FITTING B-SPLINES CURVES TO COARSE DATA CLOUD USING MODIFIED SQUARED DISTANCE MINIMIZATION METHOD

by

VINAY VIVEK MHALA

THESIS

Submitted in partial fulfillment of the requirements for

the degree of Master of Science in

Mechanical Engineering

at

The University of Texas at Arlington

August 1st, 2016

Arlington, Texas

ACKNOWLEDGEMENTS

I would like to express gratitude to my supervising professor, Dr. Robert M. Taylor for giving me this wonderful opportunity. His tremendous support and advice throughout my research work has been a great aid. I would like to thank Supervising Committee Members Dr. Ashfaq Adnan and Dr. Ratan Kumar for their valuable input and time. Finally, I would like to acknowledge, University of Texas at Arlington and Department of Aerospace and Mechanical Engineering for providing a great opportunity to nurture my interest.

DEDICATION

I will like to dedicate my work to my parents, Mr. Vivek Mhala and Ms. Usha Mhala and my sister Ms. Anushree Mhala. My achievements are result of their enormous efforts and support in all possible form. I also like to thank my friends for believing in me and encouraging me throughout my academic life. I would take this opportunity to thank my roommates Mr. Tanmay Srivastava, Mr. Kunal Choudhari and Mr. Avinash Rai for their help throughout my research work. Finally, I would like to thank god for his blessings.

ABSTRACT

Additive manufacturing has made us realize that we can fabricate complex shapes that were difficult to manufacture with subtractive processes. Topology optimizations results have complex shapes and have rough surfaces which are difficult to manufacture even by additive manufacturing. To automate the process of converting rough surfaces into smooth surfaces for the benefit of manufacturability would be very desirable.

Computing curve segments to approximate point cloud data that represents rough surfaces and then lofting it to have a smooth surface seems to be a promising methodology to achieve automation. We implement B-spline formulation along with Pottmann’s iterative method based on Squared Distance Minimization to automate the process of smoothening of rough surfaces. Calculation of foot points is a repetitive step within Squared Distance Minimization (SDM) method and accounts for considerable amount of computation time. In this research work, we proposed and implemented an algorithm on simple and complex shapes to help gain reduction in time required for foot point calculation. We will discuss ways to use this method effectively, dealing with instability, steps to reduce computation time and future work.

Keywords: B-spline, Squared Distance Minimization (SDM), SD error term, Foot point calculation, Point cloud data, Control points, Curve fitting.

TABLE OF CONTENTS:

TABLE OF FIGURES:

1. INTRODUCTION

In this section, we will discuss the problem definition and understand motivation behind the research work. The section, also explains the approach adopted to achieve the aim of automating, generation of smooth surfaces from noisy optimization result.

1.1 PROBLEM DEFINITION

Evolution of manufacturing processes have introduced us to a very versatile manufacturing methodology; 'Additive Manufacturing'. Additive manufacturing is a process of manufacturing a product by layer-by-layer deposition of materials as compared to the traditional subtractive manufacturing process, wherein material is removed from a sheet or block of material to manufacture a product. Additive manufacturing process can be classified into seven categories; VAT Photopolymerisation, Material Jetting, Binder Jetting, Material Extrusion, Powder Bed Fusion, Sheet Lamination and Directed Energy Deposition. Additive manufacturing has made us realize that we can manufacture complex geometry, which is difficult or impossible to manufacture by conventional manufacturing techniques. With advance in this field we have encountered various problems that make this method less practical.

Optimization is carried out on an object to have an optimal design for the same design consideration. Design optimization can be classified into three types: Shape optimization, Size optimization and Topology optimization. This research work deals with problems that are faced after topology optimization is performed on a design (Blattman, 2008). As material is removed from the original design the optimized result has noisy/rough surfaces, making it difficult to manufacture by subtractive as well as additive manufacturing. Also, manufacturing rough surfaces result into stress concentration, which is undesirable; Figure 1 and 2 demonstrates the scenario. Figure 1.1 and Figure 1.5 are the initial design and are consider for material reduction. Figure 1.2, Figure 1.3 and Figure 1.6 project results after topology optimization, of the initial model. Now, these rough surfaces need to be smoothened out so that the object is manufacturable. Software's such as SolidThinking, Evolve, Inspire, etc. help solve these problems, Figure 1.4, Figure 1.7 and Figure 1.8 demonstrates results of smoothened surfaces. These softwares require a professional to perform the task manually and the results will depend on knowledge and experience of the operator. This also leads to variation of results from one professional to other.

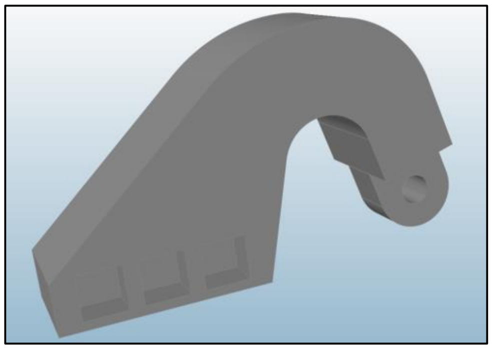


Figure 1.1: Original model considered for topology optimization

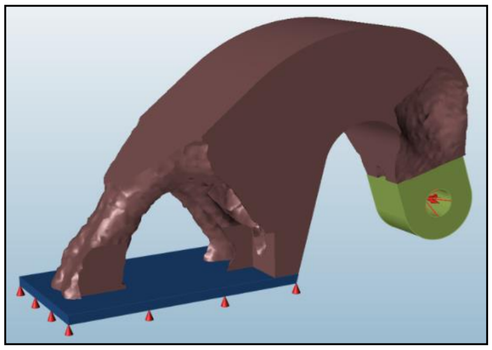


Figure 1.2: Optimized result for same design consideration with noisy surface

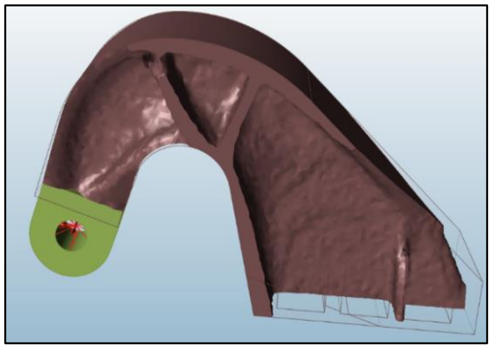


Figure 1.3: Optimized result for same design consideration with noisy surface

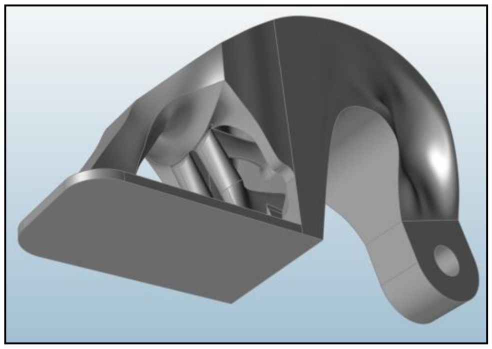


Figure 1.4: Smoothened surface using software (Evolve, Inspire)

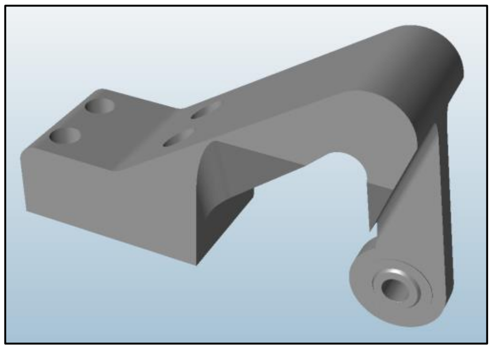


Figure 1.5: Original model considered for topology optimization

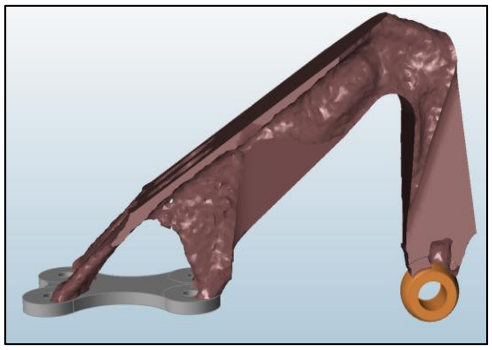


Figure 1.6: Optimized result for same design consideration with noisy surface

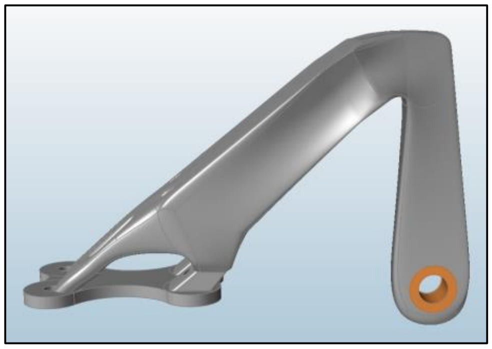


Figure 1.7: Smoothened surface using software (Evolve, Inspire)

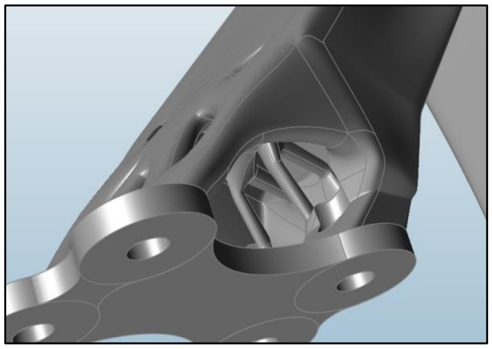


Figure 1.8: Smoothened surface using software (Evolve, Inspire)

We might be able to eliminate these restrictions and reduce the time consumed, by automating the process of smoothening of noisy surfaces. In this research paper, we have implemented one such method and have proposed an algorithm to speed it.

1.2 APPROACH

The process of acquiring smooth surfaces goes through three different stages; first we take cross-section of the result obtained after optimization and generate point cloud data from this cross-section. Then data cloud is grouped based on different contours as there could be more than one contour in a cross-section; as shown in Figure 1.9. The second step involves defining an approximate initial curve and then fitting curve segments to the data cloud with accuracy. Finally, use the curve segments to generate surfaces and deal with surface interaction at joints.

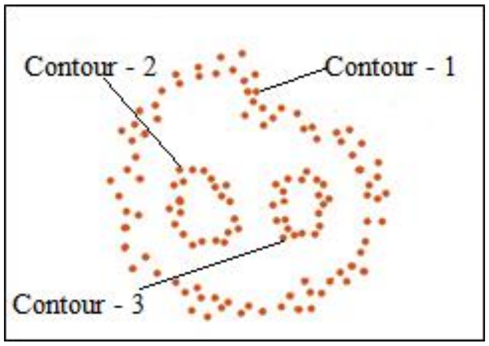


Figure 1.9: Different contours in one frame (cross-section)

In this research, we implemented B-spline curves to fit through the point cloud data that defines a rough cross-section, as B-spline formulation (equation (1)) provides local control and various other advantages over other types of curves such as Bezier Curve or Hermite Curve [Refer section 2.1 for explanation]. To improve the accuracy of fit, we will be using iterative optimization scheme based on Squared Distance Minimization (equation (5)) and propose an algorithm to reduce computation time required to calculate foot point on the curve from data cloud.

Flowchart shown in Figure 1.10, describes start to end steps involved in acquiring a manufacturable optimized part.

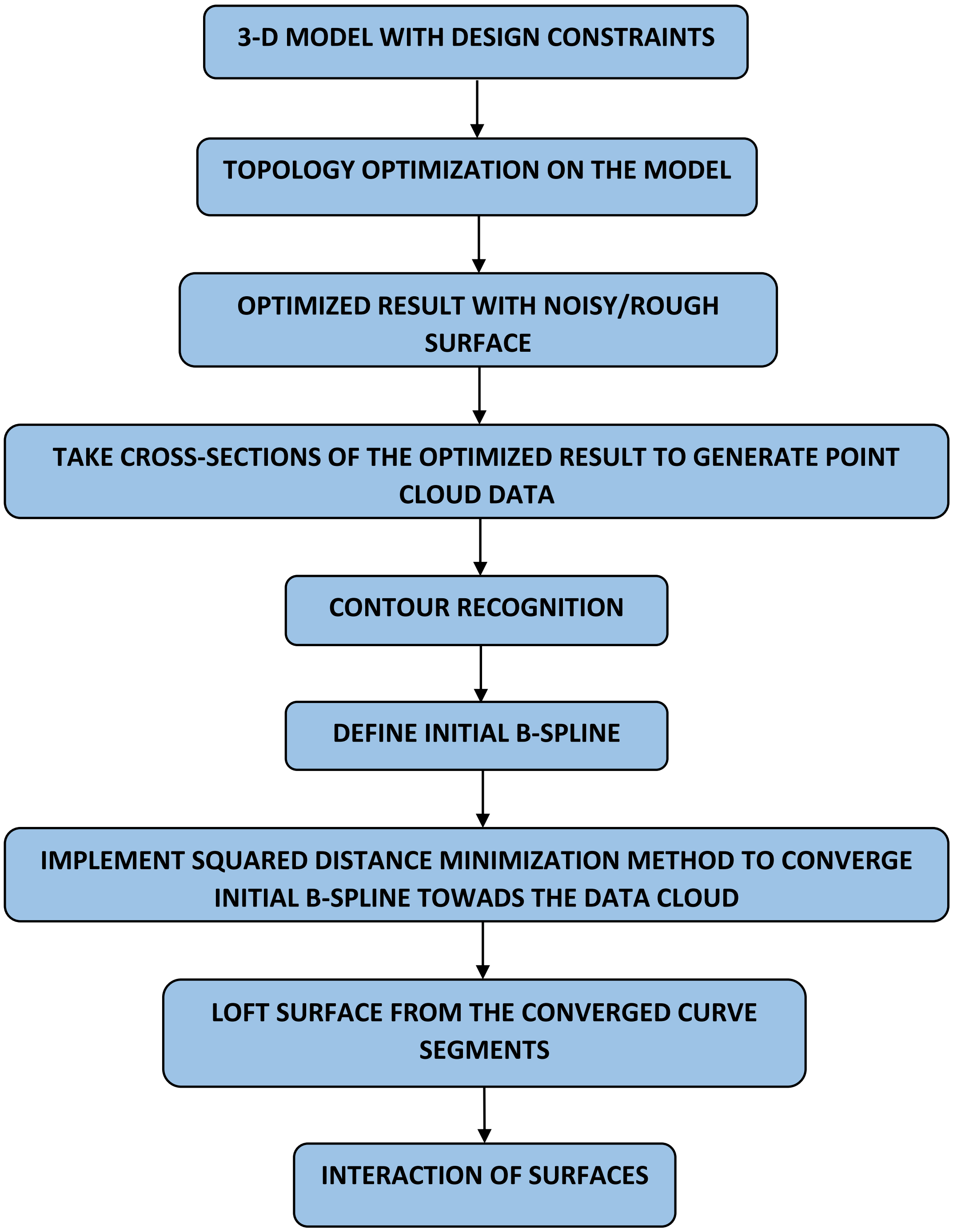

3-D MODEL WITH DESIGN CONSTRAINTS
TOPOLOGY OPTIMIZATION ON THE MODEL
OPTIMIZED RESULT WITH NOISY/ROUGH SURFACE
TAKE CROSS-SECTIONS OF THE OPTIMIZED RESULT TO GENERATE POINT CLOUD DATA
CONTOUR RECOGNITION
DEFINE INITIAL B-SPLINE
IMPLEMENT SQUARED DISTANCE MINIMIZATION METHOD TO CONVERGE INITIAL B-SPLINE TOWADS THE DATA CLOUD
LOFT SURFACE FROM THE CONVERGED CURVE SEGMENTS
INTERACTION OF SURFACES

Figure 1.10: Flow Chart

The paper will advance with discussion on background work, followed by implementation of the method and algorithm to reduce computation time. After explaining the methodology, we will examine some results and analyze limitations to SDM and ways to overcome them. Finally, we will derive conclusions from this research and comment on the future work.

2. BACKGROUND WORK

In this section, we will briefly discuss different curves (Mortenson, 2006)and error terms along with advantages B-spline and SDM formulation has over others. As mentioned in the section 1, we need curve segments that could fit through simple and complex point cloud data with ease. So, we need curve segments that will give higher degree of freedom and an error term that will give accurate updates to the control points. Now, we will briefly discuss some curve options and error terms in the following section:

2.1 CURVES

Curve can be roughly classified as Known-Form curves and Free-Form curves. Circle, ellipse, hyperbola, rectangle, triangle, etc. have a Known-Form while Bezier curves, Hermite's curves and B-spline curves don't have a known form and hence termed as Free-Form curves. Free-Form curves can be further classified as interpolation and approximation curves. Interpolation curves like Hermite's curve will always pass through the control points and have global control. Due to global control, if one control point is moved to a new location it will affect the entire curve, also this will affect the continuity of the curve segments; Refer Figure 2.1 and Figure 2.2. Hence, for our problem definition we don't opt for interpolation curves (Mortenson, 2006).

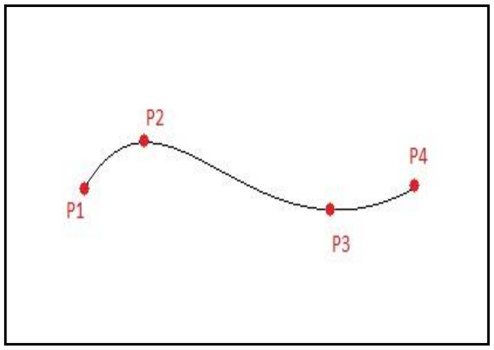


Figure 2.1: Initial Hermite Curve with 4 control points

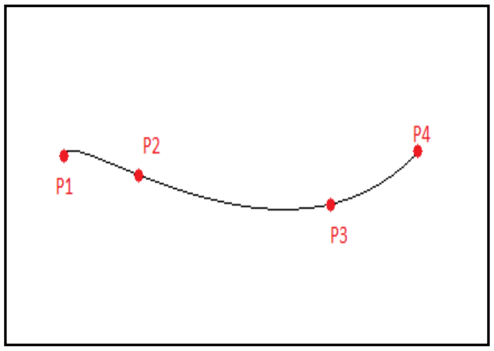


Figure 2.2: Effect of moving control point P2 on Hermite Curve

Bezier curve (Mortenson, 2006) has similar properties as of Hermite's curve but Bezier curve is an approximation curve and hence the curve segments do not pass through the control points. As mentioned, even Bezier curve has a disadvantage of providing only global control, similar to Hermite's curve [Refer Figure 2.3 and Figure 2.4]. As the number of control points increase so does the degree of curve which results in mathematical complexity because of which it is not possible to fit curve segments to complex shape with lower degree curve segments.

