2, pp. 125-133, 1996.

[7] N. Dejdumrong, *Efficient Algorithms for Non-Rational and Rational Bzier Curves*, , 2008.

[8] M. Snyder, *Chebyshev Methods in Numerical Approximation, Prentice-Hall Series in Automatic Computation*, Englewood Cliffs, Prentice-Hall, 1966.

[9] J.C. Mason, David C. Handscomb, *Chebyshev Polynomials*, 2003.

[10] T.Nunthawisuttiwong and N. Dejdumrong, *Approximating Online Handwritten Image by Bézier Curve*, The 9th International Conference Computer Graphics, Imaging and Visualization(CGIV12), pp.17-22, 2012.

[11] J. Delgado and J. M. Pena, *A Shape Preserving Representation with an Evaluation Algorithm of Linear Complexity*, Computer Aided Geometric Design, vol. 20, no.1, pp. 1-20, March 2003.

[12] Wu Hongyi, *Unifying Representation of Bézier Curve And Genaralized Ball Curves*, Appl. Math. J. Chinese Univ. Ser. B, vol. 5, no. 1, pp. 109-121, 2000.

[13] Chen Xinmeng Yu Dan, *Another Type Of Generalized Ball Curves And Surfaces*, Acta Mathematica Scientia, vol. 27B, no. 4, pp. 897-907, 2007.

2013 10th International Conference Computer Graphics, Imaging and Visualization

New Monomial Forms approach for DP and NB1 curves with Their Proofs

Dilokvith Savetseranee
Department of Computer Engineering
King Mongkut's University of Technology Thonburi
Bangkok,Thailand
Email:ds@kmutt.net

Natasha Dejdumrong
Department of Computer Engineering
King Mongkut's University of Technology Thonburi
Bangkok,Thailand
Email:natasha@cpe.kmutt.ac.th

***Abstract*—There are several methods used to construct curves in CAGD, e.g., the de Casteljaus algorithm, DP algorithm, and NB1 algorithm. However, they are represented in the forms that are not suitable for performing geometric modeling. A proposed approach for evaluating DP curve, NB1 Curve by using monomial form was introduced by Aphirukmatakun and Dejdumrong in 2009. The authors did not provide any proofs for their propositions. Thus, this paper shows the proofs for the conversions from DP polynomial into its monomial form as well as NB1 polynomial.**

***Keywords*-Monomial form; Monomial Matrix; Curve Conversion;**

I. Introduction

In geometric modeling and design, there are several algorithms used for curve, surface and solid constructions. Mathematical formulae represent geometric objects in the design process and in the industrial procedures.

Unfortunately, there are various methodologies to analyze shape especially in curve design. There are Bézier Bernstein, DP, and NB1 models used for approximating curves. But all of them use to represent the same curve. The different algorithms specify appropriated function in each application. Thus, programmers have to transform the algorithm to be the best solution for processing by using relation of mathematic function between each algorithm. As previously, curve/surface model transformations are complicated and tough to implement. The conversion computations between two curves with different degrees have to be recalculated. This can cause the chance of programming mistakes and waste the time in computational process.

In 2009, Aphirukmatakun and Dejdumrong [1] presented the solution of various designed methodologies and algorithms.This technique is called Monomial Form. They studied the relationships between curve polynomials which are the most popular bases and the monomial forms (or power bases). The matrix form simplifies the curve model. Moreover, it is easy to understand.

However, the authors did not prove their propositions in terms of mathematics so that this paper will introduce the proofs for each proposition in details.

II. Related theory

A. Polynomial Curve

There are many polynomial curves in CAGD, e.g. Bézier, Said-Ball, Wang-Ball, DP and NB1 curves. These polynomial curves can be constructed by using their control points and blending functions. Their control points and blending functions influence to the shape of polynomial curves. In each curve, their characteristic are alike but they are different in their blending functions. The definition of polynomial curves in mathematics can be explained in the following definition.

B. Bézier curve

Bézier curve can be constructed by using either de Casteljau algorithm in 1957 or a model base on Bernstein polynomials provide by P.Bézier in 1962 [2]. The curve is presented by two components Bernstein polynomials and Control Point. Bézier is popular because there is clear Bernstein polynomial form and the influence of control point is equal every points. But the high degree curve is not appropriate to draw.

C. Generalized Ball Curves

In 1974, Ball defined a set of basis function for cubic curve and bicubic surfaces. After that in 1996 Hu et.al [3] developed the recursive algorithm for Said-Ball and Wang-Ball curves.

D. DP Curves

In 2003, Delgado and Pena [4] developed DP curves which represent a curve with linear computational complexity. They proved that blending basis of DP curves is a normalized totally positive basis which claims that DP curves have shape preserving property.

E. NB1 Curves

NB1 and NB2 were first investigated by Wu [5] in 2000. Later, in 2007, Yu and Chen [6] introduced the basis functions and their properties of NB1 curves (NB2 curve in Wu).

Therefore, this paper supports the proposition of monomial form to convert every curve algorithm into monomial

978-0-7695-5051-0/13 $26.00 © 2013 IEEE
DOI 10.1109/CGIV.2013.17

basis. Flexibility, universal and simplification are the key properties of this technique. Programmers can reduce the mistake and less wasted time in computation process. Thus, monomial form is the simple solution and also develops for parallel computation by using Matrix property of Monomial basis.

F. Monomial Matrices

Aphirukmatakun and Dejdumrong presented the proposition of Monomial form in 2009. This technique showed that polynomial basis can be representative by the group of coefficient matrix-multiplication but they specify the coefficient matrix as the monomial coefficient. As shown in below the representation of Bézier basis into Monomial Matrix form.

III. Methodology

1) Proposition1 DP Monomial Form: An nth-degree DP curve, denoted by $C^n(t)$, given by a set of $n+1$ control points, denoted by $\{q_i\}_{i=0}^{n}$, can be formulated in power basis form by

$$C^n(t) = \sum_{i=0}^{n}\sum_{j=0}^{n} \mathbf{q}_i C_{i,j} t^j \tag{1}$$

where

$$C_{ij} = \begin{cases} (-1)^j\binom{n}{j} & , \text{ for } i=0, \\ (-1)^{j-i}\binom{n-i}{j-i} & , \text{ for } 0<i\le\left\lfloor\frac{n}{2}\right\rfloor-1, \\ (-1)^{j-i}(n-2i)\binom{i+1}{j-i} & \\ \quad +\left(\frac{1}{2}\right)^{n-2i}\left(\binom{0}{j}-\binom{0}{j-i-1}-(-1)^j\binom{i+1}{j}\right) & , \text{ for } i=\left\lfloor\frac{n}{2}\right\rfloor, \\ (-1)^{j-n+i}(n-2i)\binom{1}{j-n+i-1} & \\ \quad +\left(\frac{1}{2}\right)^{2i-n}\left(\binom{0}{j}-\binom{0}{j-n+i-1}-(-1)^j\binom{n-i+1}{j}\right) & , \text{ for } \left\lfloor\frac{n}{2}\right\rfloor+1, \\ (-1)^{j-i}\binom{1}{j-i} & , \text{ for } \left\lfloor\frac{n}{2}\right\rfloor+1\le i\le n-1, \\ \binom{0}{j-n} & , \text{ for } i=n. \end{cases} \tag{2}$$

Proof: The DP curve can be represented in the form of DP basis as follows:

$$C_i^n(t) = \begin{cases} (1-t)^n & , \text{ if } i=0, \\ t(1-t)^{n-i} & , \text{ for } 1\le i\le\left\lfloor\frac{n}{2}\right\rfloor-1, \\ \left(\left\lfloor\frac{n}{2}\right\rfloor-\left\lfloor\frac{n}{2}\right\rfloor\right)t(1-t)^{\left\lfloor\frac{n}{2}\right\rfloor+1} & \\ \quad +\left(\frac{1}{2}\right)^{\left\lfloor\frac{n}{2}\right\rfloor-\left\lfloor\frac{n}{2}\right\rfloor}\left(1-t^{\left\lfloor\frac{n}{2}\right\rfloor+1}-(1-t)^{\left\lfloor\frac{n}{2}\right\rfloor+1}\right) & , \text{ for } i=\left\lfloor\frac{n}{2}\right\rfloor, \\ \left(\left\lfloor\frac{n}{2}\right\rfloor-\left\lfloor\frac{n}{2}\right\rfloor\right)(1-t)t^{\left\lfloor\frac{n}{2}\right\rfloor+1} & \\ \quad +\left(\frac{1}{2}\right)^{\left\lfloor\frac{n}{2}\right\rfloor-\left\lfloor\frac{n}{2}\right\rfloor}\left(1-t^{\left\lfloor\frac{n}{2}\right\rfloor+1}-(1-t)^{\left\lfloor\frac{n}{2}\right\rfloor+1}\right) & , \text{ for } i=\left\lfloor\frac{n}{2}\right\rfloor, \\ t^i(1-t) & , \text{ for } \left\lfloor\frac{n}{2}\right\rfloor+1\le i\le n-1, \\ t^n & , \text{ for } i=n. \end{cases} \tag{3}$$

The proof can be done by transforming DP basis into the DP monomial form in each contradiction.