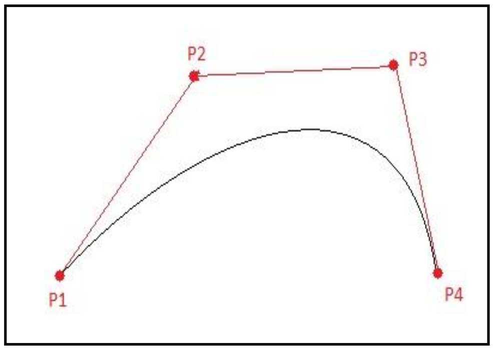


Figure 2.3: Initial Bezier Curve with 4 control points

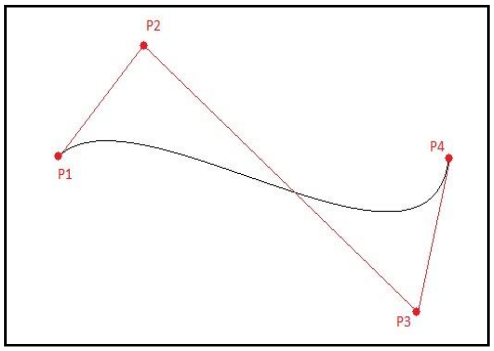


Figure 2.4: Effect of moving control point P3 on Bezier Curve

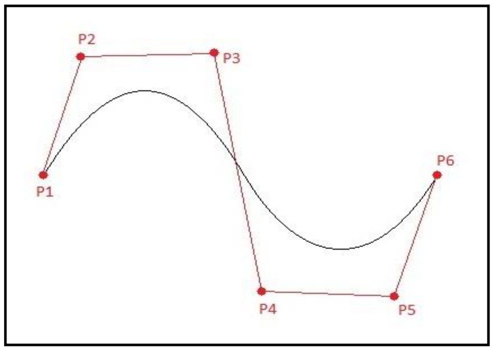


Figure 2.5: Graphic representation of B-Spline curve segment with 6 control points

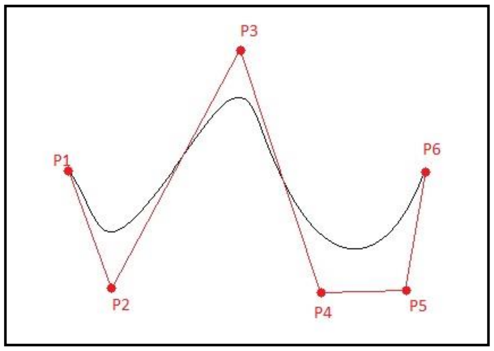


Figure 2.6: Effect of moving control point P2 on B-Spline Curve

B-spline curve is a generalized form of Bezier curve and hence share many properties with Bezier curve. B-spline provides local control, that means if one control point is moved to a new location only some part of the curve segment is affected which is decided by 'Knot' value [Refer Figure 2.5 and Figure 2.6]. Also, in case of B-spline curves degree of curve is independent of number of control points and that helps defining complex geometry with lower degree polynomial. The B-spline formulation is discussed below (Mortenson, 2006):

$$P_c(t) = \sum_{i=1}^{n} B_i(t)\, P_i \qquad (1)$$

Where,

$P_c(t)$ – B-spline curve

$B_i(t)$ – Basis function

t – Parameter $(0 \leq t \leq (n-K+2))$

P_i – Control point

i – Number of control points - Index (0,1,…n)

$$B_{i,1}(t) = \begin{cases} 1, & m_i \le t \le m_{i+1} \\ 0, & Otherwise \end{cases} \tag{2}$$

and

$$B_{i,k}(t) = \frac{(t - m_i) * B_{i,k-1}(t)}{m_{m+k-1} - m_i} + \frac{(m_{i+k} - t) * B_{i+1,k-1}(t)}{m_{i+k} - m_{i+1}} \tag{3}$$

Where,
K – Parameter controlling degree of curve (K – 1)
k – 2,…,K
i – Index (0,1,…n)
m_i – Knots

$$m_j = \begin{cases} 0 & if \quad j < K \\ i - K & if \ K \le j \le n \\ n - K + 2 & if \quad j > n \end{cases} \tag{4}$$

Where,
Value of t_j – o to (n-K+2)
j – o,…,n+k

As B-spline has many advantageous properties that will help fit complex shapes, we choose B-spline. Ultimately, our goal is to move towards Non-Uniform Rational B-spline which provides an added advantage of assigning weightages to control points giving more freedom in fitting complex and sharp shapes.

2.2 CURVE FITTING WITH ERROR TERMS

To fit the curve segments to data cloud, there are various curve fitting methods. The basic three curve fitting method could be classified as Interpolation (Brooks, Thomas, Wynne, & Coulston, 2012), Regression (Brown, 2001) (Dimitrov & Golparvar-Fard, 2014) (Fang & Gossard, 1995) and Fourier's Approximation (Brooks, Thomas, Wynne, & Coulston, 2012).In this research, we have used non-linear error type regression method. In this method, the control points of the curve are assigned updates using error terms generated by minimizing the objective function to fit the data cloud with accuracy. The objective function consists of error term and the regularization term (λf_S). In this section, we will discuss three different error terms and their advantages over one another:

a) Point Distance error term
b) Tangent Distance error term
c) Squared Distance error term

$$f_{error,k} = \frac{1}{2}\sum_{k}(error\ term) + \lambda f_S \qquad (5)$$

2.2.1 POINT DISTANCE ERROR TERM (e_{PD})

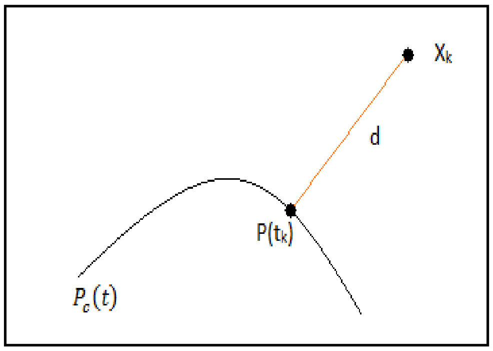


Figure 2.7: Graphical representation of point distance error term

Point distance error term calculates squared distances between foot point, P(t_k) and cloud point, X_k as shown in *Figure 2.6*Figure 2.7. So, the objective function (equation (5)) can be transformed into equation 3 to calculate PD error term to yield updates for control points (Wang, Pottmann, & Liu, 2006).

$$f_{PD,k} = \frac{1}{2}\sum_{k} e_{PD,k} + \lambda f_S \qquad (6)$$

where,

$$e_{PD} = \|P(t_k) - X_k\|$$

PD error term is commonly applied term in various area of curve fitting that includes computer graphics, surface fitting, etc. PD error terms are preferred because of their simplicity but they have poor approximation. Figure 2.9 - Figure 2.16 (Wang, Pottmann, & Liu, 2006) shows results of using PD, TD and SD error terms, PD error term takes numerous iterations to converge and fails to fit curve segments in the areas of complex shapes. PD error term has the slowest convergence and poorest approximation among the three terms. Figure 2.10 and Figure 2.14 demonstrate the issues with PD error term.

2.2.2 TANGENT DISTANCE ERROR TERM (e_{TD})

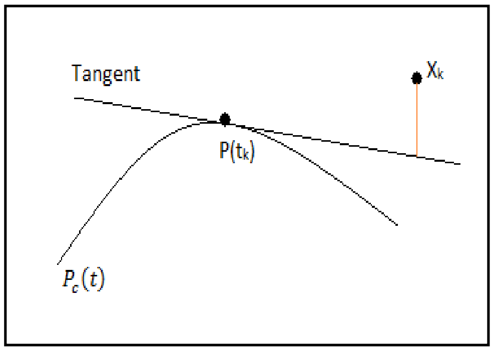


Figure 2.8: Graphical representation of point tangent error term

Tangent distance error term calculates squared distances between cloud point, X_k and tangent drawn to the curve $P_c(t)$ at $P(t_k)$ as shown in Figure 2.8. So, the objective function (equation (5)) can be transformed into equation (7) to calculate TD error term to yield updates for control points, where N_k is the normal at the foot point (Wang, Pottmann, & Liu, 2006).

$$f_{TD,k} = \frac{1}{2}\sum_k e_{TD,k} + \lambda f_S \tag{7}$$

where,

$$e_{TD,k} = [(P(t_k) - X_k)\, N_k]^2$$

TD error term converges in fewer iterations as compared to PD error term. TD error terms become unstable in case the data point lie on the tangent from the foot point as the distance yields zero value, thus resulting in poor approximation. This usually results, in areas of sharp shape change and resultant fit is undesirable as shown in Figure 2.15 (Wang, Pottmann, & Liu, 2006). While Figure 2.11 demonstrates that TD error term works with similar accuracy as SDM and requires equivalent number of iterations for simple shapes.

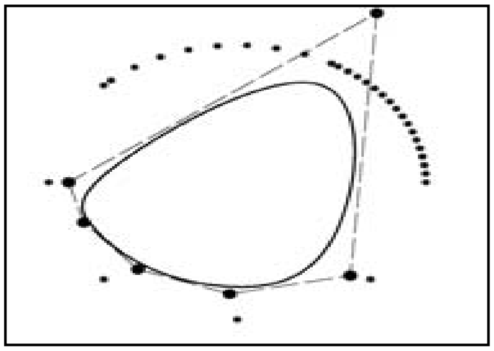


Figure 2.9: Initial Definition
Reference (Wang, Pottmann, & Liu, 2006)

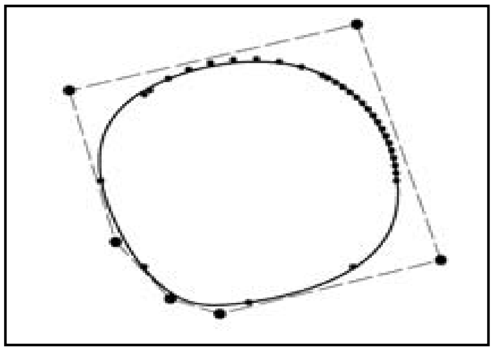


Figure 2.10: Solution by PD error term – after 10 iterations

Reference: (Wang, Pottmann, & Liu, 2006)

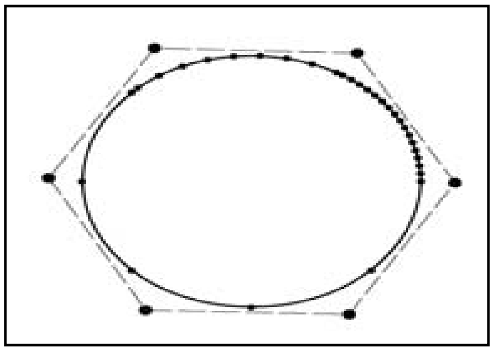


Figure 2.11: Solution by TD error term – after 10 iterations

Reference: (Wang, Pottmann, & Liu, 2006)

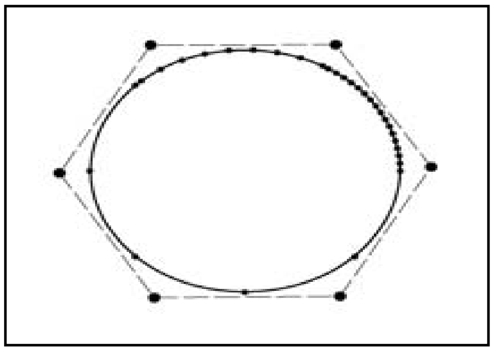


Figure 2.12: Solution by SD error term – after 10 iterations
Reference (Wang, Pottmann, & Liu, 2006)

Another example:

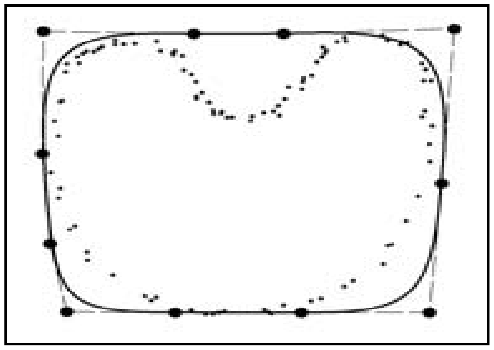


Figure 2.13: Initial Definition
Reference: (Wang, Pottmann, & Liu, 2006)

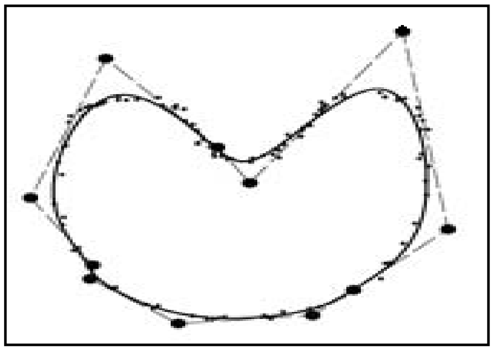


Figure 2.14: Solution by PD error term – after 20 iterations

Reference: (Wang, Pottmann, & Liu, 2006)

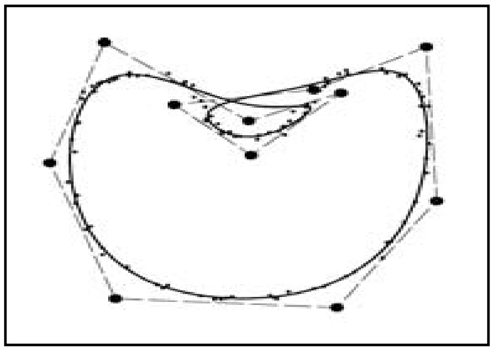


Figure 2.15: Solution by TD error term – after 20 iterations
Reference: (Wang, Pottmann, & Liu, 2006)

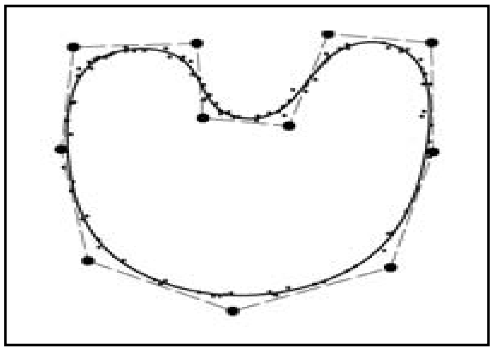


Figure 2.16: Solution by SD error term – after 20 iterations

Reference: (Wang, Pottmann, & Liu, 2006)

2.2.3 SQUARED DISTANCE ERROR TERM (e_{SD})

Squared Distance error term was introduced Weeping Wang, Helmut Pottmann and Yang Liu in (Wang, Pottmann, & Liu, 2006).SDM calculates the foot point more accurately as compared to PD and TD error term and hence the results are comparatively more stable and achieve faster convergence. Result due to PD, TD and SD error terms can be compared in Figure 2.9 - Figure 2.16. Figure 2.12 and Figure 2.16 show result due to SD error term which converged faster and in a stable fashion for simple as well as complex shape as compared to PD and TD error term. We will discuss working of SDM and Implementation of it in section 3.

3. METHODOLOGY

In this section, we will understand how Squared Distance Minimization method is implemented and ways to properly initialize the B-spline. Further, in this section we will propose and discuss an algorithm to speed up Squared Distance Minimization.

3.1 SQUARE DISTANCE MINIMIZATION

As discussed in section 1 and 2, we fit B-spline to the point cloud to obtain smooth surface from rough optimization results and to improve the accuracy of the fit, we need to adjust control points based on the point cloud data. To refine this fit, Squared Distance Minimization (SDM) method is implemented on simple and complex data clouds in this research. SDM formulation converges the initial curve towards the data cloud in 'n' number of iteration depending on complexity of data cloud and initial definition of control points (Cheng, et al.) (Yanga, Wanga, & Sun, 2003) (Wang, Pottmann, & Liu, 2006) (Pottmann, Leopoldseder, & Hofer).

Consider X_k where k = 1,2,3,...,n be a set of unorganized point cloud data obtained from a noisy cross-section as shown below Figure 3.1. To this data cloud, we define an initial B-spline $\mathrm{P_c(t)} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{B_i(t)\,P_i}$ that would approximately fit the point cloud data as shown in Figure 3.2. Here, $B_i(t)$ is the basis function of curve segments with 't' being the parameter and P_i's are the control points (Wang, Pottmann, & Liu, 2006).

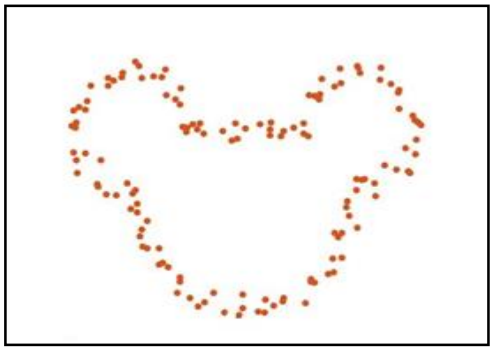


Figure 3.1: Scattered data cloud obtained by taking cross-section of optimized result

Now, to fit the initial curve more accurately to this data cloud we use the SDM formulation (equation (8)). The objective function, f_{SD} has two components, first the Squared Distance error term and second is known as regularization term (λf_S). The '½' in the objective function is considered for ease, as the objective function needs to be differentiated in future computation (Pekelny, 2005).

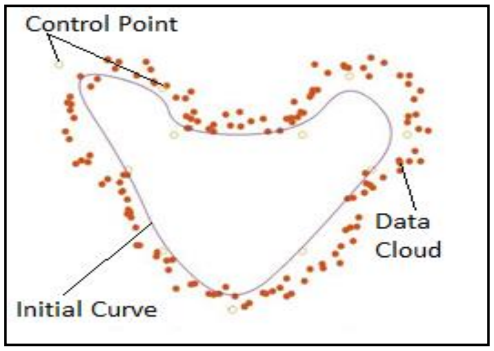


Figure 3.2: Defined control points to generate initial B-spline

$$f_{SD,k} = \frac{1}{2}\sum_{k} e_{SD,k} + \lambda f_S \tag{8}$$

Where,

f_{SD} – SDM objective function
λ – Weighing factor
f_S – Regularization term
e_{SD} – Squared distance error term

Closed Curve:

$$e_{SD,k}(D) = \begin{cases} \dfrac{d}{d-\rho}[(P_c(t_k) - X_k)^T T_k]^2 + [(P_c(t_k) - X_k)^T N_k]^2 \ , & if\ d < 0 \\ [(P_c(t_k) - X_k)^T N_k]^2 \ , & if\ 0 \le d < \rho \end{cases} \tag{9}$$

Open Curve:

$$e_{SD,k}(D) = \begin{cases} \cos\theta\, e_{PD,k} + (1-\cos\theta)e_{SD,k} \ , & for\ endpoints \\ e_{SD,k} \ , & for\ inner\ points \end{cases} \tag{10}$$

Where,

D – update for control point (D_X and D_Y for P_X and P_Y)
d – Distance between foot point and data point
ρ – radius of curvature at $P(t_k)$
T_k – Tangent at $P(t_k)$
N_k – Normal at $P(t_k)$

t_k – Parameter
θ – angle between the data point and curve segment
$e_{PD,k}$ – Point Distance error term

Squared distance error term calculates the update required to converge the curve with higher accuracy as compared to PD and TD error term. SD error term accounts Distance between foot point and data point, radius of curvature at $P(t_k)$, Tangent at $P(t_k)$, Normal at $P(t_k)$ to yield a more precise update (D_x, D_y).

Given the data cloud and define initial B-spline, let $P(t_k)$ be the closed point for a point cloud on the curve known as the foot point (Aigner & Juttler, 2005). Let the distance between X_k and $P(t_k)$ be '*d*' i.e $|d| = || X_k - P(t_k) ||$. Also, consider the local Frenet frame of curve with its origin at $P(t_k)$ with its two co-ordinate axis parallel to tangent and normal vector (T_k, N_k). The curvature of $P_c(t)$ at $P(t_k)$ be denoted by 'ρ'. If X_k and center of curvature are on the same side of the curve, then $0<d<\rho$ else *d* won't be the shortest distance, while $d > 0$ if X_k and center of curvature are on the same side of the curve. The SD error term formulation differs based on, side on which data point lie of the curve segment (equation (9)). Also, the formulation changes based on open or closed curve. Within open curve, the formulation considers for inner data points and outer data points differently as shown in equation (10).

The second half on the right hand side of equation (8) consists of, f_s which is the regularization term, ensures smoothness of the curves segments while λ alters the weight factor to the regularization term. λ is always a positive value and ranges from zero to one and is incremented with desired step (e.g.: 0.01 0r 0.001) (Mortenson, 2006). F_1 and F_2 are first order and second order regularization terms respectively as shown below and may have different weightage as shown in equation (11), (12) and (13).

$$F_1 = \int \| P_D'(t) \| \tag{11}$$

$$F_2 = \int \| P_D''(t) \| \tag{12}$$

$$f_{SD,k} = \frac{1}{2} \sum_k e_{SD,k} + \alpha F_1 + \beta F_2 \tag{13}$$

Where,
α and β are weight factors

The objective function is minimized as shown in equation (14) (Pekelny, 2005) to solving the system of linear equations to yield update D_x and D_y. D_x and D_y are assigned to P_x and P_y respectively to move the curve segment towards data cloud where P_x and P_y are X and Y co-ordinates of control points. These steps are repeated till the convergence is achieved.

$$\begin{pmatrix} \left(\frac{\partial f_{SD}}{\partial D_{1,x}}\right)_{1,x} & \left(\frac{\partial f_{SD}}{\partial D_{1,x}}\right)_{2,x} & \cdots & \left(\frac{\partial f_{SD}}{\partial D_{1,x}}\right)_{n,x} & \left(\frac{\partial f_{SD}}{\partial D_{1,x}}\right)_{1,y} & \left(\frac{\partial f_{SD}}{\partial D_{1,x}}\right)_{2,y} & \cdots & \left(\frac{\partial f_{SD}}{\partial D_{1,x}}\right)_{n,y} \\ & & & \vdots & & & & \\ \left(\frac{\partial f_{SD}}{\partial D_{n,x}}\right)_{1,x} & \left(\frac{\partial f_{SD}}{\partial D_{n,x}}\right)_{2,x} & \cdots & \left(\frac{\partial f_{SD}}{\partial D_{n,x}}\right)_{n,x} & \left(\frac{\partial f_{SD}}{\partial D_{n,x}}\right)_{1,y} & \left(\frac{\partial f_{SD}}{\partial D_{n,x}}\right)_{2,y} & \cdots & \left(\frac{\partial f_{SD}}{\partial D_{n,x}}\right)_{n,y} \\ \left(\frac{\partial f_{SD}}{\partial D_{1,y}}\right)_{1,x} & \left(\frac{\partial f_{SD}}{\partial D_{1,yy}}\right)_{2,x} & \cdots & \left(\frac{\partial f_{SD}}{\partial D_{1,y}}\right)_{n,x} & \left(\frac{\partial f_{SD}}{\partial D_{1,y}}\right)_{1,y} & \left(\frac{\partial f_{SD}}{\partial D_{1,y}}\right)_{2,y} & \cdots & \left(\frac{\partial f_{SD}}{\partial D_{1,y}}\right)_{n,y} \\ & & & \vdots & & & & \\ \left(\frac{\partial f_{SD}}{\partial D_{n,y}}\right)_{1,x} & \left(\frac{\partial f_{SD}}{\partial D_{n,y}}\right)_{2,x} & \cdots & \left(\frac{\partial f_{SD}}{\partial D_{n,y}}\right)_{n,x} & \left(\frac{\partial f_{SD}}{\partial D_{n,y}}\right)_{1,y} & \left(\frac{\partial f_{SD}}{\partial D_{n,y}}\right)_{2,y} & \cdots & \left(\frac{\partial f_{SD}}{\partial D_{n,y}}\right)_{n,y} \end{pmatrix} \begin{pmatrix} D_{1,x} \\ D_{2,x} \\ \vdots \\ D_{n,x} \\ D_{1,y} \\ D_{2,y} \\ \vdots \\ D_{n,y} \end{pmatrix} = \begin{pmatrix} b_{1,x} \\ b_{2,x} \\ \vdots \\ b_{n,x} \\ b_{1,y} \\ b_{2,y} \\ \vdots \\ b_{n,y} \end{pmatrix} \qquad (14)$$

Matrix Order,

(2n*2n) (2n*1) = (2n*1)

Where,

n = number of control points

DIFFRENTIATING (e_{SD}) WITH RESPECT TO D_X AND D_Y:

For $0 \leq d < \rho$,

$$e_{SD,k}\,(D) = \ [(P_c(t_k) - \ X_k)^T N_k]^2$$

$$\frac{\partial e_{SD,k}}{\partial D_{i,x}} = \frac{\partial[(P_c(t_k) - \ X_k)^T N_k]^2}{\partial D_{i,x}}$$

Differentiating with respect to D$_x$,

$$\frac{\partial e_{SD,k}}{\partial D_{i,x}} = 2\beta_i(t_k)N_{k,x}^2\left(\sum_{j=1}^{n}\beta_j(t_k)D_{j,x}\right) + 2\beta_i(t_k)N_{k,x}N_{k,y}\left(\sum_{j=1}^{n}\beta_j(t_k)D_{j,y}\right)$$
$$+ 2\beta_i(t_k)N_{k,x}[(P(t_k) - X_k)^T N_k]$$

Similarly, differentiating with respect to D$_y$,

$$\frac{\partial e_{SD,k}}{\partial D_{i,y}} = 2\beta_i(t_k)N_{k,x}N_{k,y}\left(\sum_{j=1}^{n}\beta_j(t_k)D_{j,x}\right) + 2\beta_i(t_k)N_{k,y}^2\left(\sum_{j=1}^{n}\beta_j(t_k)D_{j,y}\right)$$
$$+ 2\beta_i(t_k)N_{k,x}[(P(t_k) - X_k)^T N_k]$$

Similarly, we differentiate with respect to D$_x$ and D$_y$ for e$_{SD,k}$ where d < 0 and also for the regularization term.

3.1.1 STEPS

Following steps are carried out in SDM formulation:

1. Define an initial B-spline curve.
2. Calculating foot point on the initial curve for each point in the data cloud.
3. Minimize f_{SD} by solving liner system of equation to calculate SD error term, e$_{SD}$.
4. Assign updates to initial control points, generated by error term.
5. Generate the new B-spline.
6. Repeat steps 2 through 5 till converges is achieved.

3.1.2 PROBLEM

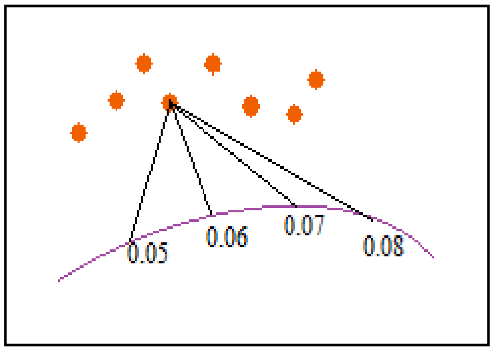


Figure 3.3: Calculating shortest distance to find foot point

Every point in the data cloud has the shortest distance on the curve as foot point. To find the foot point for any point X_k, its distance is calculated throughout the curve segments and the compared to find the shortest distance as shown in Figure 3.3. Consider the following example; there are 135 points in the point cloud data, we specify 10 control points to define initial curve. The initial curve has 10 curve segments with parameter 't_k' incrementing from 1 to 10 with an increment of 0.01. If the simulation iterates for 6 iterations to converge, the total number of equations solved is 810,000 equations. Solving these many equations require a substantial time and based on the experiments carried out, the total time required for this simulation is six hours and fourteen minutes. If the data cloud is even denser the time consumed will grow exponentially.

Exploring timing data in TABLE 1 reveals maximum time is utilized to solve SDM formulation and then for calculation of foot point. For this particular example SDM requires 1915 sec while foot point calculation take 1702 sec.

<u>TABLE 1: TIME DISTRIBUTION TABLE</u>

Functions Area	Time in seconds
Cloud Generation	0.97
B-spline Generation	9-15
Pre-requisite for SDM	1702
SDM formulation	1915
Regularization Term (F_1)	13
Regularization Term (F_2)	8

3.2 FAST ALGORITHM

As discussed in section 3.1.2, calculation of foot point takes a lot of time and should be dealt with. There are various methods in which we can encounter this problem. Fast Marching method developed by Professor James Sethian can be used for calculation of foot point (*d*). This method uses upwind, viscosity solution and finite difference scheme to solve non-linear partial differential equation for wave front propagation known as the Eikonal Equation (J.A.Sethain, 1995) (Sethain, 1999). Other possibility could be to generate a grid and approximate the value of all data points in one patch by calculating value of one data point (Wang, Pottmann, & Liu, 2006). In this paper, we would like to propose and implement an algorithm to calculate *d*, as discussed below in section 3.2.1.

3.2.1 CALCULATION OF FOOT POINT

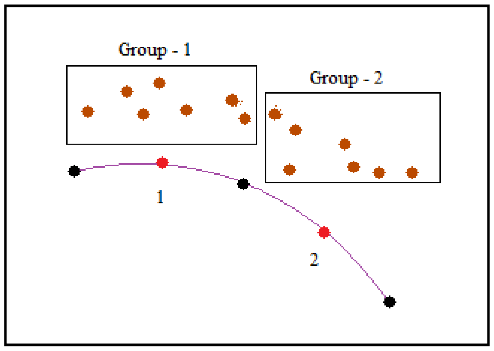


Figure 3.4: Fast Algorithm – Grouping of data points based on nearest midpoint

In this method, we divide the data cloud in groups by calculating their distances with the midpoint of each curve segments. Group all the points that have shortest distance with the same midpoint as shown in Figure 3.4. To find the foot point, we first find the distance between the start and end point of each curve segment with the point in the data cloud. The parameter value for the start and end point are known and we now approximate the parameter value for the data point in that group using equation (15) (Wang, Pottmann, & Liu, 2006) [Refer Figure 3.5 for understanding]. Consider the same example as in section 3.1.2, here we have 135 data points, 10 curve segments (due to initial definition of 10 control points) with parameter t_k incrementing from 1 to 10 with an increment of 0.01. In this case, we also evaluate unit normal, distance 'd', curvature ρ, parameter and theta in case of open curve. If the simulate takes 6 iterations to converge, we manage to bring down solving 810,000 equations to 32,400 equations. This change might not give the same accuracy but will reduce time considerably. The accuracy issues can be dealt with by dividing curve segments into more parts. Even after dividing the curve segments into smaller slices the number of equations remain comparatively low.

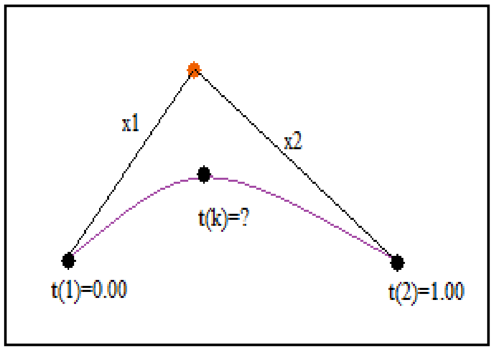


Figure 3.5: Interpolating foot point

$$t_k = (x_1 t_1 + x_2 t_2)/(x_1 + x_2) \tag{15}$$

After implementing this algorithm, the results obtain show considerable time reduction, which is demonstrated in the TABLE 2.

TABLE 2: TIME DISTRIBUTION TABLE

Functions Area	Time in seconds
Cloud Generation	0.97
B-spline Generation	120
Pre-requisite for SDM	300
SDM formulation	800
Regularization Term (F_1)	13
Regularization Term (F_2)	8

3.2.2 INITIALIZING CURVES

Defining initial curve is an important step in SDM simulation. Inaccurately defined B-spline would take numerous iteration to converge B-spline towards the point cloud. Figures below, shows two different scenarios Figure 3.6 describes less effectively define initial B-spline which takes 11 iterations and a total time of 10 hours and 30 minutes to achieve equivalent convergence as compared to a well-defined initial B-spline as shown in Figure 3.7.

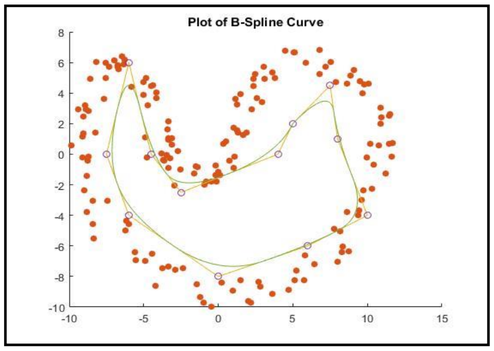


Figure 3.6: Less effective way of defining control points

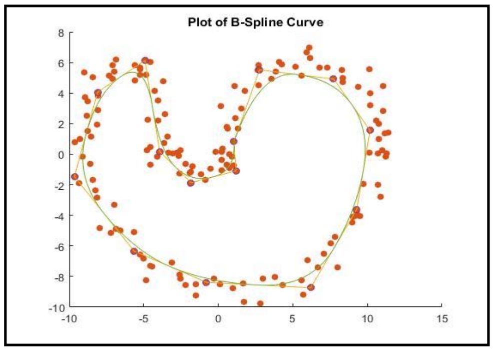


Figure 3.7: Effective way to define control points

3.2.2.1 MANUAL SETUP

Once we have the data cloud, control points can be assigned manually in an appropriate manner. The goal is to keep control points to the minimum, as number of control points increase the number of calculations increases as well and thus the time required for each iteration. Figure 4.1 in section 4.1, is an example of manual setup.

3.2.2.2 AUTOMATIC SETUP

Another technique that could result in an appropriate initial B-spline curve is by assigning every nth point in the data cloud as a control point. This works well in most of the cases as shown in Figure 4.5 in section 4.2 and Figure 4.9 in section 4.3, but in some cases if there is a quick change in shape and the nth control point does not lie in the region of quick shape change, it would take more iterations to fit the B-spline through the data cloud.

4. EXPERIMENTS AND RESULTS

In this section, we will analyze some experiments and there results that were carried out on simple and complex shapes with simple and coarse data cloud. The parameters were varied to study their effects and suggest guidelines to yield best possible results.

4.1 CASE 1: CIRCLE

The data cloud as shown in Figure 4.1, has 40 data points which represents a simple circle, to which 10 initial control points (P_x, P_y) are defined. Weightage of 0.005 is assigned to the regularization factor for smoothness of the curve segments. TABLE 3 displays various parameter and their values involved in this experiment.