1. If $i=0$

$$\begin{aligned} C_i^n(t) &= (1-t)^n \\ &= \sum_{j=0}^{n}(-1)^j\binom{n}{j}t^j \end{aligned}$$

2. For $1\le i\le\lfloor\frac{n}{2}\rfloor-1$

$$\begin{aligned} C_i^n(t) &= t(1-t)^{n-i} \\ &= t\sum_{j=0}^{n-i}(-1)^j\binom{n-i}{j}t^j \\ &= \sum_{j=i}^{n-i+1}(-1)^{j-1}\binom{n-i}{j-1}t^j \end{aligned}$$

3. If $i=\lfloor\frac{n}{2}\rfloor$

$$\begin{aligned} C_i^n(t) &= \left(\left\lfloor\frac{n}{2}\right\rfloor-\left\lfloor\frac{n}{2}\right\rfloor\right)t(1-t)^{\left\lfloor\frac{n}{2}\right\rfloor+1} \\ &\quad +\left(\frac{1}{2}\right)^{\left\lfloor\frac{n}{2}\right\rfloor-\left\lfloor\frac{n}{2}\right\rfloor}\left(1-t^{\left\lfloor\frac{n}{2}\right\rfloor+1}-(1-t)^{\left\lfloor\frac{n}{2}\right\rfloor+1}\right) \\ &= \left(\left\lfloor\frac{n}{2}\right\rfloor-\left\lfloor\frac{n}{2}\right\rfloor\right)t\sum_{j=0}^{\left\lfloor\frac{n}{2}\right\rfloor+1}(-1)^j\binom{\left\lfloor\frac{n}{2}\right\rfloor+1}{j}t^j \\ &\quad +\left(\frac{1}{2}\right)^{\left\lfloor\frac{n}{2}\right\rfloor-\left\lfloor\frac{n}{2}\right\rfloor}\Bigg(\sum_{j=0}^{0}(-1)^j\binom{0}{j}t^j-t^{\left\lfloor\frac{n}{2}\right\rfloor+1}\sum_{j=0}^{0}(-1)^j\binom{0}{j}t^j \\ &\quad -\sum_{j=0}^{\left\lfloor\frac{n}{2}\right\rfloor+1}(-1)^j\binom{\left\lfloor\frac{n}{2}\right\rfloor+1}{j}t^j\Bigg) \\ &= -\sum_{j=0}^{\left\lfloor\frac{n}{2}\right\rfloor+1}\Bigg\{(-1)^{j-(\lfloor\frac{n}{2}\rfloor+1)}\left(\left\lfloor\frac{n}{2}\right\rfloor-\left\lfloor\frac{n}{2}\right\rfloor\right)\binom{1}{j-(\lfloor\frac{n}{2}\rfloor+1)} \\ &\quad +\left(\frac{1}{2}\right)^{\left\lfloor\frac{n}{2}\right\rfloor-\left\lfloor\frac{n}{2}\right\rfloor}\left[\binom{0}{j}\right. \\ &\quad \left.-\binom{0}{j-(\lfloor\frac{n}{2}\rfloor+1)}-(-1)^j\binom{\lfloor\frac{n}{2}\rfloor+1}{j}\right]\Bigg\}t^j \end{aligned} \tag{6}$$

This equation (6) can be simplified by

$$\begin{aligned} C_i^n(t) = \sum_{j=i}^{n-i+1}\Big[&(-1)^{j-n+i}(n-2i)\binom{1}{j-n+i-1} \\ &+\left(\frac{1}{2}\right)^{n-2i}\Big[\binom{0}{j}-\binom{0}{j-n+i-1} \\ &-(-1)^j\binom{n-i+1}{j}\Big]\Big]t^j \end{aligned} \tag{7}$$

4. For $\lfloor\frac{n}{2}\rfloor+1\le i\le n-1$

$$\begin{aligned} C_i^n(t) &= t^i(1-t) \\ &= t^i\sum_{j=0}^{1}(-1)^j\binom{1}{j}t^j \\ &= \sum_{j=i}^{i+1}(-1)^{j-i}\binom{1}{j-i}t^j \end{aligned}$$

5. For $i=n$

$$C_i^n(t) = t^n = t^n \sum_{j=0}^{n} (-1)^j \binom{0}{j} t^j = \sum_{j=n}^{n} (-1)^{j-n} \binom{0}{j-n} t^j \quad (8)$$

Equation (8) can be simplified as follows

$$C_i^n(t) = \sum_{j=0}^{n} \binom{0}{j-n} t^j \quad (9)$$

2) Proposition 2 NB1 Monomial Form: An NB1 curve of degree n, $N^n(t)$, with $n+1$ control points, denoted by $\{y_i\}_{i=0}^{n}$, can be formed by the power basis form as follows:

$$N^n(t) = \sum_{i=0}^{n} \sum_{j=0}^{n} \mathbf{y}_i \cdot g_{i,j} \cdot t^j,$$

$$g_{i,j} = \begin{cases} (-1)^{j-i} \binom{\lfloor \frac{n}{2} \rfloor - 1 + i}{i} \binom{\lfloor \frac{n}{2} \rfloor}{j-i} & \text{,for } 0 \le i \le \lfloor \frac{n}{2} \rfloor - 2, \\ (-1)^{j-i} \binom{2i}{i} \binom{i+2}{j-i} & \text{,for } i = \lfloor \frac{n}{2} \rfloor - 1, \\ (-1)^{j-i} 2 \binom{2i-2}{i-1} \binom{n-i}{j-i} & \text{,for } i = \lfloor \frac{n}{2} \rfloor, \\ (-1)^{j-i} 2 \binom{2(n-i-1)}{n-i-1} \binom{n-i}{j-i} & \text{,for } i = \lceil \frac{n}{2} \rceil, \\ (-1)^{j-n+i} \binom{2(n-i)}{n-i} \binom{n-i}{j-n+i-2} & \text{,for } i = \lceil \frac{n}{2} \rceil + 1, \\ (-1)^{j-\lfloor \frac{n}{2} \rfloor} \binom{\lfloor \frac{n}{2} \rfloor - 1 + n - i}{n-i} \binom{n-1}{j-\lfloor \frac{n}{2} \rfloor} & \text{,for } \lceil \frac{n}{2} \rceil + 2 \le i \le n. \end{cases} \quad (10)$$

Proof: The NB1 curve can be expressed in the form of the bases as follows:

$$N_i^n(t) = \begin{cases} \binom{\lfloor \frac{n}{2} \rfloor - 1 + i}{i} t^i (1-t)^{\lfloor \frac{n}{2} \rfloor} & \text{, for } 0 \le i \le \lfloor \frac{n}{2} \rfloor - 2, \\ \binom{2\lfloor \frac{n}{2} \rfloor - 2}{\lfloor \frac{n}{2} \rfloor - 1} t^{\lfloor \frac{n}{2} \rfloor - 1} (1-t)^{\lfloor \frac{n}{2} \rfloor + 1} & \text{, if } i = \lfloor \frac{n}{2} \rfloor - 1, \\ 2 \binom{2\lfloor \frac{n}{2} \rfloor - 2}{\lfloor \frac{n}{2} \rfloor - 1} t^{\lfloor \frac{n}{2} \rfloor} (1-t)^{\lfloor \frac{n}{2} \rfloor} & \text{, if } i = \lfloor \frac{n}{2} \rfloor, \\ N_{n-i}^n (1-t) & \text{, for } \lfloor \frac{n}{2} \rfloor + 1 \le i \le n. \end{cases} \quad (11)$$

The proof can be given by transforming NB1 basis into the NB1 monomial form.