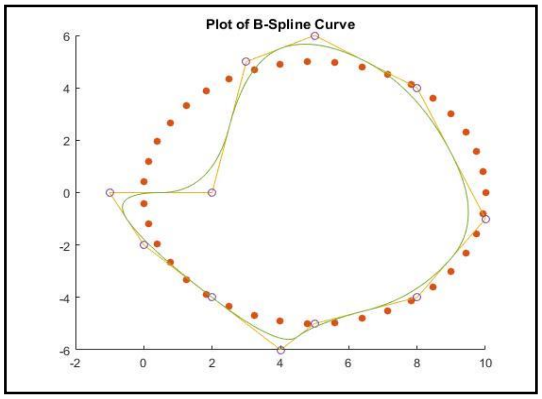


Figure 4.1: Simple data cloud representing a circle with control points defined less effectively

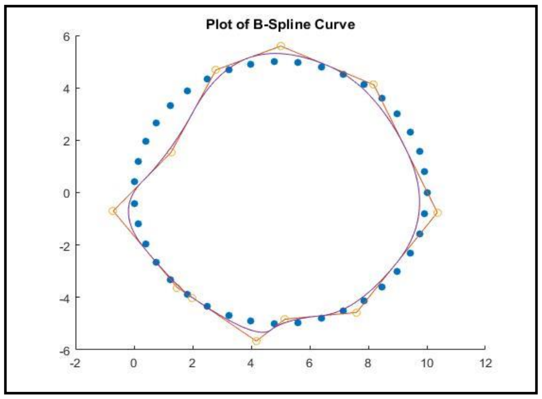


Figure 4.2: Convergence after 4 iterations

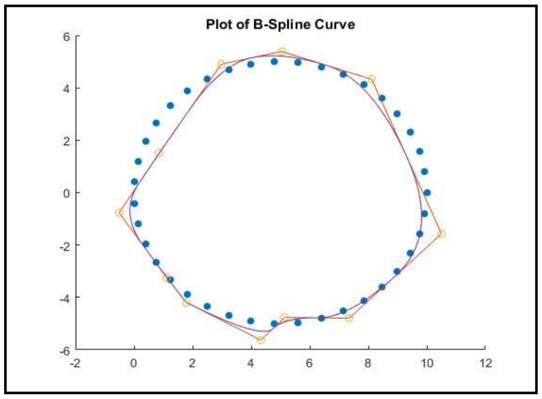


Figure 4.3: Convergence after 6 iterations

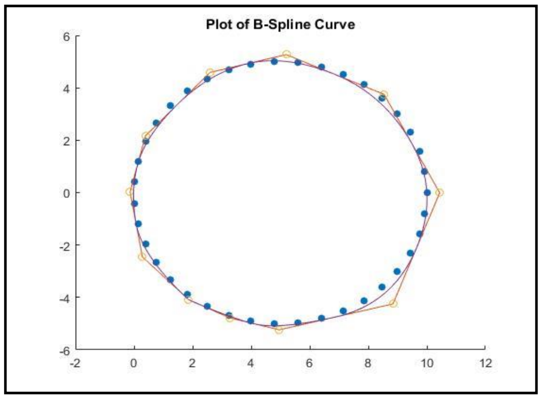


Figure 4.4: After 9 iterations - SDM formulation converges B-spline towards the data cloud

TABLE 3: PARAMETER SPECIFICATION

Parameters	Value
Data cloud	40
Control points	10
Curve segments	10
Weightage for Regularization term, λ	0.005
Iterations	9

Figure 4.2 and Figure 4.3 project results after 4th and 6th iteration during the simulation while Figure 4.4 shows converged B-spline after 9 iterations. The control points are defined in a less effective way so the simulation takes 9 iterations to converge with predefined convergence percentage of 3%. We can observe that, due to continuity issues we have C^0 continuity at start and end point of curve. As the curve segments are very close to the data cloud and the approximation represent a circle, we can comment that the SDM formulation successfully yields a good fit.

4.2 CASE 2: IRREGULAR SHAPE

The data cloud as shown in Figure 4.5, has 163 data points, to which 8 initial control points (P_x, P_y) are defined. The eight curve segments are further divided into 50 slices to improve accuracy and reduce number of iterations. Weightage of 0.005 is assigned to the regularization factor for smoothness of the curve segments.

TABLE 4 displays various parameter and their values involved in this experiment.

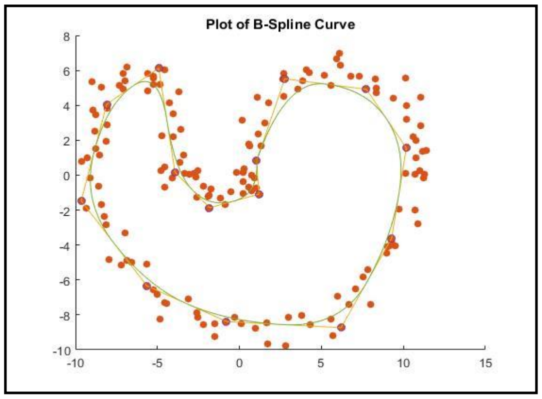


Figure 4.5: Random shape with control points defined closely to the data cloud (automatically defined control points)

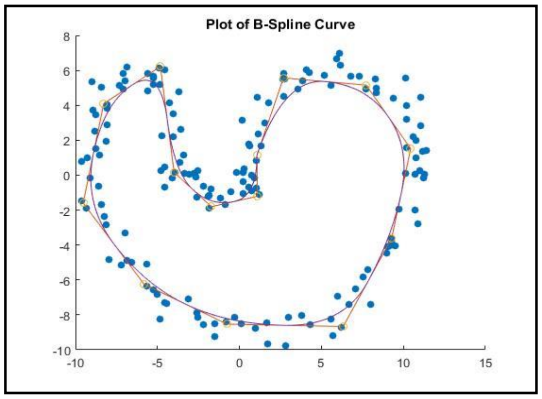


Figure 4.6: Convergence after 3 iterations

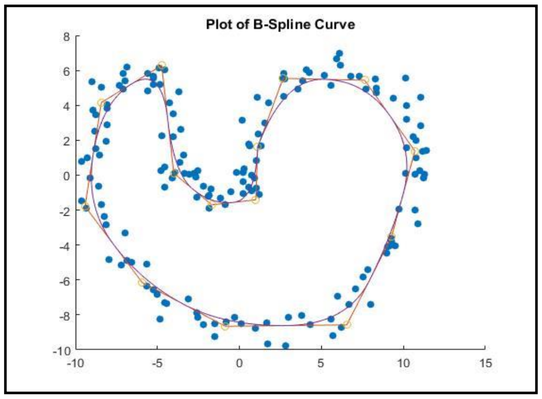


Figure 4.7: Convergence after 4 iterations

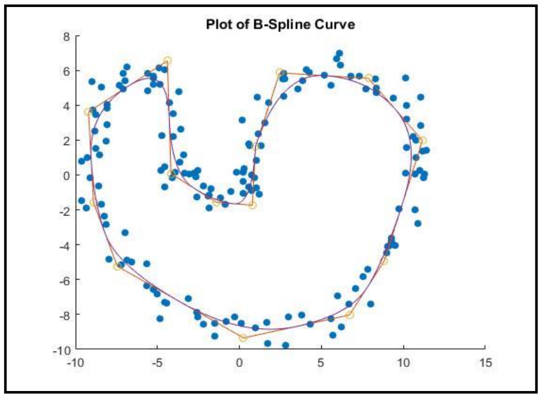


Figure 4.8: After 6 iterations - SDM formulation converges B-spline towards the data cloud

TABLE 4: PARAMETER SPECIFICATION

Parameters	Value
Data cloud	163
Control points	8
Curve segments	50
Weightage for Regularization term, λ	0.005
Iterations	6

Figure 4.6 and Figure 4.7 project results after 3rd and 4th iteration during the simulation while Figure 4.8 shows converged B-spline after 6 iterations. The control points are defined in an effective way, so the simulation takes 6 iterations to converge with predefined convergence percentage of 3%. The curve passes well within the data cloud, accounting for the coarseness of the data cloud. We can also observe, that due to continuity issues we have C^0 continuity at start and end point of curve. The SDM formulation is able to converge curve segments through irregular change of curvature, yielding a good fit.

4.3 CASE 3: IRREGULAR SHAPE WITH STRAIGHT LINE

The data cloud as shown in Figure 4.9, has 138 data points, to which initial control points (P_x, P_y) are defined. The 11 curve segments are further divided into 40 slices to improve accuracy and reduce the number of iterations. Weightage of 0.005 is assigned to the regularization factor for smoothness of the curve segments.

TABLE ***5*** displays various parameters and their values involved in this experiment.

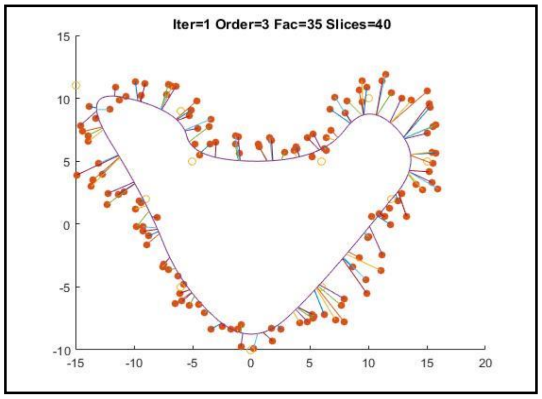


Figure 4.9: Random shape with control points defined closely to the data cloud (manually defined control points)

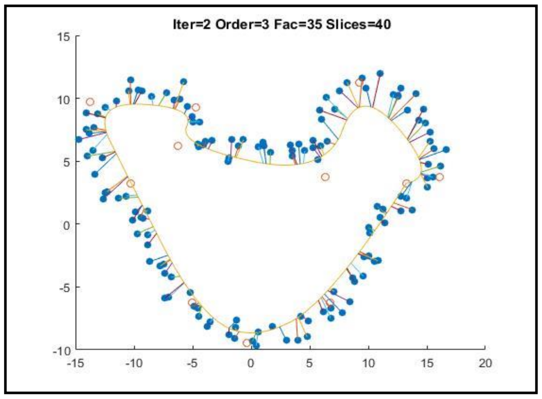


Figure 4.10: Convergence after 2 iterations

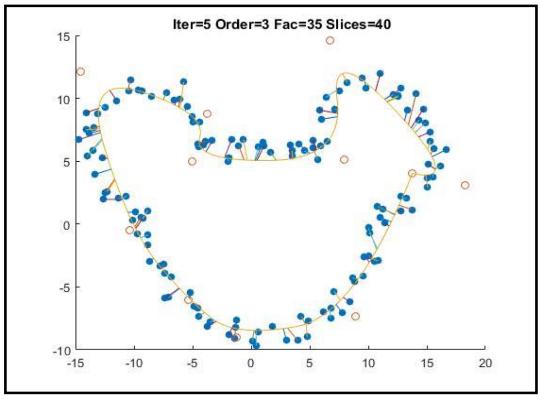


Figure 4.11: Convergence after 5 iterations

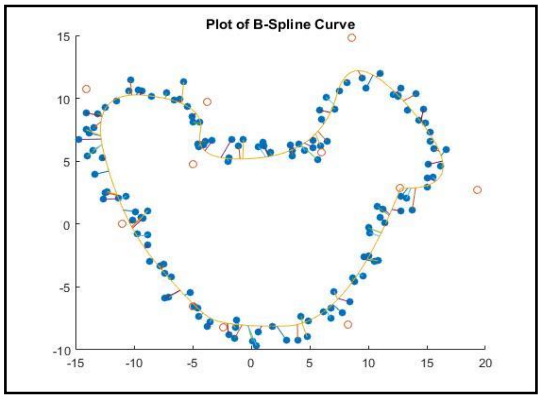


Figure 4.12: After 6 iterations - SDM formulation converges B-spline towards the data cloud

TABLE 5: PARAMETER SPECIFICATION

Parameters	Value
Data cloud	138
Control points	11
Curve segments	40
Weightage for Regularization term, λ	0.005
Iterations	6

Figure 4.10 and Figure 4.11 project results after 2nd and 5th iteration during the simulation while Figure 4.12 shows converged B-spline after 6 iterations. The control points are defined automatically in an effective way so the simulation takes 6 iterations to converge with predefined convergence percentage of 2%. We can also observe, that due to continuity issues we have C^0 continuity at start and end point of curve but the curve passes well within the data cloud in region of curvature change but has difficulty at corners where shape changes to a straight line.

4.4 CASE 4: IRREGULAR SHAPES WITH DEEP AND HIGH CURVATURE CAVITIES

The data cloud as shown in Figure 4.13, has 174 data points, to which 19 initial control points (P_x, P_y) are defined. The number of control points is high in this case due to the complexity of the data cloud. The 19 curve segments are further divided into 90 slices to improve accuracy and reduce number of iterations. Weightage of 0.005 is assigned to the regularization factor for smoothness of the curve segments. Table 6 displaces various parameters and their values involved in this experiment.

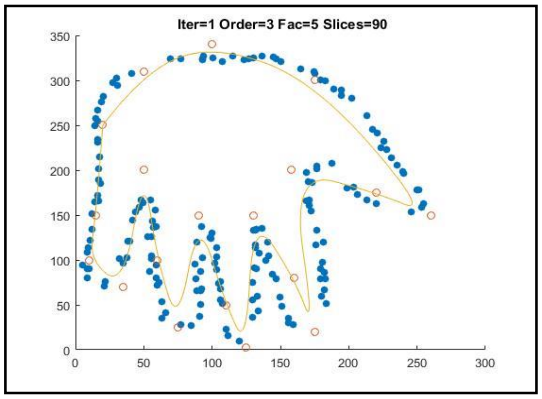


Figure 4.13: Irregular data cloud with complex and sharp change in shape

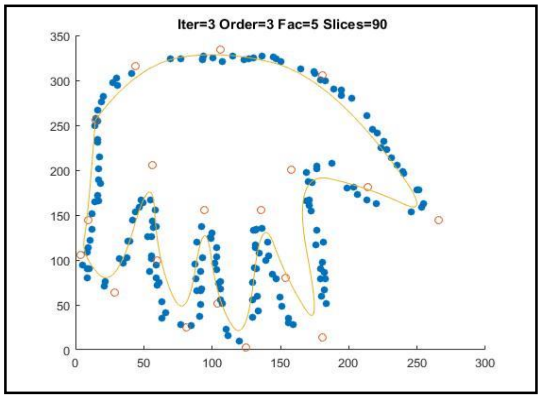


Figure 4.14: Convergence after 3 iterations

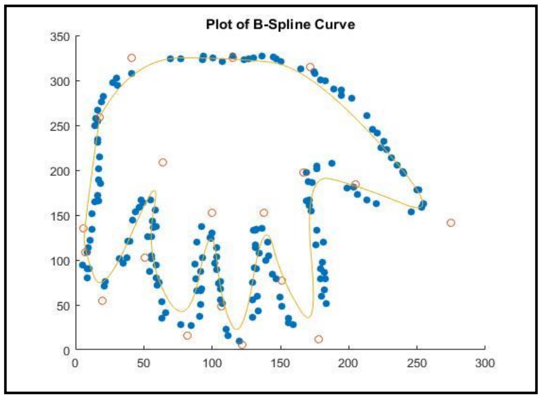


Figure 4.15: After 6 iterations - SDM formulation converges B-spline towards the data cloud

TABLE 6: PARAMETER SPECIFICATION

Parameters	Value
Data cloud	174
Control points	19
Curve segments	90
Weightage for Regularization term, λ	0.005
Iterations	6

Figure 4.14 project's result after 3rd iteration during the simulation while Figure 4.15 shows converged B-spline after 6 iterations. The control points are defined manually in an effective way due to the complex nature of the data cloud which helps gain convergence in 6 iterations with a predefined convergence percentage of 0.5%. We can also observe, that due to continuity issues we have C^0 continuity at start and end point of curve. Also, the data cloud is sparse in the region of fingertip which generates small errors and hence the curve segments do not accurately fit fingertips. On contrary, due to adequate data points in the base region the curve fitting has better approximation.

4.5 CASE 4: SHARP EDGES

The data cloud as shown in Figure 4.16 has 140 data points, to which initial control points (P_x, P_y) are defined. The 14 curve segments are further divided into 30 slices to improve accuracy and reduce the number of iterations. Weightage of 0.001 is assigned to the regularization factor for smoothness of the curve segments. TABLE 7 displays various parameter and their values involved in this experiment.

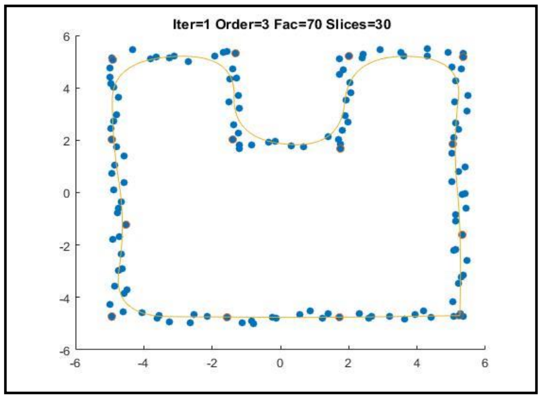


Figure 4.16: Irregular data cloud with sharp change in shape

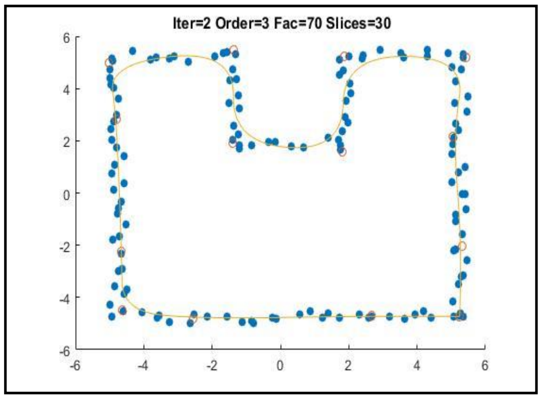


Figure 4.17: Convergence after 2 iterations

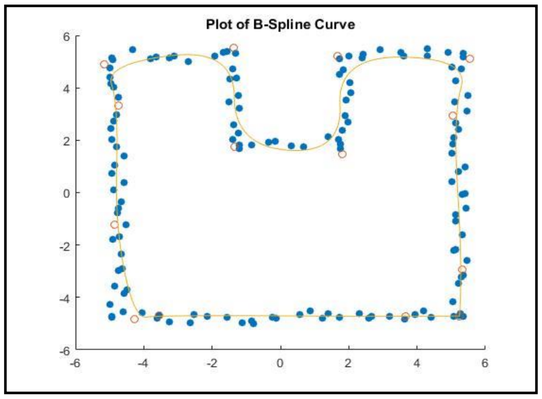


Figure 4.18: After 3 iterations - SDM formulation converges B-spline towards the data cloud

TABLE 7: PARAMETER SPECIFICATION

Parameters	Value
Data cloud	140
Control points	14
Curve segments	30
Weightage for Regularization term, λ	0.001
Iterations	3

Figure 4.17 project's result after 2nd iteration during the simulation while Figure 4.18 shows converged B-spline after 3 iterations. The control points are defined automatically in an effective way to fit the sharp edges of the data cloud, so the simulation takes 3 iterations to converge with predefined convergence percentage of 0.5%. The curve segments have C^n continuity and so cannot fit through corners, as in case of C^0 continuity. The curve fits, edges and corners with fair approximation.

4.6 EFFECT OF CHANGE IN PARAMETERS

The parameters such as weightage of regularization terms, number of control points and number of curve slices have a considerable effect on accuracy of fit and time required for convergence. In this section we will analyze effects of varying these parameters.

4.6.1 EFFECT OF INITIAL DEFINITION

Initial definition of the B-spline curve plays a very important role in quality of output and time required to achieve convergence. The experiment carried out below [Refer Figure 4.19 and Figure 4.22] demonstrates the outcome of effectively defined control points and ineffectively defined control points.

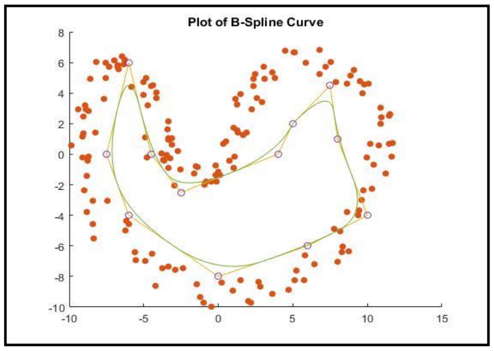


Figure 4.19: Less effective way of defining control points

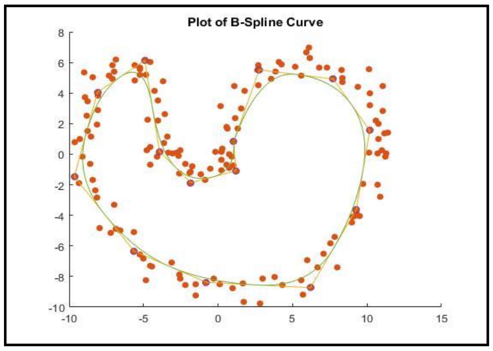


Figure 4.20: Effective way of defining control points

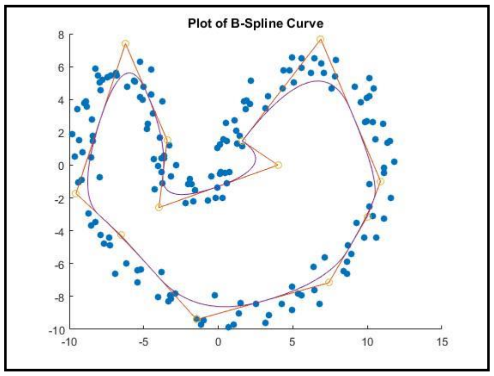


Figure 4.21: Result after 11 iterations in case of less effective way of defining control points

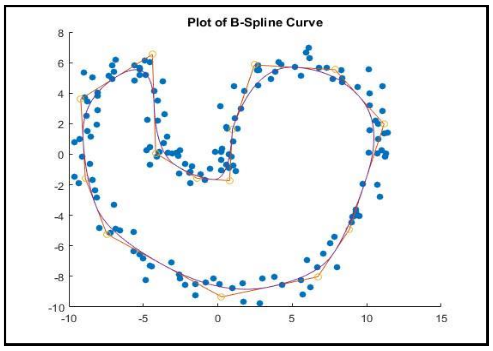


Figure 4.22: Result after 6 iterations in case of effective way of defining control points

Figure 4.19 requires 11 iterations to converge compared to 6 iterations for *Figure 4.20*. Due to the poor definition of control points the results are poor approximation of data cloud. There are less control points in the region on shape change which results into failure to accurately fit curve segments through data cloud. Quality of fit is affected by initial definition of control as shown in Figure 4.21 and Figure 4.22.

4.6.2 EFFECT OF NUMBER OF CURVE SLICES

As SDM formulation on its own requires considerable amount of time, a faster algorithm was implemented. The algorithm compromises on the quality of fit but gives considerable time reduction. To improve the accuracy of the fit, the curve segments are further divided into multiple parts as per the complexity of data cloud. This help in better interpolation of the parameter value for foot point computation. Figure 4.23 - Figure 4.26 demonstrates the experiments carried out on same data cloud with 14, 30, 50 and 120 curve slices.

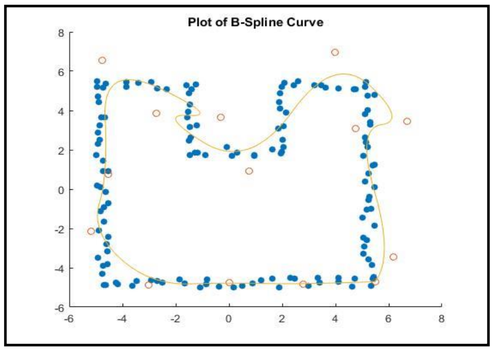


Figure 4.23: Number of curve slices - 14

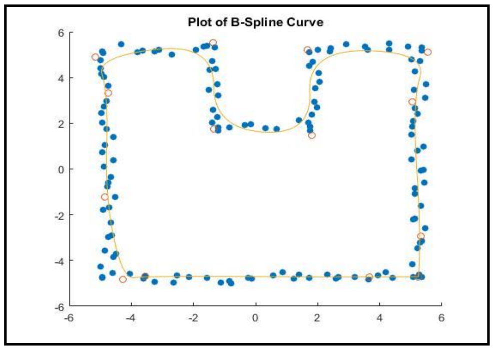


Figure 4.24: Number of curve slices - 30

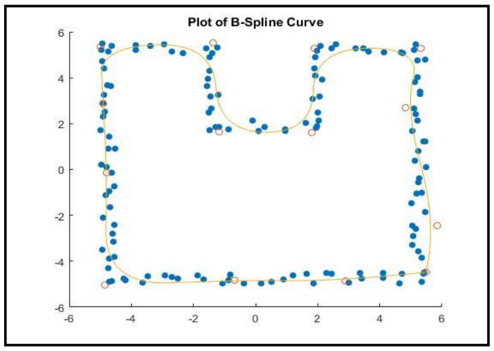


Figure 4.25: Number of curve slices - 50

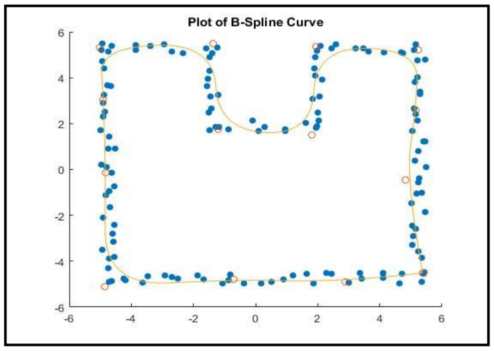


Figure 4.26: Number of curve slices - 120

Form above experiments we can observe that Figure 4.24 gives better approximation of the data cloud as compared to Figure 4.23, while there is no considerable change in Figure 4.25 and Figure 4.26 with increase in number of slices. From this study we understand, that accuracy increases as number of curve slices are increased but after a certain values the change in accuracy is minimal.

4.6.3 EFFECT OF 'λ' VALUE

Regularization term in objective function helps gain smoothness to the curve segments, weightage of this regularization term is govern by λ. Experiments carried out with different λ are shown in Figure 4.27 - Figure 4.30.

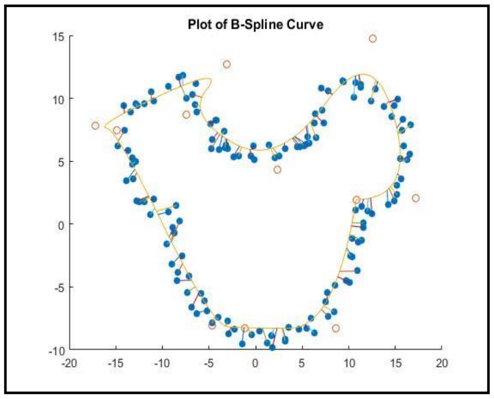


Figure 4.27: λ = 0.5

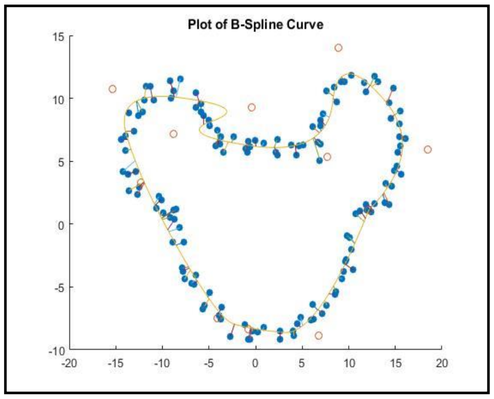


Figure 4.28: λ = 0.05

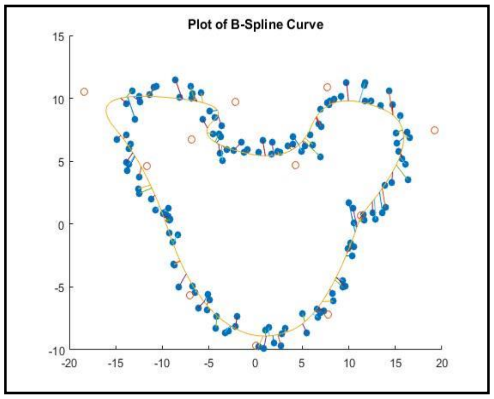


Figure 4.29: λ = 0.01

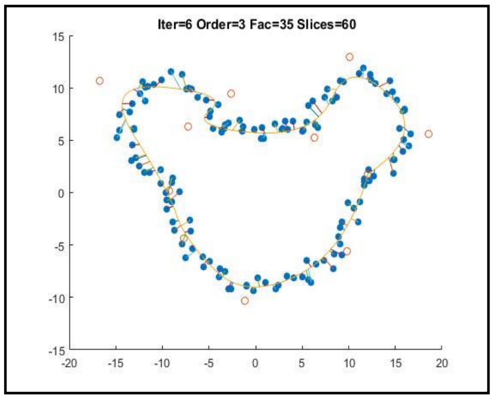


Figure 4.30: λ = 0.005

From the above experiments we can observe, that as the λ values is reduced, the SDM formulation gain better fit to data cloud. Figure 4.30 has better fit compared to Figure 4.29, Figure 4.28 and Figure 4.27.

4.7 OBSERVATIONS AND GUIDELINES

- Better initialization results into quicker convergence.

  Guidelines:
  1) For simple shapes, define control points by automatic method of control point selection. [Refer section 3.2.2.2]
  2) For complex shapes input control points manual. [Refer section 3.2.2.1]

- Higher number of control points does not necessarily result into better fit.

  Guidelines:
  1) For simple shapes use fewer control points. Example: For simple shape like circle 6 control points would yield similar results as 10 control points or for shapes like rectangle 8 control points are sufficient to yield a good fit.
  2) In case of complex geometry, assign control points just before, in the region and just after the sharp change. This gives higher flexibility to fit sudden shape change. While fitting rest of the simple shape with fewer control points. This keeps the total control points to minimum and will require comparatively less time.
  3) In regions, where data cloud closely represents a straight line only 2 control points are required; one near the start and other near the end.
  4) Remember, the aim is to fit curve segments to the data cloud with fewer control points and with accuracy.

- Accuracy can be increased by having optimal number of curve slices. [Refer section 4.6.2]

  Guidelines:
  1) For simple data clouds such as circle or rectangle the number of segments generated by definition of control points is adequate to give good results. This is valid only if the size of simple shapes like circle or rectangle is small, in case of a larger shape and bigger data cloud the number of slices should be increased.
  2) Data cloud with complex and sharp change should have multiple slices. Example: If the size of data cloud ranges between 100 – 250, curve slices can range between 40-100 slices.

- Regularization factor works best with appropriate, λ value. [Refer section 4.6.3]

  Guidelines:

  1) Weightage, λ between 0 and 0.010 for the regularization term yield better results as compared to higher λ values.

5. METHOD LIMITATIONS

In this section, we will discuss the method limitations [also refer: (Wang, Pottmann, & Liu, 2006)] with implementing Squared Distance Minimization method, instability issues and ways to overcome them.

5.1 ABSENCE OF POINTS IN DATA CLOUD:

Sparse data cloud or absence of data points results into instability and undesirable outcome. In this section, we will study effect of sparse data cloud or absence of data points on the final result.

5.1.1 CASE: 1

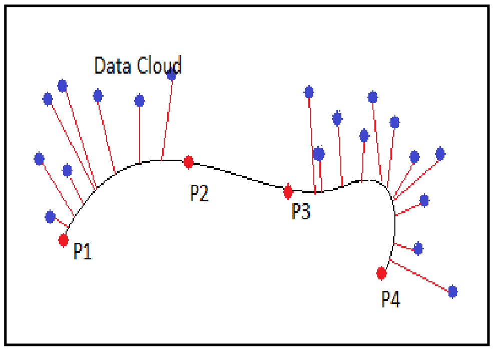


Figure 5.1: Sparse data cloud

In a closed data cloud, a case may arise where there is no foot point on a particular curve segment due to sparse data cloud as shown in Figure 5.1. In such cases, the SDM algorithm cannot calculate the appropriate update for the control point in the region. The result is an inaccurate final curve which is undesirable.

5.1.2 CASE: 2

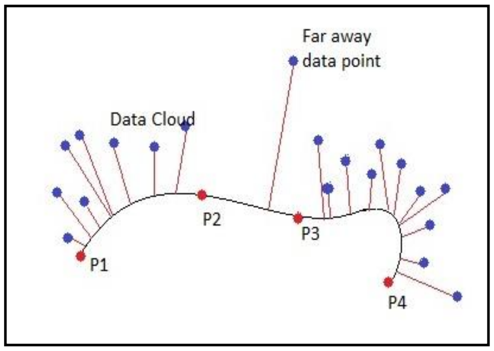


Figure 5.2: Sparse data cloud with far away data point

In a similar case, where the data cloud is sparse and a data point that has foot point on the curve segment is far away as compared to the rest of the cloud points, the error term might become huge and the resultant update will be abnormal as shown in Figure 5.2. This undesirable error term may amplify the error in successive iterations making the curve unstable. To encounter such issues there are two alternatives; (1) a faraway point in a data cloud can be neglected for initial or all iterations or (2) movement of the curve to fit the data cloud can be restricted or controlled. In the latter case, the update is restricted to some ‘x’ unit of maximum change. This is an idea method to maintain stability in the simulation.

5.2 INSTABILITY DUE TO UPDATE

Linear system of equations is evaluated to find SD error term; this error term produces an update to the existing location of control points which converges the curve towards data cloud. These updates yield appropriate relative movements but have an amplified numerical value, which induces instability in the result. By assigning a scaling factor this instability can be controlled. Also, this increases the number of iterations required, but can be considered as a tradeoff between time and instability.

5.3 INSTABILITY DUE TO FOOT POINT CALCULATION

This is a very rare scenario; foot point calculations may sometimes make the fitting B-spline unstable. The calculation for distance between foot point and data point may yield a zero value. Situation like these causes the error term to rise to infinity resulting into undesirable outcome. As these problem exists near convergence they can be encountered by neglecting such data points. In our experiments, we have specified a value of 1*e-^{6} as zero values, such that if distance between foot point and data point is less that 1*e-^{6} the data point will be neglected.

6. CONCLUSION

In this research work, we have studied different types of curve along with various curve fitting error term and have implemented fitting of B-spline to point cloud data using modified SDM method. Based on implementation and experimentations with Squared Distance Minimization Method, we can derive following conclusions.

Ability to gain local control of the fitting curve was the desired characteristic required for your problem. Use of B-spline has successfully fulfilled this requirement and by varying the degree of the curve, the number of control points controlling the curve segments can be varied. This proves advantageous when tracing a geometry with high curvature.

A method was required to fit the initial curve to the desired data cloud accurately, the aim was achieved by implementing Squared Distance Minimization method. The method can be implemented on simple as well as complex data cloud with same accuracy. The regularization term improves the smoothness of fitting B-spline, when a suitable weighing factor is used. The SDM formulation when implemented consumes a lot of time in foot point calculation. The SDM formulation iterates to fit B-spline to the data cloud. The iteration stops when the curves fit the data cloud accurately and successive percentage change drops below an assigned value. The defined value of converges can be increased or decreased depending on the desired accuracy of fit.

Time consumed to calculate foot point was encountered by algorithm mentioned in 3.2.1. The algorithm when implemented brings down the time required considerably. Time required for calculation of foot point is reduced approximately by 82% per iteration. Implementation of Fast Algorithm does bring down the accuracy of approximation but can be improved by increasing number of iteration and/or curve segments, which is a considerable tradeoff when time is a priority.

Stability and implementation issues were resolved as mentioned in section 5. By assigning a scaling factor to the numerical value of the update, the iterations have stable convergence. The results acquired by implementing refined codes were stable and operate with similar accuracy for complex and simple data cloud.

7. FUTURE WORK

The future work that can be undertaken to improve implementation of Squared Distance Minimization method and also to achieve your aim of automating the process of smoothening of surfaces from the noisy optimization results is discussed in this section.

In this research work, to fit the data cloud with curve segments we have used B-spline curves which gives local control as compared to the global control in case of Bezier or Hermite curve. This proves to be an advantage while fitting through complex geometry but this capability can be enhanced by incorporating Non Uniform Rational B-Spline (NURBS) curve (Mortenson, 2006). Use of NURBS curve will help assign weightage to control points. By assigning weightage to control points in the region of sharp or complex geometry, we will be able to fit curve segments with higher accuracy and achieve convergence much faster.