1. For $0 \le i \le \lfloor \frac{n}{2} \rfloor - 2$

$$N_i^n(t) = \binom{\lfloor \frac{n}{2} \rfloor - 1 + i}{i} t^i (1-t)^{\lfloor \frac{n}{2} \rfloor} = \sum_{j=i}^{\lfloor \frac{n}{2} \rfloor + i} (-1)^{j-i} \binom{\lfloor \frac{n}{2} \rfloor - 1 + i}{i} \binom{\lfloor \frac{n}{2} \rfloor}{j-i} t^j$$

2. If $i = \lfloor \frac{n}{2} \rfloor - 1$

$$N_i^n(t) = \binom{2\lfloor \frac{n}{2} \rfloor - 2}{\lfloor \frac{n}{2} \rfloor - 1} t^{\lfloor \frac{n}{2} \rfloor - 1} (1-t)^{\lfloor \frac{n}{2} \rfloor + 1} = \binom{2\lfloor \frac{n}{2} \rfloor - 2}{\lfloor \frac{n}{2} \rfloor - 1} t^{\lfloor \frac{n}{2} \rfloor - 1} \sum_{j=0}^{\lfloor \frac{n}{2} \rfloor + 1} (-1)^j \binom{\lfloor \frac{n}{2} \rfloor + 1}{j} t^j = \sum_{j=\lfloor \frac{n}{2} \rfloor - 1}^{2\lfloor \frac{n}{2} \rfloor} (-1)^{j-(\lfloor \frac{n}{2} \rfloor - 1)} \binom{2\lfloor \frac{n}{2} \rfloor - 2}{\lfloor \frac{n}{2} \rfloor - 1} \binom{\lfloor \frac{n}{2} \rfloor + 1}{j-(\lfloor \frac{n}{2} \rfloor - 1)} t^j$$

It can be simplified as follows

$$N_i^n(t) = \sum_{j=i}^{n} (-1)^{j-i} 2^j \binom{n-i}{j-i} t^j \quad (9)$$

3. If $i = \lfloor \frac{n}{2} \rfloor$

$$N_i^n(t) = 2 \binom{2\lfloor \frac{n}{2} \rfloor - 2}{\lfloor \frac{n}{2} \rfloor - 1} t^{\lfloor \frac{n}{2} \rfloor} (1-t)^{\lfloor \frac{n}{2} \rfloor} = 2 \binom{2\lfloor \frac{n}{2} \rfloor - 2}{\lfloor \frac{n}{2} \rfloor - 1} t^{\lfloor \frac{n}{2} \rfloor} \sum_{j=0}^{\lfloor \frac{n}{2} \rfloor} (-1)^j \binom{\lfloor \frac{n}{2} \rfloor}{j} t^j = \sum_{j=\lfloor \frac{n}{2} \rfloor}^{\lfloor \frac{n}{2} \rfloor} (-1)^{j-\lfloor \frac{n}{2} \rfloor} 2 \binom{2\lfloor \frac{n}{2} \rfloor - 2}{\lfloor \frac{n}{2} \rfloor - 1} \binom{\lfloor \frac{n}{2} \rfloor}{j-\lfloor \frac{n}{2} \rfloor} t^j$$

It can be simplified as follows

$$N_i^n(t) = \sum_{j=i}^{n-i} (-1)^{j-i} 2 \binom{2i-2}{i-1} \binom{n-i}{j-i} t^j$$

4. If $i = \lceil \frac{n}{2} \rceil$

$$N_i^n(t) = N_{n-i}^n(1-t) = 2 \binom{2\lfloor \frac{n}{2} \rfloor - 2}{\lfloor \frac{n}{2} \rfloor - 1} (1-t)^{\lfloor \frac{n}{2} \rfloor} t^{\lfloor \frac{n}{2} \rfloor} = \sum_{j=\lfloor \frac{n}{2} \rfloor}^{\lfloor \frac{n}{2} \rfloor + \lfloor \frac{n}{2} \rfloor} (-1)^{j-\lfloor \frac{n}{2} \rfloor} 2 \binom{2\lfloor \frac{n}{2} \rfloor - 2}{\lfloor \frac{n}{2} \rfloor - 1} \binom{\lfloor \frac{n}{2} \rfloor}{j-\lfloor \frac{n}{2} \rfloor} t^j$$

It can be simplified as follows:

$$N_i^n(t) = \sum_{j=i}^{n} (-1)^{j-i} 2 \binom{2(n-i-1)}{n-i-1} \binom{n-i}{j-i} t^j.$$

5. Case $i = \lceil \frac{n}{2} \rceil + 1$

$$N_i^n(t) = N_{n-i}^n(1-t) = \binom{2\lfloor \frac{n}{2} \rfloor - 2}{\lfloor \frac{n}{2} \rfloor - 1} (1-t)^{\lfloor \frac{n}{2} \rfloor - 1} t^{\lfloor \frac{n}{2} \rfloor + 1} = \sum_{j=\lfloor \frac{n}{2} \rfloor + 1}^{2\lfloor \frac{n}{2} \rfloor} (-1)^{j-(\lfloor \frac{n}{2} \rfloor + 1)} \binom{2\lfloor \frac{n}{2} \rfloor - 2}{\lfloor \frac{n}{2} \rfloor - 1} \binom{\lfloor \frac{n}{2} \rfloor - 1}{j-(\lfloor \frac{n}{2} \rfloor + 1)} t^j$$

Figure 1. Example1 A curve with DP control point

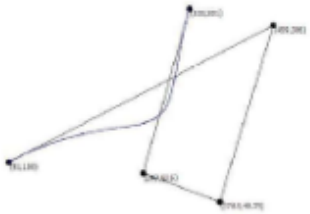


Figure 2. Same curve with Fig.1 after transform from DP control point to Wang-Ball control point by using Monomial form conversion.

It can be simplified as follows:

$$N_i^n(t) = \sum_{j=n-i+2}^{2(n-i+1)} (-1)^{j-n+i} \binom{2(n-i)}{n-1} \binom{n-1}{j-n+i-2} t^j$$

6. For $\lfloor \frac{n}{2} \rfloor + 1 \leq i \leq n$.

$$\begin{aligned} N_i^n(t) &= N_{n-i}^n(1-t) \\ &= \binom{\lfloor \frac{n}{2} \rfloor - 1 + n - i}{n-i} (1-t)^{n-i} t^{\lfloor \frac{n}{2} \rfloor} \\ &= \sum_{j=\lfloor \frac{n}{2} \rfloor}^{n-i+\lfloor \frac{n}{2} \rfloor} (-1)^j \binom{\lfloor \frac{n}{2} \rfloor - 1 + n - i}{n-i} \binom{n-i}{j-\lfloor \frac{n}{2} \rfloor} t^j \end{aligned}$$

IV. Empirical Results

In general, the technique [3] used to convert between 2 polynomial has to transform by converting to Bézier control before. For the instance to convet DP control point to Wang-Ball control point is to convert DP control point into Bézier control points and then transforms such Bézier control points to Wang-Ball points. This technique has been used until now. The disadvantages of this method are its complicated steps and the complexity in the implementation. This leads to our two new representations that can convert all polynomials into their corresponding forms. Monomial matrix was simply method to transform all Non-Rational curves.

Figure.1 shows the curve represented by DP control point. The set of DP control points is converted to new set of Wang-Ball control points via Monomial matrix method. D is represented Said-Ball monomial matrix and W^{-1} is represented invert matrix of Wang-Ball monomial form. DW^{-1} is the conversion matrix used to convert Said-Ball control point to Wang-Ball control points. Thus, the result of matrix multiplication between set of DP control points and DW^{-1} is a new set of Wang-Ball control points which represented same curve Figure.2

The result of old technique [3] and Monomial matrix conversion give the same curve after transformation. However, Monomial Matrix is simple and more application to convert into other form.

V. Conclusion

Obviously, the property of monomial form that related to monomial matrix can use to apply for many applications, for instance, Derivatives, degree elevation, degree reduction and conversion between each model of CAGD curves. Monomial matrix is evaluated curve model with simple programming for most programming languages. It is more basic than the previous techniques and also using parallel programming to improve next step of computation process.

References

[1] Aphirukmatakun, Chanon and Dejdumrong Natasha, *Monomial Form for Curves in CAGD with Their Applications*, The 6th International Conference Computer Graphics, Imaging and Visualization(CGIV09), Tianjin University, Tianjin, China August 11-14,pp.211-216,2009.

[2] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design: A practical guide* , Academic Press, Morgan Kaufman Publishers,San Francisco, 5th edition,2002.

[3] G.Z. Wang S.M. Hu and T.G. Jin, *Properties of Two Types of Generalized Ball Curves*, Computer-Aided Design, vol. 28, no. 2, pp. 125-133, 1996.

[4] J. Delgado and J. M. Pena, *A Shape Preserving Representation with an Evaluation Algorithm of Linear Complexity*, Computer Aided Geometric Design, vol. 20, no.1, pp. 1-20, March 2003.

[5] Wu Hongyi, *Unifying Representation of Bézier Curve And Genaralized Ball Curves*, Appl. Math. J. Chinese Univ. Ser. B, vol. 5, no. 1, pp. 109-121, 2000.

[6] Chen Xinmeng Yu Dan, *Another Type Of Generalized Ball Curves And Surfaces*, Acta Mathematica Scientia, vol. 27B, no. 4, pp. 897-907, 2007.

JCSSE 2013, May 29-31, 2013

Monomial Forms of Two Generalized Ball Curves and Their Proofs

Dilokvith Savetseranee
Department of Computer Engineering
Faculty of Engineering
King Mongkut's University of Technology
Thonburi,Bangkok, Thailand
ds@kmutt.net

Natasha Dejdumrong
Department of Computer Engineering
Faculty of Engineering
King Mongkut's University of Technology
Thonburi,Bangkok, Thailand
natasha@cpe.kmutt.ac.th

Abstract— **There are several methods used to construct curves in CAGD, e.g., the de Casteljau's algorithm, Wang algorithm, Said-Ball algorithm. However, they are represented in the forms that are not suitable for performing geometric modeling. A proposed approach for evaluating Said-Ball and Wang-Ball curves by using monomial form was introduced by Aphirukmatakun and Dejdumrong in 2009. The authors did not provide any proofs for their propositions. Thus, this paper shows the proofs for the conversions from Said-Ball polynomials into its monomial form as well as that of the Wang-Ball polynomials.**

Keywords—Monomial form, Monomial Matrix, Curve Conversion.