An average data cloud with 170 points and 11 control points (as shown in Figure 4.5 would require 6 iterations to converge and estimated time of 7000 seconds. When explored, 800 seconds are utilized per iteration, in solving the SDM formulation which in this example accounts for 69% of the simulation time. If this time can be brought down or an alternate method with same accuracy can be used it would be more desirable method. Some research papers suggest a C++ program would be around 500 times faster than a Matlab code (Andrews).

Computation of foot point for each point in data cloud is time consuming step and was tackled in this research work by method mention in section 3.2. The method speeds up the simulation by compromises the accuracy to some extent. By studying and implementing alternative methods like Fast Marching method, with SDM formulation we could compare time required and accuracy to select a better method (J.A.Sethain, 1995).

As discussed in section 5.1, a coarse data cloud with sparse data points will result in instability in simulation and undesirable results. A probable solution would be to manually input some data cloud to help gain accuracy in the result. Other method by which this can be avoided is by aiming for denser point cloud data while taking a cross-section of the noisy surface.

The number of control points are user defined and hence fixed. The number of control points does affect the accuracy of fit. Automating change in number of control points will be very advantageous, as it will remove control points where there are more than required and add control points in areas where they are required.

The simulation stops when a set value of convergence is achieved. This convergence does not define the quality of fit with which the curve has fit the data cloud. There is a desperate need to find a metric to check quality of fit.

As discussed in section 1, the big picture of this work is to automate the process of converting rough surfaces to smooth once to improve its manufacturability. Merging generation of data cloud, data cloud recognition, fitting B-spline to data cloud using SDM and lofting surfaces will give much clear idea about areas were improvement or change is required and hence compilation for this different sections is required.

8. REFERENCES

1. Aigner, M., & Juttler, B. (2005). Robust Computation of Foot Points on Implicitly Defined Curves. Brentwood, Tennessee: Nashboro Press.
2. Blattman, W. R. (2008). Generating CAD Parametric Features Based onTopology Optimization Results. *Brigham Young University*.
3. Cheng, K.-S. D., Wang, W., Qin, H., Wong, K.-Y. K., Yang, H., & Liu, Y. (n.d.). Fitting Subdivision Surfaces to Unorganized Point Data Using SDM. *Pacific Conference on Computer Graphics and Applications.* IEEE.
4. Dimitrov, A., & Golparvar-Fard, M. (2014). Robust NURBS Surface Fitting from Unorganized 3D Point Clouds for Infrastructure As-Built Modeling.
5. J.A.Sethain. (1995). A Fast Marching Level Set Method for Monotonically Advancing Fronts. *Proceedings of the National Academy of Science* .
6. Mortenson, M. E. (2006). *Geometric Modelling, Third Edition.* New York: Indusrtial Press Inc.
7. Pekelny, Y. (2005). Implementation of fitting B-Spline curves to Point Clouds by Squared Distance Minimization.
8. Pottmann, H., & Hofer, M. (2002, january). Geometry of the Squared Distance Function to Curves and Surfaces.
9. Pottmann, H., Leopoldseder, S., & Hofer, M. (n.d.). Approximation with Active B-spline Curves and Surfaces. Vienna, Austria: Institute of Geometry, Vienna University of Technology.
10. Sethain, J. (1999). Level Set Method and Fast Marching Method. *Cambridge University Press*.
11. WANG, W., POTTMANN, H., & LIU, Y. (2006). Fitting B-Spline Curves to Point Clouds by Curvature-Based Squared Distance Minimization.
12. Yang, Z., Deng, J., & Chen, F. (2005). Fitting unorganized point clouds with active implicit B-spline curves. *Springer-Verlag 2005*.
13. Yanga, H., Wanga, W., & Sun, J. (2003). Control point adjustment for B-spline curve approximation. *Elsevier Ltd.*

9. APPENDIX

Implemented MATLAB code is as follows:

9.1 DATA CLOUDS

Five data clouds for which experiments were carried out are as follows:

9.1.1 SHAPE: CIRCLE

9.1.1.1 EXCEL FILE

X - CO-ORDINATE	Y - CO-ORDINATE
10	0
9.935251313	0.802056404
9.74268221	1.583339969
9.427280128	2.32361586
8.997213817	3.003711321
8.463621768	3.606012237
7.840323734	4.114919329
7.143462807	4.517252173
6.39108732	4.802590558
5.602683401	4.96354437
4.798670299	4.995944991
3.999871531	4.89895326
3.226975565	4.675081213
2.5	4.330127019
1.837773122	3.873024809
1.257446259	3.315613291
0.774049572	2.672329131
0.400102782	1.959833049
0.145290913	1.196578321
0.016213459	0.402332844
0.016213459	-0.402332844
0.145290913	-1.196578321

0.400102782	-1.959833049
0.774049572	-2.672329131
1.257446259	-3.315613291
1.837773122	-3.873024809
2.5	-4.330127019
3.226975565	-4.675081213
3.999871531	-4.89895326
4.798670299	-4.995944991
5.602683401	-4.96354437
6.39108732	-4.802590558
7.143462807	-4.517252173
7.840323734	-4.114919329
8.463621768	-3.606012237
8.997213817	-3.003711321
9.427280128	-2.32361586
9.74268221	-1.583339969
9.935251313	-0.802056404
10	-1.22E-15

9.1.1.2 CODE

```
function [pcx,pcy]=cloud_gen_circle()
disp('Entering Cloud_gen');
N=20;
theta1=linspace(0,2*pi,2*N);
r=5;
X=r+(r*cos(theta1));
Y=r*sin(theta1);

title('Proposed Point cloud data');
hold on;
pcx=X;
pcy=Y;
scatter(X,Y,'filled');
disp('Exiting Cloud_gen');
end
```

9.1.2 SHAPE: IRREGULAR SHAPE

9.1.2.1 EXCEL FILE

X - CO-ORDINATE	Y - CO-ORDINATE
0.794039854	1.244973648
1.30435201	1.585876229
0.541498267	1.480367585
1.273632868	1.36137088
0.421931343	1.988165465
1.1870588	3.418809665
1.6383329	2.791405778
1.211235264	3.947865137
2.370257319	3.962625542
1.640501706	3.448612051
2.793761614	4.483090166
3.259936225	3.964568203
3.036638221	4.542581458
3.74159826	4.61660668
3.520332136	5.107369433
4.466836533	5.690387854
3.920317232	5.279747647
4.200588009	5.906147594
5.253803212	5.21819497
5.020153965	5.710064568
6.563483325	5.824207531
5.65209324	5.04433876
7.207723312	5.576015542
6.965056664	5.987056114
7.319454739	5.473606116
8.210947866	5.82446695
7.993034418	5.305304057
9.302577956	4.228870371
9.042755629	4.493854556
8.927326407	4.001763299
9.880703138	4.808437263
9.277070983	4.175429224
9.519647399	4.16086759
10.63067248	2.959955925
10.69319302	2.707363423
10.95616761	2.118103369
9.964574239	2.083065504
11.30053306	1.092705309
10.63233823	1.526072437

10.05877673	1.419487485
11.14550999	0.838230826
10.26159137	0.214701474
10.7201164	0.428429659
11.15496107	-0.147029051
10.79244541	-1.508574153
11.05014732	-1.911798824
9.841663502	-2.546331747
10.37097799	-2.803767201
10.24129378	-2.63890027
10.31716786	-3.808324357
10.06134601	-4.903219185
8.410311631	-5.071477235
8.897404129	-5.167226907
9.050151973	-5.571616199
7.995712425	-5.969117882
8.04493742	-7.139555334
7.786495114	-7.311943711
6.892957859	-7.480262527
6.301511635	-7.190380389
5.558622301	-8.059168248
5.142841395	-8.455023545
4.863332677	-7.672614511
4.594894552	-8.60716572
3.661235876	-7.953615172
3.49810308	-8.307489289
3.296254644	-9.149382316
3.168659865	-8.649562605
2.024686698	-8.69056292
2.145527261	-9.324262256
0.767699344	-9.101485455
1.086780999	-8.943356437
-0.231678402	-8.865686182
0.103898907	-9.09730864
-1.042728492	-9.360176787
-2.266415141	-9.441328232
-2.747436083	-8.878986512
-3.378389893	-8.034463799
-2.924129146	-9.210991191
-4.327495296	-8.680415967
-4.163059486	-8.556731795

-5.291237549	-7.386724877
-5.216699403	-7.01235884
-5.520261187	-7.210683384
-6.51129402	-7.451259846
-6.839664948	-6.273511856
-6.986115426	-5.52106471
-6.412520688	-4.937023535
-7.2937667	-5.633208435
-7.99382623	-4.722818947
-7.47237625	-4.555307278
-8.181673111	-3.651939371
-7.730343667	-3.889762997
-8.239908668	-2.774997438
-8.844027619	-1.814402946
-9.058101055	-2.362562913
-8.824033528	-1.177825405
-9.380771152	-0.386186986
-8.444131214	0.408532385
-9.795271543	-0.183861529
-9.9645513	0.911148916
-9.833786018	0.611189262
-8.661846607	1.312284868
-9.398458586	1.287970584
-9.230935925	2.885717615
-9.623659956	2.619562031
-8.419458669	2.422069391
-8.541236313	4.246329365
-9.007624286	3.372637171
-8.239847426	4.89971574
-8.169679737	5.215026461
-8.598774955	5.288984934
-8.159100767	5.037544443
-8.223128835	5.847898538
-7.288326401	5.892542311
-6.780057174	5.004760388
-6.318857376	5.72228915
-6.367281143	6.113607606
-6.008262786	5.210676403
-6.326311114	5.831077146
-5.376013274	6.037334032
-5.200156145	5.49317545

-5.447387948	5.69199523
-6.058775867	5.485478205
-5.302420573	3.919625312
-4.613011133	3.767777062
-5.351611374	4.080905001
-5.039604703	3.264082473
-4.692510462	2.966543666
-4.611014461	2.828119898
-4.857313077	1.733561496
-4.906093494	1.460231019
-3.857642789	0.841262705
-3.889451796	1.039227621
-3.76388268	1.241966324
-4.545292992	0.193756318
-4.157006692	-0.049477612
-3.621369162	0.417188361
-3.933441157	-0.655276703
-3.621293637	-0.938404925
-3.805594617	-0.936085196
-3.507086572	-0.231693066
-2.985635109	-0.434722864
-3.550793702	-1.276383863
-3.593867021	-1.073466905
-2.522375739	-1.036688736
-1.686405137	-1.014648273
-2.290182591	-2.466554113
-1.824029787	-2.184737602
-2.035727346	-2.007624141
-1.951620884	-1.345765814
-0.708415718	-1.593925832
-1.067456137	-1.073941307
-0.500120631	-0.772465436
0.242217642	-1.53446707
-0.061931449	-1.385382511
0.249192916	-0.524311619
0.609045017	-1.324631191
0.875733951	-1.251158891
0.390581871	0.047388448
1.041193928	-0.247324953
1.058853174	0.800974682
0.349249972	-0.075902082

0.337601823	0.525020879

9.1.2.2 CODE

```
function [pcx,pcy]=cloud_gen()
disp('Entering Cloud_gen');

N=20;
theta1=linspace(pi,0,2*N);
r=5;
x1=r+(r*cos(theta1));
y1=r*sin(theta1);
theta2=linspace(0,-pi,3*N);
R=r*2;
x2=R*cos(theta2);
y2=R*sin(theta2);
theta3=linspace(pi,0,5*N/3);
x3=-(3*r/2)+(r*cos(theta3)/2);
y3=r*sin(theta3);
theta4=linspace(pi,0,1.5*N);
x4=-(r/2)+(r*cos(theta4)/2);
y4=-0.5*r*sin(theta4);
X=[x1,x2, x3,x4];
Y=[y1,y2,y3,y4];
%scatter(X,Y);
% figure;
%plot(X,Y);
hold on;

temp=size(X);
xsize=temp(2);
for i=1:xsize
    xnew(i)=(rand(1,1)*1.5)+X(i);
    ynew(i)=(rand(1,1)*1.5)+Y(i);
end

%removing Redundancies
for i=1:xsize
    for j=i+1:xsize
        if(xnew(i)==xnew(j))&&(ynew(i)==ynew(j))
            xnew(i)=[];
            ynew(i)=[];
            xsize=xsize-1;
        end
    end

end
scatter(xnew,ynew,'filled');
title('Proposed Point cloud data');
hold on;
pcx=xnew;
pcy=ynew;
disp('Exiting Cloud_gen');
```

```
end
```

9.1.3 SHAPE: IRREGULAR SHAPE WITH STRAIGHT LINE

9.1.3.1 EXCEL FILE

X - CO-ORDINATE	Y - CO-ORDINATE
11.72934509	0.839920739
11.15092128	1.240560224
12.28899304	1.9772807
13.83454363	2.038134805
13.98231434	2.194197173
14.91686857	1.221322197
14.28579666	1.5533569
15.70127618	3.075944493
15.13530253	3.083936123
15.75083777	3.822095612
15.01138136	4.057336821
15.11655041	5.371436454
15.2417281	6.110357337
16.71752742	6.600062478
14.99563882	7.683307023
16.25490979	7.86656865
14.25670828	8.039164346
13.918543	9.770746808
14.20274388	10.10185596
14.15297706	10.31558069
13.42680538	11.26037219
12.86978845	11.50610981
11.82781851	10.20232
10.36059307	11.42345015
10.16777551	11.32219941
10.192237	10.93673724
9.673339943	10.6907956
8.606655405	10.39865428
8.933024845	9.823648337
6.778977087	9.094821427
7.460659577	9.795956239
7.648045298	7.994835613

6.92304783	8.133330858
5.986088852	7.350264903
6.036539629	6.973903014
6.487412289	5.603906938
5.17922387	6.651930965
5.190938724	6.550609707
4.182267602	5.218722419
5.045610794	6.752702008
4.646643952	6.728509372
2.132578574	6.089364884
3.4658206	6.022016308
2.629377293	5.140442945
2.269195121	6.845426859
0.834404192	5.862924107
-0.206663811	6.441446856
-1.443277102	5.748120386
-0.213453089	6.092959296
-1.699285084	5.993074637
-2.617351955	6.901675118
-1.85954285	6.027126207
-2.426585069	5.911687404
-4.147918238	5.426429558
-4.613501039	6.665508492
-3.591976992	6.730657717
-3.522024222	7.353933091
-4.298437445	7.391694946
-4.532114779	7.850022838
-5.986395532	9.48406484
-5.327567798	9.053286189
-6.248513854	10.30490086
-6.520041438	9.658135783
-7.904729216	9.683173189
-8.051070388	10.85161092
-9.232065769	11.31354862
-10.18848341	10.23431605
-8.988533856	11.87135588
-11.48325589	10.74304627
-10.44397444	9.990596672
-11.08594458	10.80004865
-12.03174742	8.769833727
-13.00590945	10.09850502

-12.81216294	8.512131169
-13.24093492	7.655957741
-13.61245644	8.508542519
-13.22028391	6.459672845
-13.63005	5.940573214
-13.00601793	4.719776047
-14.37402779	5.510088561
-14.51232815	4.305633807
-12.8017317	4.108303171
-13.76175901	2.936862254
-12.91321357	2.501681749
-11.94264221	1.382453978
-10.75383517	1.988144529
-11.63924602	1.534182319
-9.466951954	1.746389464
-9.299792055	0.977689761
-9.479363809	1.138536342
-9.502458674	0.638603183
-8.156064322	1.102513745
-8.321596033	0.518750061
-9.441937149	-0.956777555
-8.390681118	-0.815589327
-8.118754408	-2.501862932
-7.726483704	-2.268623032
-7.724498772	-4.340107063
-6.75929979	-4.988970795
-7.190365698	-5.42176439
-7.087707872	-4.571833824
-6.928329225	-4.939001983
-5.526630687	-6.431939963
-5.958439294	-7.130814475
-4.397310258	-7.950976761
-4.010246396	-7.18836647
-3.312448292	-6.814987985
-3.441520599	-8.550497933
-1.839138656	-7.719614143
-2.040513649	-9.369101596
-1.189663852	-9.352820859
-0.599914374	-9.150635896
-0.371389907	-8.496295213
0.697395946	-8.413125841

0.743134933	-8.638203482
1.105928002	-8.347433426
3.020430823	-7.96868465
2.501317788	-7.910709896
3.982661883	-8.321217842
4.22622013	-9.176831147
5.910575756	-8.781855323
5.546789137	-7.524716416
7.375927336	-7.394234448
7.912819878	-7.597631606
7.252288888	-6.243384586
7.081205796	-5.692913674
8.257321919	-5.085212461
8.607204424	-5.559363249
10.21651774	-4.223109133
9.43038251	-4.898905951
10.39430661	-3.235488611
9.35418262	-3.662876419
11.40988333	-2.286649735
9.689519059	-2.534290921
10.08136465	-0.204911228
10.84241503	-0.211156077
10.96658148	-0.53235536
11.79529308	0.577130618

9.1.3.2 CODE

```
function [pcx,pcy]=cloud_gen1()
disp('Entering Cloud_gen');
N=6;

theta1=linspace(-1*pi/2,pi,3*N);
theta2=linspace(0,3*pi/2,3*N);
theta3=linspace(-1*pi,0,4*N);

r1=5;
r2=10;

x1=(r1*cos(theta1))+2*r1;
y1=(r1*sin(theta1))+r1;

x2=linspace(5,-5,1.5*N);
y2=5*ones(1,1.5*N);

x3=(r1*cos(theta2))-2*r1;
```

```
y3=(r1*sin(theta2))+r1;

x4=(r2*cos(theta3));
y4=(r2*sin(theta3));

X=[x1,x2, x3,x4];
Y=[y1,y2,y3,y4];

temp=size(X);
xsize=temp(2);
for i=1:xsize
    xnew(i)=(rand(1,1)*2)+X(i);
    ynew(i)=(rand(1,1)*2)+Y(i);

end

%removing Redundancies
for i=1:xsize
    for j=i+1:xsize
        if(xnew(i)==xnew(j))&&(ynew(i)==ynew(j))
            xnew(i)=[];
            ynew(i)=[];
            xsize=xsize-1;
        end
    end

end
scatter(xnew,ynew,'filled');
% hold on;
% pcx=xnew;
% pcy=ynew;
pcx=X;
pcy=Y;
disp('Exiting Cloud_gen');

end
```