I. INTRODUCTION

In geometric modeling and design, there are several algorithms used for curve, surface and solid constructions. Mathematical formulae represent geometric objects in the design process and in the industrial procedures. Convenience and construction simplification are the key concepts for product design. For instance, there are several rigid and flexible parts in a model in automobile factory; rooftop curve should be the same design and massive production.

Unfortunately, there are various methodologies to analyze shape especially in curve design. Bézier /Bernstein, Wang-Ball, and Said-Ball models are example models used for approximating curves. When any of them is used to represent the same curve, different algorithms will be specified by corresponding functions in each application. Consequently, programmers have to transform one model into another for the best solution by using relationships between two models. For example, transforming a Bézier curve into Wang-Ball model is the popular way to draw the Bézier curve using Wang algorithm. Such an algorithm is more effective performance to draw a curve in linear time complexity, $(O(n))$, than performance of Bézier algorithm, which is $(O(n^2))$.

In the past, curve/surface model transformations are complicated and tough in the implementation because of the polar form techniques. The conversion computations between two curves with the same degree have to be carefully programed. This can lead to the chance of programming mistakes and, sometimes, waste the time in computational process.

In 2009, Aphirukmatakun and Dejdumrong [1] presented a solution of various designed methodologies and algorithms. This technique is called "Monomial Form". They studied the new representations of CAGD curve polynomials and represented them in the terms of monomial forms (or power bases). The coefficient matrices will make the curve representations simplified. Moreover, it is easy to understand in the implementation process.

However, the authors did not provide the proofs of their propositions in terms of mathematics. Thus, this paper will introduce the proofs for each proposition in details.

II. LITERATURE REVIEW

A. Bézier curve

Bézier curve can be constructed by using either the de Casteljau algorithm or the linear combinations based on Bernstein polynomials provide by P. Bézier in 1962 [2]. Generally, this curve is composed of two components Bernstein polynomials and control points. This is why the Bézier curve is well-known and popular because each Bernstein polynomial and its corresponding control point influence the shape of the curve. However, it is not appropriate to efficiently draw the higher degree curves. Although there are some research presented the better algorithm for the curve evaluation. Those are Said-Ball and Wang-Ball algorithms.

B. Said-Ball Curve

In 1974, Ball defined a set of basis function for cubic curve and bicubic surfaces. Later in 1989, Said [3] investigated Said-Ball curves by generalizing the cubic Ball model into the higher degrees. Said-Ball curve is similar to Bézier curve except their blending functions. After that, Hu et. al. (1996) developed a recursive algorithm for Said-Ball curves. It makes the simplicity for Said-Ball curve construction. So Said was the best algorithm to draw a curve in that time. Unfortunately, Said-Ball algorithm is more complex to use than Wang-Ball algorithm.

C. Wang-Ball Curve

Wang-Ball curve were investigated by Wang [4] in Chinese language in 1987 but publicized later in 1996 by Hu et.al [5] with its degree elevation and degree reduction. A Wang-Ball

curve can be computed more efficient than both Bézier and Said-Ball curve because it requires linear time complexity, $O(n)$. After that, in 2001 Dejdumrong et al [6] extended to the rational Wang-Ball curves. The only inconvenience of Wang-Ball curve is its low influence of control points to the curve's shape, i.e., changing the control point positions are less affected to the curve.

Perhaps, users may need to transform a curve representation into another appropriate form for some specific applications, e.g., monomial forms, polar forms, and interpolation diagrams. Each method has its own benefit and weakness depending on its properties. Since the derivation of model transformation can be obtained by various techniques, there is no standard for the derivation of all curve algorithms of any degrees.

Therefore, this paper supports the propositions of monomial forms for two generalized Ball curves, which is to convert from the Said-Ball (or Wang-Ball) into its power basis. Flexibility, universal and simplification are the key properties of this technique. Programmers can reduce the mistake and avoid the waste time in the programming process. Thus, monomial form is a simple solution and probably develops for parallel computation by using coefficient matrix of power basis.

D. Monomial Matrices

Aphirukmatakun and Dejdumrong presented the proposition of monomial form in 2009. This technique showed that polynomial basis can be represented by a group of coefficient matrices. It can be called *monomial matrices*. The representations of two generalized Ball bases into their monomial form are expressed and well proven.

III. METHODOLOGY

Proposition1 Said-Ball Monomial Form

An n^{th}-degree Said-Ball curve, denoted by $S^n(t)$, given by n+1 control points, denoted by $\{V_i\}_{i=0}^{n}$, can be expressed in power basis form as follows:

$$S^n(t) = \sum_{i=0}^{n}\sum_{j=0}^{n} V_i \cdot \boldsymbol{S}_{i,j} \cdot t^j ,$$

where

$$S_{i,j} = \begin{cases} (-1)^{j-i}\binom{i+\lfloor \frac{n}{2} \rfloor}{i}\binom{\lfloor \frac{n}{2} \rfloor+1}{j-i} & , \quad \text{for } 0 \le i \le \lfloor \frac{n}{2} \rfloor - 1, \\ (-1)^{j-i}\binom{n}{i}\binom{i}{j-i} & , \quad \text{for } i = \frac{n}{2}, \\ (-1)^{j-\lfloor \frac{n}{2} \rfloor-1}\binom{\lfloor \frac{n}{2} \rfloor+n-i}{n-i}\binom{n-i}{j-\lfloor \frac{n}{2} \rfloor-1} & , \quad \text{for } \lfloor \frac{n}{2} \rfloor + 1 \le i \le n. \end{cases} \tag{1}$$

Proof:
The Said-Ball curve can be represented in the form of Said-Ball basis as follows:

$$S_i^n(t) = \begin{cases} \binom{\lfloor \frac{n}{2} \rfloor+i}{i} t^i (1-t)^{\lfloor \frac{n}{2} \rfloor+1} & , \text{ for } 0 \le i \le \lfloor \frac{n}{2} \rfloor, \\ \binom{n}{i} t^i (1-t)^i & , \text{ if } i = \frac{n}{2}, \\ S_{n-i}^n(1-t) & , \text{ for } \lfloor \frac{n}{2} \rfloor + 1 \le i \le n. \end{cases} \tag{2}$$

The proof can be done by transforming Said-Ball basis into the Said-Ball monomial form for each circumstance.

1. For $0 \le i \le \lfloor \frac{n}{2} \rfloor$,

$$\begin{aligned} S_i^n(t) &= \binom{\lfloor \frac{n}{2} \rfloor+i}{i} t^i (1-t)^{\lfloor \frac{n}{2} \rfloor+1} \\ &= \binom{\lfloor \frac{n}{2} \rfloor+i}{i} t^i \sum_{j=0}^{\lfloor \frac{n}{2} \rfloor+1} (-1)^j \binom{\lfloor \frac{n}{2} \rfloor+1}{j} t^j \\ &= \sum_{j=0}^{\lfloor \frac{n}{2} \rfloor+1} (-1)^j \binom{\lfloor \frac{n}{2} \rfloor+i}{i}\binom{\lfloor \frac{n}{2} \rfloor+1}{j} t^{j+i} \end{aligned}$$

Thus,

$$= \sum_{j=i}^{\lfloor \frac{n}{2} \rfloor+1+i} (-1)^{j-i}\binom{\lfloor \frac{n}{2} \rfloor+i}{i}\binom{\lfloor \frac{n}{2} \rfloor+1}{j-i} t^j \tag{3}$$

2. If $i = \frac{n}{2}$,

$$\begin{aligned} S_i^n(t) &= \binom{n}{i} t^i (1-t)^i \\ &= \binom{n}{i} t^i \sum_{j=0}^{i} (-1)^j \binom{i}{j} t^j \\ &= \sum_{j=0}^{i} (-1)^j \binom{n}{i}\binom{i}{j} t^{j+i} \end{aligned}$$

Thus,

$$= \sum_{j=i}^{2i} (-1)^{j-i}\binom{n}{i}\binom{i}{j-i} t^j \tag{4}$$

3. For $\lfloor \frac{n}{2} \rfloor + 1 \le i \le n$,

$$\begin{aligned} S_i^n(t) &= S_{n-i}^n(1-t) \\ &= \binom{\lfloor \frac{n}{2} \rfloor+(n-i)}{n-i}(1-t)^{n-i} t^{\lfloor \frac{n}{2} \rfloor+1} \\ &= \binom{\lfloor \frac{n}{2} \rfloor+(n-i)}{n-i} t^{\lfloor \frac{n}{2} \rfloor+1} \sum_{j=0}^{n-i} (-1)^j \binom{n-i}{j} t^j \\ &= \sum_{j=0}^{n-i} (-1)^j \binom{\lfloor \frac{n}{2} \rfloor+(n-i)}{n-i}\binom{n-i}{j} t^{j+\lfloor \frac{n}{2} \rfloor+1} \\ &= \sum_{j=\lfloor \frac{n}{2} \rfloor+1}^{n-i+\lfloor \frac{n}{2} \rfloor+1} (-1)^{j-(\lfloor \frac{n}{2} \rfloor+1)} \binom{\lfloor \frac{n}{2} \rfloor+(n-i)}{n-i}\binom{n-i}{j-(\lfloor \frac{n}{2} \rfloor+1)} t^j \end{aligned}$$

It can be simplified as follows:

$$S_i^n(t) = \sum_{j=\lfloor \frac{n}{2} \rfloor+1}^{n-i+\lfloor \frac{n}{2} \rfloor+1} (-1)^{j-(\lfloor \frac{n}{2} \rfloor+1)} \binom{\lfloor \frac{n}{2} \rfloor+(n-i)}{n-i}\binom{n-i}{j-(\lfloor \frac{n}{2} \rfloor+1)} t^j \tag{5}$$

Equations (3), (4) and (5) can be integrated into (1).

Proposition 2 Wang-Ball Monomial Form

A Wang-Ball curve, denoted by $A^n(t)$, provided by n+1 control points, denoted by $\{\mathrm{P}_i\}_{i=0}^{n}$, can be shown as

$$A^n(t) = \sum_{i=0}^{n}\sum_{j=0}^{n} \mathbf{P}_i \cdot a_{i,j} \cdot t^j,$$

$$a_{i,j} = \begin{cases} (-1)^{j-i}2^i\binom{i+2}{j-i} & , \quad \text{for } 0 \le i \le \left\lfloor \frac{n}{2} \right\rfloor - 1, \\ (-1)^{j-i}2^i\binom{n-i}{j-i} & , \quad \text{for } i = \left\lfloor \frac{n}{2} \right\rfloor \\ (-1)^{j-i}2^{n-i}\binom{n-i}{j-i} & , \quad \text{for } i = \left\lceil \frac{n}{2} \right\rceil, \\ (-1)^{j-n+i}2^{n-i}\binom{n-i}{j-n+i-2} & , \quad \text{for } \left\lceil \frac{n}{2} \right\rceil + 1 \le i \le n. \end{cases} \tag{6}$$

Proof:

The Wang-Ball curve can be expressed in the form of the bases as follows:

$$A_i^n(t) = \begin{cases} (2t)^i(1-t)^{i+2} & , \text{ for } 0 \le i \le \left\lfloor \frac{n}{2} \right\rfloor - 1, \\ (2t)^i(1-t)^{n-i} & , \text{ if } i = \left\lfloor \frac{n}{2} \right\rfloor, \\ (2(1-t))^{n-i}t^i & , \text{ if } i = \left\lceil \frac{n}{2} \right\rceil, \\ A_{n-i}^n(1-t) & , \text{ for } \left\lceil \frac{n}{2} \right\rceil + 1 \le i \le n. \end{cases} \tag{7}$$

The proof can be given by transforming Wang-Ball basis into the Wang-Ball monomial form.

1. For $0 \le i \le \left\lfloor \frac{n}{2} \right\rfloor - 1$,

$$\begin{aligned} A_i^n(t) &= (2t)^i(1-t)^{i+2} \\ &= 2^i t^i \sum_{j=0}^{i+2}(-1)^j\binom{i+2}{j}t^j \\ &= \sum_{j=0}^{i+2}(-1)^j 2^i\binom{i+2}{j}t^{j+i} \end{aligned}$$

Thus,

$$= \sum_{j=i}^{2i+2}(-1)^{j-i}2^i\binom{i+2}{j-i}t^j \tag{8}$$

2. If $i = \left\lfloor \frac{n}{2} \right\rfloor$,

$$\begin{aligned} A_i^n(t) &= (2t)^i(1-t)^{n-i} \\ &= 2^i t^i \sum_{j=0}^{n-i}(-1)^j\binom{n-i}{j}t^j \\ &= \sum_{j=0}^{n-i}(-1)^j 2^i\binom{n-i}{j}t^{j+i} \end{aligned}$$

Thus,

$$= \sum_{j=i}^{n}(-1)^{j-i}2^i\binom{n-i}{j-i}t^j \tag{9}$$

3. If $i = \left\lceil \frac{n}{2} \right\rceil$,

$$\begin{aligned} A_i^n(t) &= \left(2(1-t)\right)^{n-i}t^i \\ &= 2^{n-i}t^i\sum_{j=0}^{n-i}(-1)^j\binom{n-i}{j}t^j \\ &= \sum_{j=0}^{n-i}(-1)^j 2^{n-i}\binom{n-i}{j}t^{j+i} \end{aligned}$$

Thus,

$$= \sum_{j=i}^{n}(-1)^{j-i}2^{n-i}\binom{n-i}{j-i}t^j \tag{10}$$

4. For $\left\lceil \frac{n}{2} \right\rceil + 1 \le i \le n$,

$$\begin{aligned} A_i^n(t) &= A_{n-i}^n(1-t) \\ &= \left(2(1-t)\right)^{n-i}t^{n-i+2} \\ &= 2^{n-i}t^{n-i+2}\sum_{j=0}^{n-i}(-1)^j\binom{n-i}{j}t^j \\ &= \sum_{j=0}^{n-i}(-1)^j 2^{n-i}\binom{n-i}{j}t^{j+n-i+2} \\ &= \sum_{j=n-i+2}^{2(n-i+1)}(-1)^{j-(n-i+2)}2^{n-i}\binom{n-i}{j-(n-i+2)}t^j \\ &= \sum_{j=n-i+2}^{2(n-i+1)}(-1)^{j-(n-i)}2^{n-i}\binom{n-i}{j-(n-i+2)}t^j \end{aligned}$$

It can be simplified as follows:

$$A_i^n(t) = \sum_{j=n-i+2}^{2(n-i+1)}(-1)^{j-n+i}2^{n-i}\binom{n-i}{j-n+i-2}t^j. \tag{11}$$

Integrating equations from (8) to (11), it reaches to Eq. (6).

IV. Empirical Results

In general, the technique [5] used to convert from Said-Ball into Wang-Ball polynomials has to transform from Said-Ball into Bézier forms and then into Wang-Ball form. In other words, converting from Said-Ball control points into Wang-Ball control points is to transform from Said-Ball into Bézier control points and then transform such Bézier control points into Wang-Ball points. This technique has been used until

now. The disadvantages of this method are its complicated steps and the complexity in the implementation. This leads to our two new representations that can convert both generalized Ball polynomials into their corresponding forms. Monomial matrices provide a simple method to transform Ball curve interchangeably.

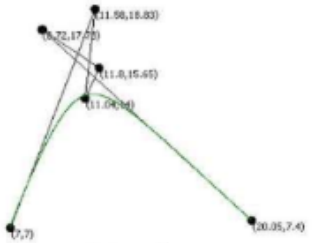


Fig. 1. A given Said-Ball Curve.

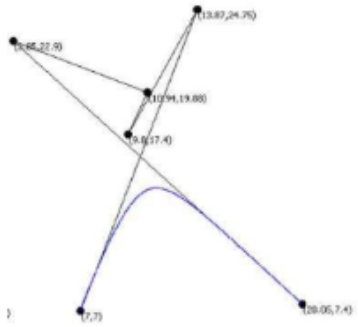


Fig. 2. The same curve as in Fig.1 after the transformation from Said-Ball into Wang-Ball control points using monomial matrixconversion.

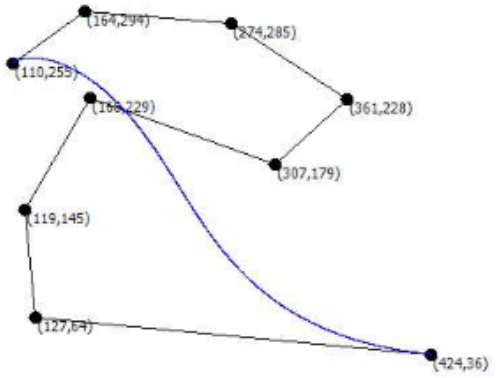


Fig. 3. Wang-Ball Curve for an original curve of example 2.

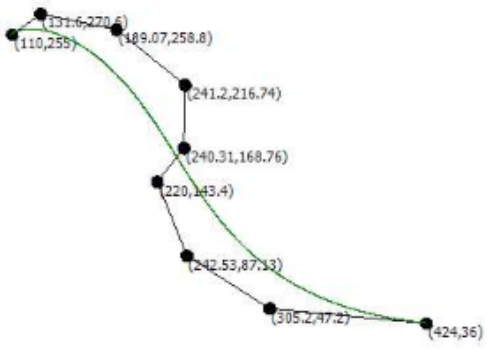


Fig. 4. Said-Ball curve after using monomial form in the conversion from Wang-Ball.

Figure 1 shows the Said-Ball curve represented by Said-Ball control points. The set of Said-Ball control points is converted into a new set of Wang-Ball control points via monomial matrix method. Let **S** be a Said-Ball monomial matrix and $\mathbf{W}^{-1}$ be an inverse matrix of Wang-Ball monomial form.