9.1.4 SHAPE: IRREGULAR SHAPE WITH DEEP AND HIGH CURVATURE CAVITY

9.1.4.1 EXCEL FILE

X - CO-ORDINATE	Y - CO-ORDINATE
5.076187259	93.3767767
6.903516096	89.47324415
8.622444362	79.75280162
9.481700743	89.13136037
7.820725008	107.7565363
8.109000879	115.5047373

11.44036899	120.8143695
12.0796673	131.2152972
14.44874972	151.7717387
14.19203663	250.2674433
15.72157996	163.4855143
13.31291484	232.8254029
13.1913752	259.953028
14.15200614	173.8169773
16.96948934	204.7497983
16.05345594	265.9636192
16.03986111	186.0358966
18.97337106	234.426823
18.01143136	251.440194
18.38818092	166.7223808
18.92287207	213.0447047
16.37923073	274.803854
19.56029804	188.5281553
18.81128972	283.6087984
22.30799333	71.23230763
22.60945432	77.53692526
26.80231804	294.9674892
30.03932726	295.1637029
30.10975488	302.0244312
32.49884456	99.74773112
34.21719388	94.16344132
34.97006524	105.7467344
38.44076096	122.588913
38.91767642	120.2599844
39.95847756	308.0792465
43.95896057	142.7347028
45.4466269	150.1305303
45.32783208	157.994988
48.57746583	163.6769519
52.8088204	164.1730809
53.0056539	126.3142892
56.6619064	104.5823861
54.93002647	126.7709599
54.24701981	143.2212551
57.01516709	83.52778992
55.42368848	167.5834839
58.54208491	104.0849582

58.37632288	135.1522826
56.00137153	153.9535185
58.91614446	92.81046438
56.86510482	136.9053381
59.37294317	80.21761431
61.87081153	77.58387932
59.76010191	70.0071973
63.84705505	54.47090716
63.47323164	35.15609191
67.3927781	40.41725402
70.06572964	321.0300296
75.75573345	29.7840815
80.49414583	322.4531058
83.41827736	26.96774732
88.24133324	80.45090843
88.20323636	38.19256686
90.18256651	68.12436906
88.40439373	95.21718942
90.83481847	66.97171377
88.64990594	116.4542654
92.6515017	86.92662792
92.40722381	326.2396552
90.74703818	49.98879943
90.21675537	70.43584583
93.10892972	104.0442564
90.11100035	137.9615411
93.0037597	322.3279899
93.69553256	326.5025456
99.30053933	128.0039415
98.6140752	124.4271488
99.5758032	131.4024306
101.351479	96.10078597
100.0240439	325.2076479
102.9896331	75.15137137
104.4836501	113.8913743
104.6723437	52.71531117
104.2019814	106.8395003
105.3840867	70.23428284
106.9381294	91.54369779
104.855348	49.13852025
105.8044952	72.05517823

106.894843	324.8047884
107.9958091	22.5456912
111.7257335	18.32354594
117.2985873	326.8119944
120.5222407	12.70347417
125.9633745	324.9505323
129.787664	324.0403988
130.1675167	131.808694
131.3967948	90.56172332
130.953506	114.9057542
131.0931962	33.71966121
129.8286323	54.29360552
129.4434809	75.50084022
130.3196159	325.3684207
133.2684754	115.1267413
132.0844882	134.0972388
131.4810511	43.50180061
132.3865947	88.33847754
133.2984663	108.4557506
132.7942523	59.68980385
135.6073255	132.7966287
135.1931366	133.9860811
138.5596169	326.0056584
138.107981	122.1358327
141.2969982	97.65121761
140.0590547	105.8110592
143.0269886	324.5250644
143.2597261	79.43431984
144.937005	83.81401621
148.2551183	323.5737384
147.2143007	47.50020631
151.0999767	55.66119048
152.6488756	319.2768231
153.3191187	33.81694268
158.0688589	28.2798926
162.8001358	25.63283914
164.1457835	314.7485165
167.5645993	161.0483466
170.8853077	168.7153805
170.9284154	189.9993921
170.6292784	195.9579668

172.0834019	165.8763094
173.3695516	185.6517316
174.2649411	154.1755104
172.8764208	310.2380999
174.750317	308.9474089
176.0797355	202.3183234
176.8243391	204.3813457
178.0114965	133.9868888
179.4605306	115.2721108
180.8741838	89.39502525
179.1879083	301.6118796
178.4281606	78.89666468
180.4547294	119.1318728
181.2664823	61.24527663
181.3071034	81.77611678
182.4858086	299.0304088
183.1552927	95.89212256
184.3152068	50.28992635
184.9045861	68.11284416
181.2913232	88.00489442
188.3247545	207.6893524
189.3080972	290.2162771
193.1530013	288.8531795
197.283001	283.807626
197.3050916	182.8346843
199.9397058	279.5955822
204.0724977	180.3300533
209.9814952	175.5990039
211.8158113	261.7292128
214.7506102	167.2734157
215.5233901	247.5383179
218.4529145	161.6304778
220.4747989	240.873595
224.0136554	227.6421431
224.8257241	229.3544147
229.2965025	223.9477298
231.0488759	215.3183739
236.5133108	202.2438009
237.7635059	196.337032
241.2529505	197.1572423
246.0722316	150.9323611

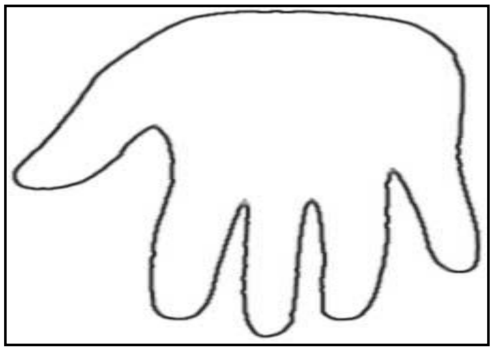
248.349447	179.8358955
252.4439261	180.6433444
253.4155167	156.3887452
255.0148724	165.8101126

9.1.4.2 CODE

```
function [pcx,pcy]=cloud_gen_hand()
disp('Entering Cloud_gen');
 I=imread('hand.png');
 I=rgb2gray(I);
 the_edge = edge(I);
 [y, x] = find(the_edge);
 temp=size(x);
xsize=temp(1);
count=1;
for i=1:18:xsize
    xnew(count)=(rand(1,1)*4)+x(i);
    ynew(count)=(rand(1,1)*4)+y(i);
    count=count+1;
end
pcx=xnew;
pcy=ynew;
scatter(pcx,pcy)
end
```

9.1.4.3 IMAGE

The image used in above code is:



9.1.5 SHAPE: SHARPE EDGES

9.1.5.1 EXCEL FILE

X - CO-ORDINATE	Y - CO-ORDINATE
5.251920041	-4.693595206
5.064883535	-4.734055415
4.411382376	-4.773053267
4.179472701	-4.516973625
3.930717182	-4.652305024
3.635944375	-4.826552405
3.189529693	-4.721652684
2.664454506	-4.718971977
2.588780955	-4.786772232
2.314686934	-4.634306467
1.731739659	-4.772893814
1.400091501	-4.612222679
1.229204518	-4.784861077
0.864117666	-4.527393256
0.564530092	-4.647214071
-0.117746673	-4.805034671
-0.221789014	-4.770309976
-0.836898972	-4.885656208
-0.789802021	-4.992177654
-1.119868705	-4.960965473
-1.562030429	-4.749894338
-2.132382411	-4.714192137
-2.525112321	-4.664416884
-2.631241709	-4.972011921
-3.24769056	-4.923749681
-3.610879122	-4.782412227
-3.549406504	-4.691304914
-4.05028012	-4.568065889
-4.606323455	-4.545973898
-4.945991653	-4.741501621
-4.928421989	-4.720314714
-4.997710188	-4.271831414
-4.575645387	-3.851934192
-4.506515863	-3.71295069
-4.864289188	-3.570314399
-4.746075585	-2.983057506
-4.618556452	-2.889553158

-4.669201903	-2.327717389
-4.914475991	-1.772100378
-4.704758411	-1.676234384
-4.529040535	-1.223767228
-4.774027145	-0.787047841
-4.733688249	-0.585125433
-4.659967235	-0.333646427
-4.880354697	0.117047953
-4.566556473	0.375802173
-4.943692429	0.739164298
-4.849907799	1.062762392
-4.583318218	1.408710883
-4.804912031	1.731948585
-4.92987232	2.026616821
-4.95659245	2.456077979
-4.871358608	2.734984589
-4.787570794	2.990638112
-4.752466538	3.629065683
-4.878213314	4.013224696
-4.962955212	4.162458955
-4.998302939	4.420683274
-4.999349716	4.749762249
-4.928757973	5.134038
-4.912553967	5.069324486
-4.330186824	5.450528953
-3.789569376	5.110592228
-3.647553201	5.188005558
-3.2566285	5.132436297
-3.113969538	5.218163539
-2.690851259	5.013053554
-1.93006827	5.21529826
-1.55625775	5.381207242
-1.662992336	5.34001932
-1.31369129	5.322564393
-1.390511744	4.738683995
-1.28048356	4.373273421
-1.481234307	4.334353299
-1.238478213	3.719735293
-1.507657108	3.452465863
-1.211569053	3.232326872
-1.370869462	2.573693111

-1.240134852	2.258235984
-1.214488928	1.68325637
-1.400453425	2.02491534
-1.206645374	1.83493313
-0.832069452	1.827630258
-0.3536272	1.940949817
-0.160815889	1.943032733
0.322590888	1.787417538
0.677128145	1.743746391
1.404134106	2.134497354
1.705653515	2.030797591
1.754572531	1.846852155
1.761061653	1.667265865
1.824876424	2.38684553
1.979294257	2.678938495
1.886185268	2.921491412
1.91749622	3.528921241
2.047870691	3.806546469
2.040498085	4.211656142
1.728276426	4.511458189
1.840297323	4.675703472
1.740591401	5.099084851
2.002801785	5.215755591
2.384238992	5.128392282
2.412286732	5.266141536
2.91747376	5.473115077
3.601369781	5.196342288
3.530946135	5.335718398
4.307474207	5.485749819
4.287725702	5.225161906
4.92086478	5.343318907
5.359716376	5.325020376
5.363457275	5.186923833
5.290791042	4.71323167
5.028827181	4.80022744
5.142411863	4.263004391
5.481080516	3.713578787
5.096519908	3.446684121
5.466448948	3.126368251
5.136608354	2.662180436
5.198554421	2.428740544

5.065557354	2.114072083
5.045756584	1.859037617
5.005489546	1.493526743
5.394864929	0.979752352
5.224009857	0.801920471
5.030700721	0.420558236
5.321157617	-0.061780928
5.418528223	-0.031703764
5.423186444	-0.609069238
5.139437806	-0.833587941
5.118465192	-1.073051497
5.310130018	-1.596420651
5.086302251	-2.196205941
5.127631101	-2.156921631
5.455533527	-2.581217599
5.362591178	-3.16091903
5.288026728	-3.215375603
5.201921684	-3.471297608
5.044999407	-4.149874311
5.255704469	-4.624869231
5.362843962	-4.721722126

9.1.5.2 CODE

```
function [pcx,pcy]=cloud_gen_sq()
disp('Entering Cloud_gen');

N=30;
x1=linspace(5,-5,N);
y1=-5*ones(1,N);
x2=-5*ones(1,N);
y2=linspace(-5,5,N);
x3=linspace(-5,-10/6,N/3);
y3=5*ones(1,N/3);
x4=(-10/6)*ones(1,N/3);
y4=linspace(5,10/6,N/3);
x5=linspace(-10/6,10/6,N/3);
y5=(10/6)*ones(1,N/3);
x6=(10/6)*ones(1,N/3);
y6=linspace(10/6,5,N/3);
x7=linspace(10/6,5,N/3);
y7=5*ones(1,N/3);
x8=5*ones(1,N);
y8=linspace(5,-5,N);
X=[x1,x2,x3,x4,x5,x6,x7,x8];
Y=[y1,y2,y3,y4,y5,y6,y7,y8];
```

```
plot(X,Y);
% pcx=0;
% pcy=0;
temp=size(X);
xsize=temp(2);
for i=1:xsize
    xnew(i)=(rand(1,1)*0.5)+X(i);
    ynew(i)=(rand(1,1)*0.5)+Y(i);
end
pcx=xnew;
pcy=ynew;
scatter(pcx,pcy)
end
```

9.2 MAIN.M

```
%Shree
%Main Control Program
clear all
close all
clc

 K=3;                   %Order of the curve of b-spline curve
 red_fac=70;            %Ratio of reduction | scaling factor
 run_num=15;            %Number of iteration runs
 open=0;                %open = 0 curve is closed
                        %open = 1 curve is open
 cur_sli=9;             %Decides number of parts the curve should be divided in
 max_per=0.5;           %Minimum allowable percentage of distance change
 max_mov=1;             %Maximum Units a point can move

% GRAPH LABELS
str_iter1=' Iter=';
str_order1=' Order=';
str_red_fac1=' Fac=';
str_cur_sli1=' Slices=';
str_red_fac2=int2str(red_fac);
str_order2=int2str(K);
str_cur_sli2=int2str(cur_sli);
str_order=strcat(str_order1,str_order2);
str_red_fac=strcat(str_red_fac1,str_red_fac2);
str_cur_sli=strcat(str_cur_sli1,str_cur_sli2);

%Generating point cloud data
[pcx,pcy]=cloud_gen_sq;
[temp,cl_size]=size(pcx);

 %Creating control points for first b-spline curve
 count=0;
 for i=1:10:cl_size

    count=count+1;
    PX(count)=pcx(i);
    PY(count)=pcy(i);


 end

%
% PX= [-5,0,5,5.5,5,2,2,0,-2,-1.6,-5,-5];
% PY= [-5,-5,-5,0,5.5,5.5,2,2,2,5.5,5.5,0];


% assuming PX and PY are row arrays
PX=PX';  % UNCOMMENT WHEN AUTOMATIC C. P
PY=PY';
if open==0
[a,b]=size(PX);
PX(a+1)=PX(1);
PY(a+1)=PY(1);
count=count+1;
end

d_mean1=0;




for j=1:run_num





str_iter2=int2str(j);
tit=strcat(str_iter1,str_iter2,str_order,str_red_fac,str_cur_sli);
title(tit);
disp('Run number');
j
%Plotting Cloud point data
scatter(pcx,pcy,'filled');
hold on;

%Initial B-spline curve
[basis,bspx,bspy,u,knots,basis_pre,piece_u,piece_x,piece_y,piece_mid_x,piece_
mid_y,piece_mid_u,fx,fy]=b_spline(PX,PY,K,cur_sli);
max_u=max(piece_u);

%Find theta,Unit tangential, normal,d and rho for each cloud point data
[N,T,d,rho,t,fpx,fpy,side]=pre_req(fx,fy,pcx,pcy,basis,PX,PY,piece_mid_x,piec
e_mid_y,piece_u,piece_x,piece_y,knots,K);

% for i=1:cl_size
% plot([pcx(i) fpx(i)],[pcy(i) fpy(i)]); % connect point to foot point
% end
% hold on;
% CONVERGENCE
d_mean1
d_mean2=eval(mean(d))
```

```
d_per=abs(d_mean2-d_mean1)*100/d_mean1
d_rec(j)=d_per;

if (d_per<max_per)
    disp('Convergence due to maximum percenatge');
    break

end

if ((d_mean1<d_mean2)&&(j>1))
    disp('Convergence due to increasing distance');
    break
end

%SDM formulation
%function giving change in control points as output
%count=No. of control points on the curve
if open==0
[Dx,Dy]=min_fun(N,T,d,rho,t,fpx,fpy,side,pcx,pcy,count,basis,basis_pre,knots,
K,PX,PY,max_u);
end
% REMOVED OPEN CURVE MIN FUN



%1st pt is last pt
Dx(count)=Dx(1);
Dy(count)=Dy(1);
for i=1:count
     if (Dx(i)<=(-max_mov*red_fac))
         Dx(i)= -max_mov*red_fac;
     end
     if (Dx(i)>=(max_mov*red_fac))
         Dx(i)= max_mov*red_fac;
     end
     if (Dy(i)<=(-max_mov*red_fac))
         Dy(i)= -max_mov*red_fac;
     end
     if (Dy(i)>=(max_mov*red_fac))
         Dy(i)= max_mov*red_fac;
     end
end

% scaling
PX=PX+(Dx/red_fac);
PY=PY+(Dy/red_fac);

d_mean1=d_mean2;


title(tit);
figure;

end
% centroid
X_cen=eval(mean(bspx));
```

```
X_cen=num2str(X_cen);
Y_cen=eval(mean(bspy));
Y_cen=num2str(Y_cen);
% plot(X_cen,Y_cen);

cent=['The Centroid Coordinates are X=',X_cen,'Y=',Y_cen];
%plot Convergence
num=linspace(1,j,j);
figure;
d_rec(1)=d_rec(2);
plot(num,d_rec);
```