$\mathbf{SW}^{-1}$ is a direct conversion matrix used to convert Said-Ball into Wang-Ball control points. Thus, the result of matrix multiplication between a set of Said-Ball control points and SW^{-1} is a new set of Wang-Ball control points, which represented same curve Fig.2

Fig.3 and Fig.4 are the second example to represent the conversion from Wang-Ball control points into Said-Ball control points by monomial matrix.

The results from the technique in [5] and our monomial matrix conversion are the same after transformation. However, monomial matrix is simple and more applicable to convert into other form.

V. SUMMARY

Obviously, the property of monomial form that related to monomial matrix can use to be applied in many applications, for instances, derivatives, degree elevation, degree reduction and conversion between any models of CAGD curves. Monomial matrix is another kind of the curve model provided with a simple programming for most programming languages. It is easier than any existing techniques and can also be used for parallel and scalable programming to improve the speed of the computational process.

REFERENCES

[1] Aphirukmatakun, Chanon and Dejdumrong Natasha, Monomial Form for Curves in CAGD with Their Applications, The 6th International Conference Computer Graphics, Imaging and Visualization(CGIV'09),Tianjin University, Tianjin, China August 11-14,pp.211-216,2009.

[2] G. Farin, Curves and Surfaces for Computer Aided Geometric Design: A practical guide., Academic Press, Morgan Kaufman Publishers, San Francisco, 5th edition,2002.

[3] H. B. Said, "Generalized Ball Curve and Its Recursive Algorithm," ACM Transactions on Graphics, vol. 8, pp. 360–371, 1989..

[4] G. J.Wang, "Ball Curve of High Degree and Its Geometric Properties," Appl. Math.:A Journal of Chinese Universities, vol. 2, pp. 126–140, 1987.

[5] G.Z. Wang S.M. Hu and T.G. Jin, "Properties of Two Types of Generalized Ball Curves," Computer-Aided Design, vol. 28, no. 2, pp. 125–133, 1996.

[6] N. Dejdumrong, "Efficient Algorithms for Non-Rational and Rational Bézier Curves," 2008.

APPENDIX C

Conversion Test with Commercial Geometric Figure (Open Source)

This part is the commercial figure conversion test. The vector files are downloaded from free online database. Mathematica Software can read files and display as the figure as approximation curve model. Most of examples are commercial logo. There are both figures and letters in the examples. Each figure is composed by several curves. Thus, conversion step are convert each little fragment then assemble all as the result.

1. Horse Logo

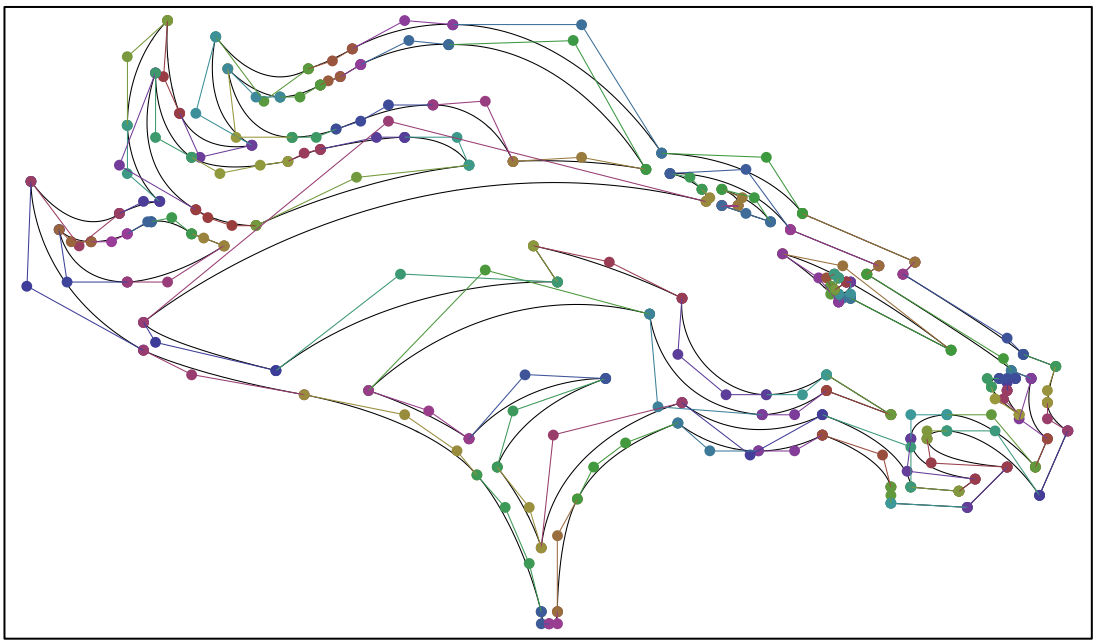


Figure 1 Input of Conversion is displayed as Approximation (Bézier) curve with its control points.

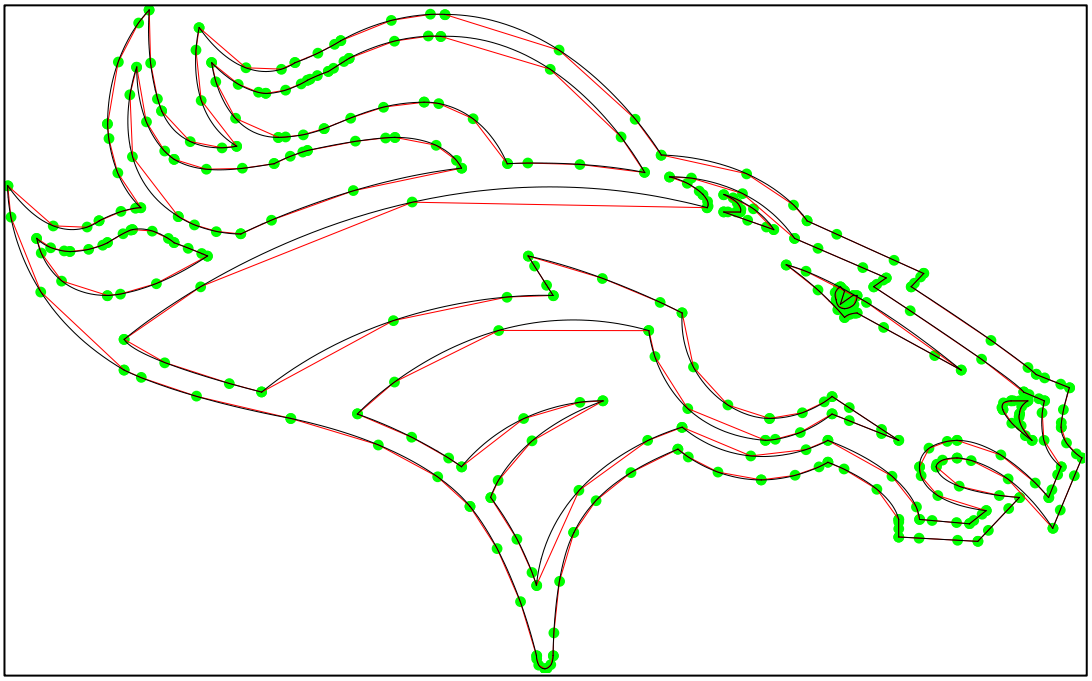


Figure 2 Conversion output is showed by Interpolation curve model.

2. Logo with Letters

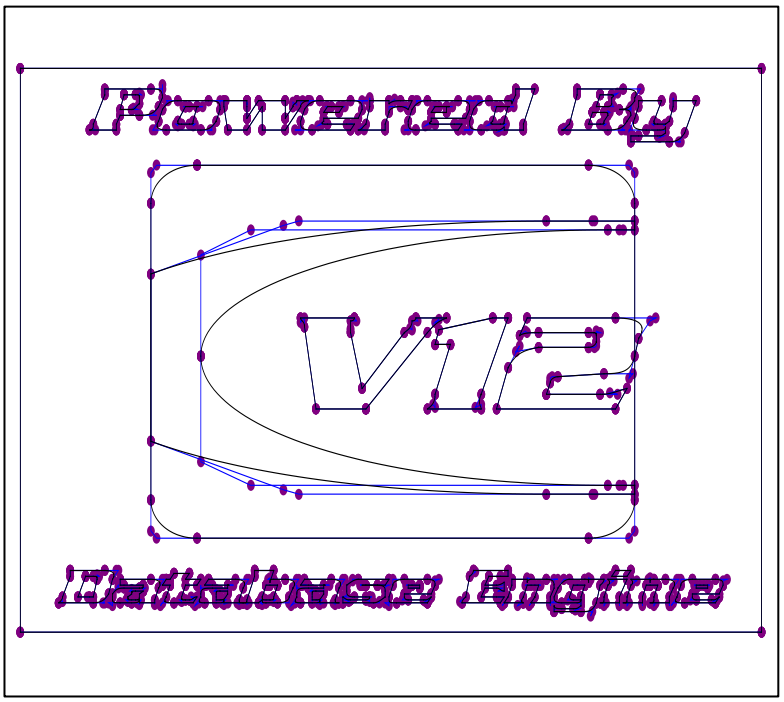


Figure 3 Input of Conversion is displayed Approximation (Wang-Ball) curve with its control points.

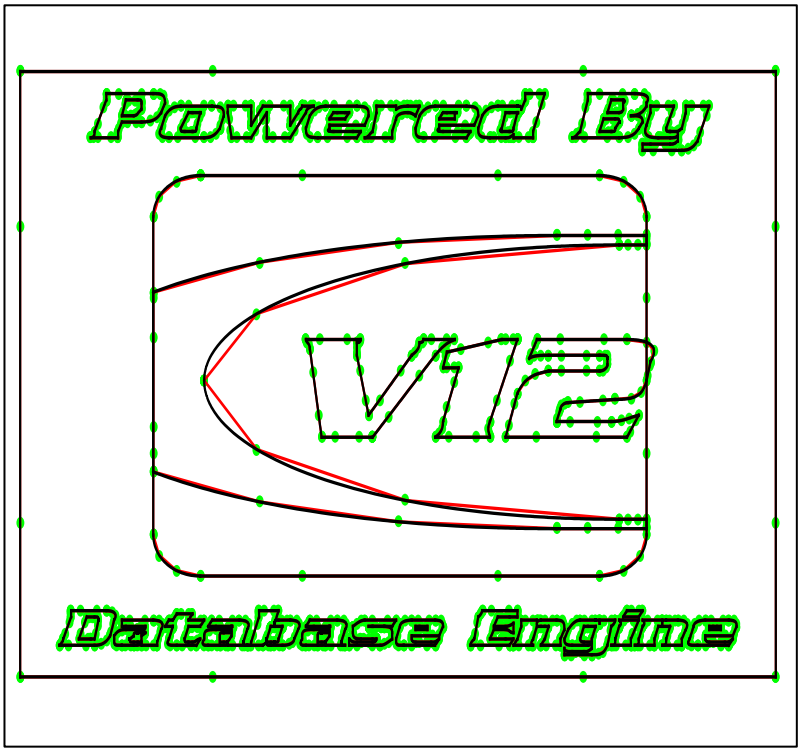


Figure 4 Conversion output is showed by Interpolation curve model.

3. Coca-Cola Logo (Letter Logo)

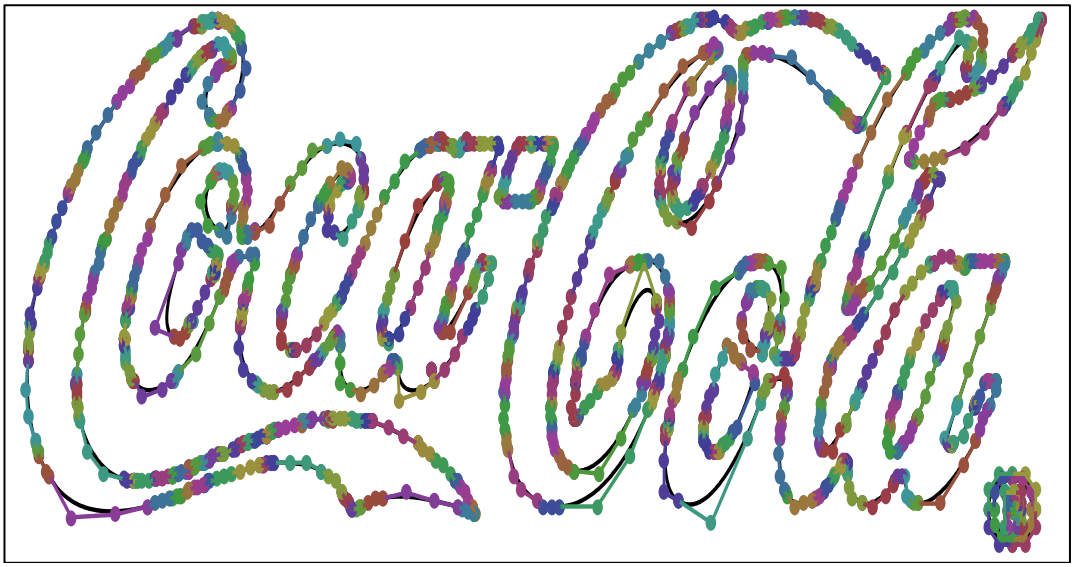


Figure 5 Input of Conversion is displayed as Approximation (Bézier) curve with its control points.

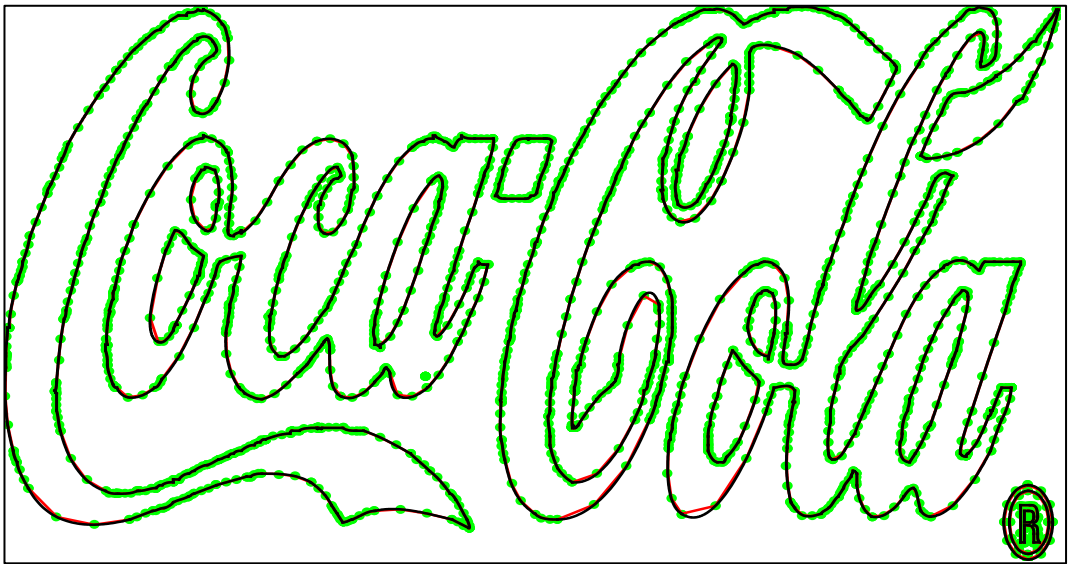


Figure 6 Conversion output is showed by Interpolation curve model.

4. Puma Logo (Image and Letter Logo)

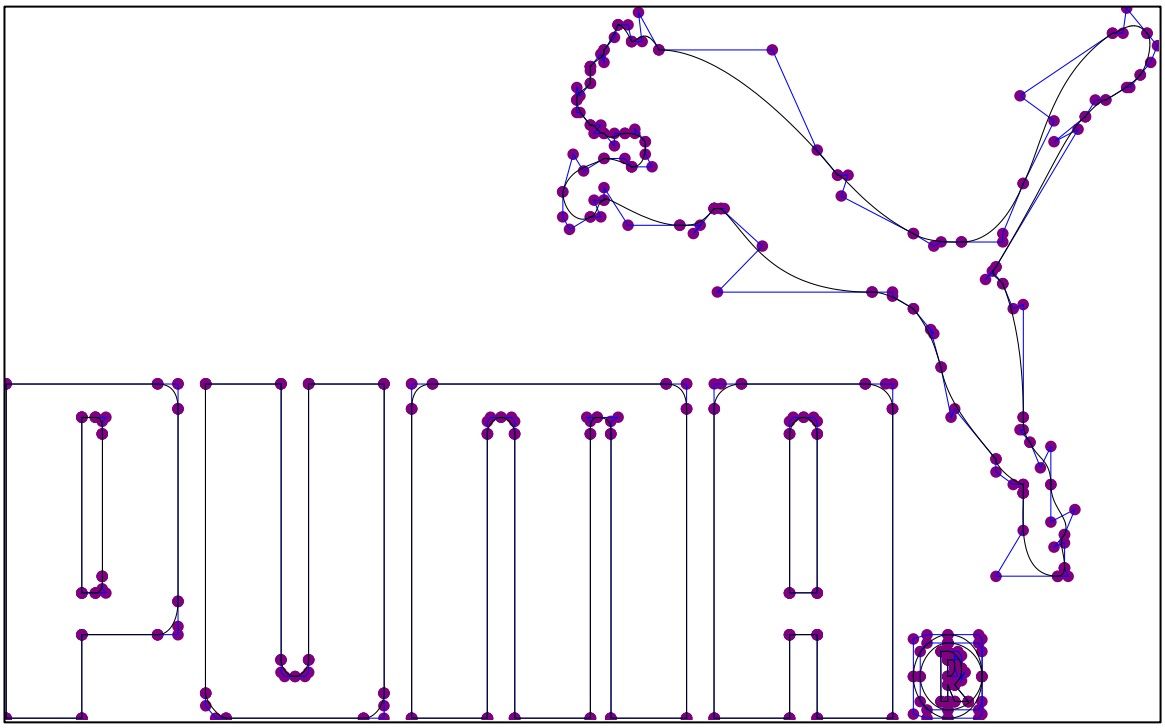


Figure 7 Input of Conversion is displayed as Approximation (Wang-Ball) curve with its control points.

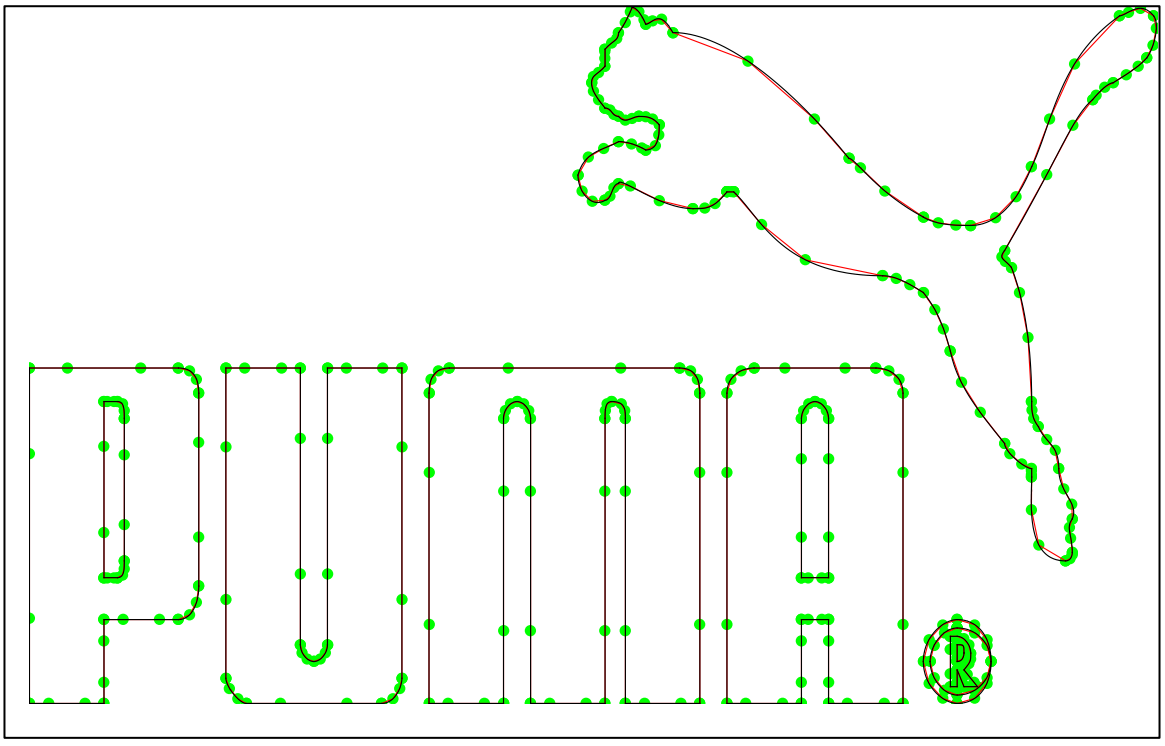


Figure 8 Conversion output is showed by Interpolation curve model.

5. **England Rugby Logo (Complex Logo)**

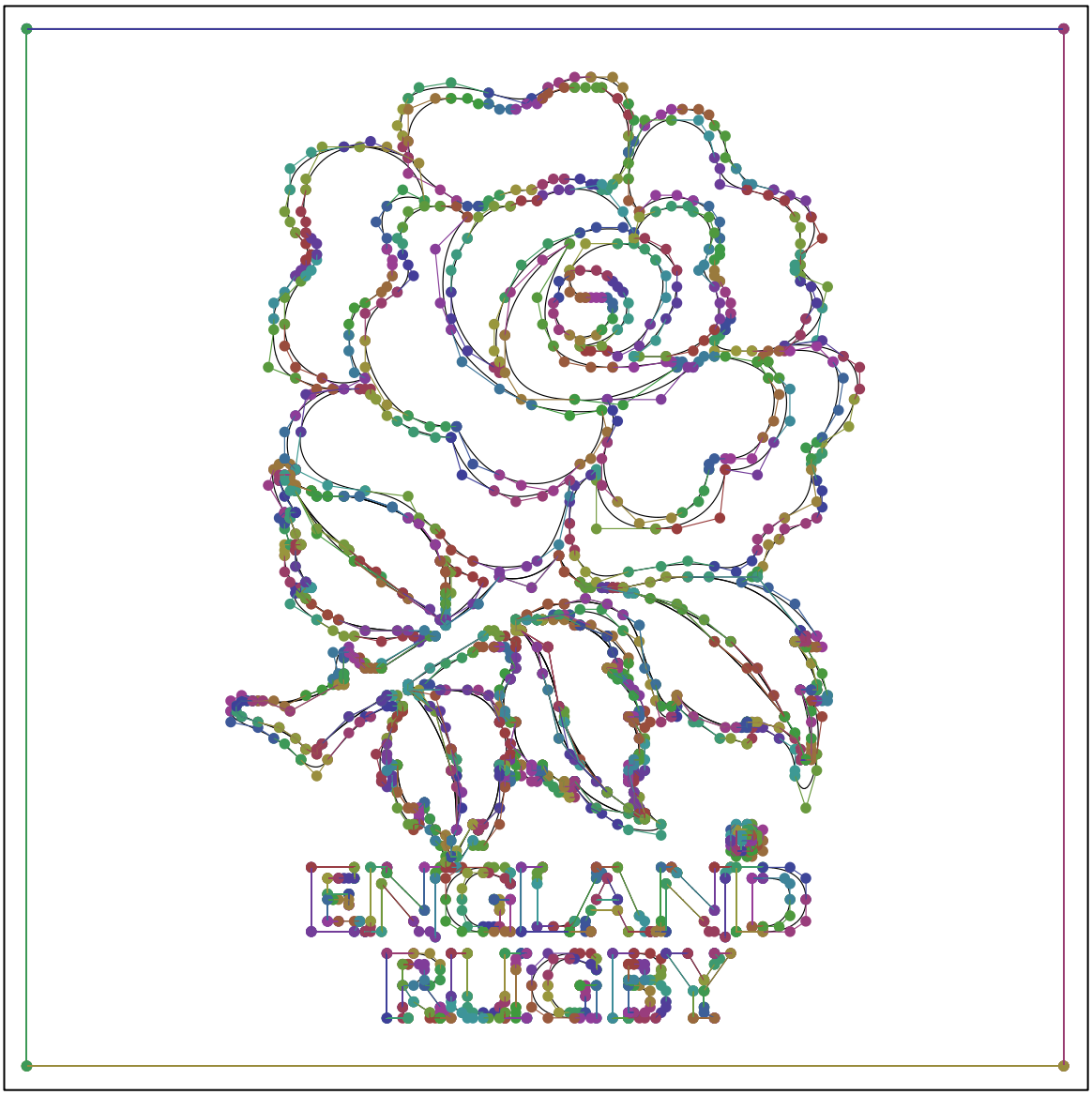


Figure 9 Input of Conversion is displayed as Approximation (Bézier) curve with its control points.

Figure 10 Conversion output is showed by Interpolation curve model.

CURRICULUM VITAE

NAME	Mr. Dilokvith Savetseranee
DATE OF BIRTH	8 August 1987
EDUCATIONAL RECORD	
HIGH SCHOOL	High School Graduation Wat Songtham School, Samutparkarn, 2006
BACHELOR'S DEGREE	Bachelor of Engineering (Computer Engineering) King Mongkut's University of Technology Thonburi, 2009
MASTER'S DEGREE	Master of Engineering (Computer Engineering) King Mongkut's University of Technology Thonburi, 2014
EMPLOYMENT RECORD	Southrim Technology Co. Ltd., Information Technology advisor FernUniversität in Hagen: Fakultät für Mathematik und Informatik Training Engineer, 2012 MICROSOFT (THAILAND) LIMITED Microsoft Executive Trainee 2011, 2010-2011
PUBLICATION	Savetseranee, D. and Dejdumrong, N., 2014, "Relationship of Interpolation and Approximation Curve Conversion Based on Monomial Form", **The 11th International Conference Computer Graphics, Imaging and Visualization (CGIV'14) Computer Graphics,** pp. 41 - 45 Savetseranee, D. and Dejdumrong, N., 2013, "New Monomial Forms Approach for DP and NB1 Curves with Their Proofs", **The 10th International Conference Computer Graphics, Imaging and Visualization (CGIV'13) Computer Graphics,** pp. 42 - 45

Savetseranee, D. and Dejdumrong,N., 2013, “Monomial forms of two generalized ball curves and their proofs”, **Computer Science and Software Engineering (JCSSE), 2013 10th International Joint Conference**, pp. 235- 239