9.3 B-SPLINE.M

```
% close all
% clear all
% clc
function
[basis,bspx,bspy,u_mem,t,basis_pre,piece_u,piece_x,piece_y,piece_mid_x,piece_
mid_y,piece_mid_u,fx,fy]=b_spline(PX,PY,K,cur_sli)
disp('Entering b_spline');
% plot(PX,PY);
hold on;
%dependancy of n
[a,~]=size(PX);
n=(a-1);
syms Us;
% Knots
for j = 0:(n+K)
    if j < K
         t(j+1) = 0;
    else if (K<=j)&&(j<=n)
             t(j+1) = j-K+1;
    else if j > n
             t(j+1) = n-K+2;
            end
        end
    end
end
% Basics function for K = 1
Ng = zeros(n+1,n+K);
for i = 1:n+1
    for l = 1:n+K
        if i == l
            Ng(i,l)= 1;
        end
        if (i == n+1)&&(l>n+1)
            Ng(i,l)=1;
        end
    end

end
%evaluating non N part
```

```
% Basis function for K = 2
for Kv=2
    for i = 1:n+1
    for l = 1:n+K

        W1 = ((Us-t(i))*(Ng(i,l))/((t(i+Kv-1))-(t(i))));
         if ((W1==Inf)||(isnan(W1)))
            W1 = 0;
         end

       if i==n+1
           c2=0; %thats N(i+1,K-1)of U
       else
           c2=((Ng((i+1),l)));
       end
        W2 = ((t(i+Kv)- Us)*c2/((t(i+Kv))-(t(i+1))));
        if ((W2==Inf)|| (isnan(W2)))
            W2 = 0;
        end


    Ne(i,l) = W1+W2;
   end
        end
end

for Kv=3:K
    for i = 1:n+1
    for l = 1:n+K

        W1 = ((Us-t(i))*(Ne(i,l))/((t(i+Kv-1))-(t(i))));
         if ((W1==Inf)||(isnan(W1)))
            W1 = 0;
         end

       if i==n+1
           c2=0; %thats N(i+1,K-1)of U
       else
           c2=((Ne((i+1),l)));
       end
        W2 = ((t(i+Kv)- Us)*c2/((t(i+Kv))-(t(i+1))));
        if ((W2==Inf)|| (isnan(W2)))
            W2 = 0;
        end


    Nf(i,l) = W1+W2;
   end
    end
    basis_pre=Ne;
    Ne=Nf;

end
```

```
if K>2
for i=1:K-1
    Nf(:,1)=[];
    [a,b]=size(Nf);
    Nf(:,b)=[];
    basis_pre(:,1)=[];
    [a,b]=size(basis_pre);
    basis_pre(:,b)=[];
end
else if K==2
    Nf=Ne;
    end
end

basis_pre(1,:)=0;

fx=(Nf')*PX;
fy=(Nf')*PY;

count=1;
q=1;
edge=(n-K+2)/cur_sli;
num=1;

for u=0:0.01:n-K+2
    count=int16(fix(u))+1;
    if(count>t(n+K))
        count=t(n+K);
    end

    Cx(q)=subs(fx(count),Us,u);
    Cy(q)=subs(fy(count),Us,u);
    u_mem(q)=u;

     % If loop to store edges of curve slices
    if rem(u,edge)==0
        piece_u(num)=u;
        piece_x(num)=Cx(q);
        piece_y(num)=Cy(q);
    % loop to store mid points of all the curve pieces
        if num>1
            mid_u=(piece_u(num)+piece_u(num-1))/2;
            count=int16(fix(mid_u))+1;
            if(count>t(n+K))
                count=t(n+K);
            end
            piece_mid_x(num-1)=subs(fx(count),Us,mid_u);
            piece_mid_y(num-1)=subs(fy(count),Us,mid_u);
            piece_mid_u(num-1)=mid_u;
        end
        num=num+1;
    end
    q=q+1;
```

```
end
scatter(PX,PY);
hold on;
plot(Cx,Cy);
hold on;
title('Plot of B-Spline Curve');
bspx=Cx;
bspy=Cy;
basis=Nf;
disp('Exiting b_spline');
end
```

9.4 PRE_REC.M

```
function
[N,T,d,rho,fpu,fpx,fpy,side]=pre_req(fx,fy,pcx,pcy,basis,PX,PY,piece_mid_x,pi
ece_mid_y,piece_u,piece_x,piece_y,knots,K)


syms Us
disp('Entering new pre_req');
[~,pc_len]=size(pcx);              % length of point cloud data

% calacualtion of normal and tangent
dbasis=transpose(diff(basis,Us));
dfx=((dbasis)*PX);
size(dfx);
dfy=((dbasis)*PY);
size(dfy);
[a,~]=size(PX);
n=(a-1);
zero_margin=0.000001;

tic;
%
%Fast Algorithm
for i=1:pc_len
    %Finding the curve piece associated with each cloud point
    piece_num=find_piece(piece_mid_x,piece_mid_y,pcx(i),pcy(i));

    %Finding the distance of point from each end-point of the curve
    dist_a=sqrt(((pcx(i)-piece_x(piece_num))^2)+((pcy(i)-
piece_y(piece_num))^2));
    dist_b=sqrt(((pcx(i)-piece_x(piece_num+1))^2)+((pcy(i)-
piece_y(piece_num+1))^2));

    %Finding the 'u' value of foot point

fpu(i)=((dist_b*piece_u(piece_num))+(dist_a*piece_u(piece_num+1)))/(dist_a+di
st_b);

    %Finding the x-y coordinate of the foot point
    [fpx(i),fpy(i)]=(xy_u(fpu(i),fx,fy,n,K,knots));
    fpx(i)=eval(fpx(i));
    fpy(i)=eval(fpy(i));

    %Finding the distance between foot point and cloud point
    d(i)=sqrt(((fpx(i)-pcx(i))^2)+((fpy(i)-pcy(i))^2));
    if d(i)<zero_margin
        d(i)=0;
    end
    d(i)=eval(d(i));
    T(i,:)=tn(dfx,dfy,fpu(i),n,K,knots,1);
    N(i,1)=-T(i,2);
    N(i,2)=T(i,1);

    %

    %Calculating radius of curvature at foot point(rho)

   [prev_x,prev_y]=xy_u(fpu(i)-0.001,fx,fy,n,K,knots);
   [post_x,post_y]=xy_u(fpu(i)+0.001,fx,fy,n,K,knots);

    mr=(fpy(i)-prev_y)/(fpx(i)-prev_x);
    mt=(post_y-fpy(i))/(post_x-fpx(i));

   % cal. center of curvature
    x_cen=((mr*mt*(post_y-prev_y))+(mr*(fpx(i)+post_x))-
(mt*(prev_x+fpx(i))))/(2*(mr-mt));
    y_cen=((-1/mr)*(x_cen-((prev_x+fpx(i))/2)))+((fpy(i)+prev_y)/2);
    rho(i)=sqrt(((fpx(i)-x_cen)^2)+((fpy(i)-y_cen)^2));
    rho(i)=eval(rho(i));


    %Finding the side of the point
    m=T(i,2)/T(i,1);
    side1=y_cen-(m*x_cen)-fpy(i)+(m*fpx(i));
    side1=eval(side1);
    side2=pcy(i)-(m*pcx(i))-fpy(i)+(m*fpx(i));
    side2=eval(side2);
    if ((side1/side2)>0)% For same sides
        side(i)=0;
    else
        side(i)=1;%For opposite sides
    end


end
disp('The time for new pre_req is');
toc




end
```

9.5 MIN_FUN.M

```
%This function will be where matrices will be formed
function
[Dx,Dy]=min_fun(N,T,d,rho,t,fpx,fpy,side,pcx,pcy,cp_count,basis,basis_pre,kno
ts,order,PX,PY,max_u)
disp('Entering min_fun');
syms Us; % symbolic u

%Ax=b - this is b - constants
right_side=zeros(2*(cp_count),1);
f1_side=zeros(2*(cp_count),1);
f2_side=zeros(2*(cp_count),1);
min_mat=zeros(2*(cp_count));
f1_mat=zeros(2*(cp_count));
f2_mat=zeros(2*(cp_count));
[~,pc_size]=size(pcx);
const=0;
lambda=0;
time_sdm=0;
time_f1=0;
time_f2=0;

for i=1:cp_count
    %Basis function for ith control point
    tic;
    Ni=basis(i,:);
    for j=1:cp_count
        %Basis Function for jth control point
        Nj=basis(j,:);
        %________________________________________________________________
        %SQUARE DISTANCE MINIMIZATION MATRIX FORMATION
        for k=1:pc_size
            %To find which column to select for a particular value of t(k)
            count=int16(fix(t(k)))+1;
            if count>max_u    % For foot point that lie on the very end of
curve
                count=count-1;
            end
            %Subsistuting and finding values for Bi and Bj
            Bi=subs(Ni(count),Us,t(k));
            if Bi==0 %time shortening strategy
                continue;
            end
            Bj=subs(Nj(count),Us,t(k));

            %when d and rho lie on the same side of curve
            if (side(k)==0)&&(Bj~=0)
            %7A
            right_side(i)=right_side(i)-(2*Bi*N(k,1)*(((fpx(k)-
pcx(k))*N(k,1))+((fpy(k)-pcy(k))*N(k,2))));
            min_mat(i,j)=min_mat(i,j)+(2*Bi*Bj*N(k,1)*N(k,1));
min_mat(i,j+cp_count)=min_mat(i,j+cp_count)+(2*Bi*Bj*N(k,1)*N(k,2));
            %7B
            right_side(i+cp_count)=right_side(i+cp_count)-
(2*Bi*N(k,2)*(((fpx(k)-pcx(k))*N(k,1))+((fpy(k)-pcy(k))*N(k,2))));

min_mat(i+cp_count,j)=min_mat(i+cp_count,j)+(2*Bi*Bj*N(k,1)*N(k,2));

min_mat(i+cp_count,j+cp_count)=min_mat(i+cp_count,j+cp_count)+(2*Bi*Bj*N(k,2)
*N(k,2));%Fixed issue here
            continue;
            end

            if (side(k)==0)&&(Bj==0)
            %7A
            right_side(i)=right_side(i)-(2*Bi*N(k,1)*(((fpx(k)-
pcx(k))*N(k,1))+((fpy(k)-pcy(k))*N(k,2))));
            %7B
            right_side(i+cp_count)=right_side(i+cp_count)-
(2*Bi*N(k,2)*(((fpx(k)-pcx(k))*N(k,1))+((fpy(k)-pcy(k))*N(k,2))));
            continue;
            end



            %when d and rho lie on the opposite side of curve
            if (side(k)==1)&&(Bj~=0)
            %7C
            right_side(i)=right_side(i)-(((2*d(k))/(d(k)-
rho(k)))*Bi*T(k,1)*(((fpx(k)-pcx(k))*T(k,1))+((fpy(k)-pcy(k))*T(k,2))))-
(2*Bi*N(k,1)*(((fpx(k)-pcx(k))*N(k,1))+((fpy(k)-pcy(k))*N(k,2))));
            min_mat(i,j)=min_mat(i,j)+(((2*d(k))/(d(k)-
rho(k)))*Bi*Bj*((T(k,1))^2))+(2*Bi*Bj*((N(k,1))^2));
            min_mat(i,j+cp_count)=min_mat(i,j+cp_count)+(((2*d(k))/(d(k)-
rho(k)))*Bi*Bj*(T(k,1))*(T(k,2)))+(2*Bi*Bj*(N(k,1))*(N(k,2)));


            %7D
            right_side(i+cp_count)=right_side(i+cp_count)-(((2*d(k))/(d(k)-
rho(k)))*Bi*T(k,2)*(((fpx(k)-pcx(k))*T(k,1))+((fpy(k)-pcy(k))*T(k,2))))-
(2*Bi*N(k,2)*(((fpx(k)-pcx(k))*N(k,1))+((fpy(k)-pcy(k))*N(k,2))));
            min_mat(i+cp_count,j)=min_mat(i+cp_count,j)+(((2*d(k))/(d(k)-
rho(k)))*Bi*Bj*((T(k,1))*(T(k,2))))+(2*Bi*Bj*((N(k,1))*(N(k,2))));

min_mat(i+cp_count,j+cp_count)=min_mat(i+cp_count,j+cp_count)+(((2*d(k))/(d(k
)-rho(k)))*Bi*Bj*((T(k,2))^2))+(2*Bi*Bj*((N(k,2))^2));
            continue;
            end

            if (side(k)==1)&&(Bj==0)
            %7C
            right_side(i)=right_side(i)-(((2*d(k))/(d(k)-
rho(k)))*Bi*T(k,1)*(((fpx(k)-pcx(k))*T(k,1))+((fpy(k)-pcy(k))*T(k,2))))-
(2*Bi*N(k,1)*(((fpx(k)-pcx(k))*N(k,1))+((fpy(k)-pcy(k))*N(k,2))));
            %7D
            right_side(i+cp_count)=right_side(i+cp_count)-(((2*d(k))/(d(k)-
rho(k)))*Bi*T(k,2)*(((fpx(k)-pcx(k))*T(k,1))+((fpy(k)-pcy(k))*T(k,2))))-
(2*Bi*N(k,2)*(((fpx(k)-pcx(k))*N(k,1))+((fpy(k)-pcy(k))*N(k,2))));
            continue;
            end


        end
    end

    time_sdm=time_sdm+toc;



%     %_________________________________________________________________
%     %      F1 REGULARISATION TERM
    tic;
    %Right side of the matrix(9A-1)and (9A-6)
    temp1=0;
    temp2=0;
    for j=1:cp_count-1
        %f1_int - Aij
        A1ij=f1_int(basis_pre,i,j,knots,order);
        temp1=temp1+((PX(j+1)-PX(j))*A1ij/(knots(j+order)-knots(j+1)));
        temp2=temp2+((PY(j+1)-PY(j))*A1ij/(knots(j+order)-knots(j+1)));
    end
    temp1=temp1*((order-1)^2)*2;
    temp2=temp2*((order-1)^2)*2;
    f1_side(i)=f1_side(i)+lambda*temp1;
    f1_side(i+cp_count)=f1_side(i+cp_count)+lambda*temp2;
    if isfinite(temp1)==1
        right_side(i)=right_side(i)+lambda*temp1;
    end

    if isfinite(temp2)==1
        right_side(i+cp_count)=right_side(i+cp_count)+lambda*temp2;
    end

    %LEFT SIDE (9A-2)AND (9A-7)
    A1ij=f1_int(basis_pre,i,1,knots,order);
    temp1=(2*lambda*((order-1)^2)*A1ij/(knots(order+1)-knots(2)));
    f1_mat(i,1)=f1_mat(i,1)+temp1;
    f1_mat(i+cp_count,1+cp_count)=f1_mat(i+cp_count,1+cp_count)+temp1;
    if isfinite(temp1)==1
        min_mat(i,1)=min_mat(i,1)+temp1;
        min_mat(i+cp_count,1+cp_count)=min_mat(i+cp_count,1+cp_count)+temp1;
    end

    %LEFT SIDE (9A-4) AND (9A-9)
    A1ij=f1_int(basis_pre,i,cp_count-1,knots,order);
    temp1=(2*lambda*((order-1)^2)*A1ij/(knots(order)-knots(2)));
    f1_mat(i,cp_count)=f1_mat(i,cp_count)+temp1;
    f1_mat(i+cp_count,2*cp_count)=f1_mat(i+cp_count,2*cp_count)+temp1;
    if isfinite(temp1)==1
        min_mat(i,cp_count)=min_mat(i,cp_count)+temp1;
        min_mat(i+cp_count,2*cp_count)=min_mat(i+cp_count,2*cp_count)+temp1;
    end


    %LEFT SIDE (9A-3) AND (9A-8)
    for j=2:cp_count-1
        A1ij=f1_int(basis_pre,i,j,knots,order);
        Bij=f1_int(basis_pre,i,j-1,knots,order);
        temp1=(Bij/(knots(j+order-1)-knots(j)))-(A1ij/(knots(j+order)-
knots(j+1)));
        temp1=temp1*2*((order-1)^2);
        f1_mat(i,j)=f1_mat(i,j)+lambda*temp1;

f1_mat(i+cp_count,j+cp_count)=f1_mat(i+cp_count,j+cp_count)+lambda*temp1;
        if isfinite(temp1)==1
            min_mat(i,j)=min_mat(i,j)+lambda*temp1;

min_mat(i+cp_count,j+cp_count)=min_mat(i+cp_count,j+cp_count)+lambda*temp1;
        end
    end

    time_f1=time_f1+toc;

%
% %
%___________________________________________________________________________
% % %                     F2 REGULARISATION TERM
% %
tic;
%    RIGHT SIDE TERM
     sumx=0;
     sumy=0;
     for j=1:cp_count-1
         Qix(j)=(order-1)*(PX(j+1)-PX(j))/(knots(order+j)-knots(j+1));
         Qiy(j)=(order-1)*(PY(j+1)-PY(j))/(knots(order+j)-knots(j+1));
     end
     for j=1:cp_count-2
         Rix=(order-2)*(Qix(j+1)-Qix(j))/(knots(order+j)-knots(j+2));
         Riy=(order-2)*(Qiy(j+1)-Qiy(j))/(knots(order+j)-knots(j+2));
         Iij=f2_int(basis,i,j,knots,order);
         sumx=sumx+(Rix*Iij*2*(order-1)*(order-2));
         sumy=sumy+(Riy*Iij*2*(order-1)*(order-2));
     end
     temp1=(2*lambda*(order-1)*(order-2)*sumx);
     f2_side(i)=f2_side(i)+temp1;
     if isfinite(temp1)==1
         right_side(i)=right_side(i)+temp1;
     end
     temp1=(2*lambda*(order-1)*(order-2)*sumy);
     f2_side(i+cp_count)=f2_side(i+cp_count)+temp1;
     if isfinite(temp1)==1
         right_side(i+cp_count)=right_side(i+cp_count)+temp1;
     end
     con=2*((order-1)^2)*((order-2)^2);
     %   LEFT SIDE TERM
     for j=1:cp_count
```

```
            j=j-2;
            aj2=1/((knots(order+j)-knots(j+2))*(knots(j+order+1)-knots(j+2)));
            % f2_int - % Regularization term - Iij
            Iij2=f2_int(basis,i,j,knots,order);
            j=j+1;
            aj1=1/((knots(order+j)-knots(j+2))*(knots(j+order+1)-knots(j+2)));
            cj1=1/((knots(order+j)-knots(j+2))*(knots(j+order)-knots(j+1)));
            Iij1=f2_int(basis,i,j,knots,order);
            j=j+1;
            cj=1/((knots(order+j)-knots(j+2))*(knots(j+order)-knots(j+1)));
            Iij=f2_int(basis,i,j,knots,order);
            temp1=(con*lambda*((aj2*Iij2)-((aj1+cj1)*Iij1)+(cj*Iij)));
            f2_mat(i,j)=f2_mat(i,j)+temp1;
            f2_mat(i,j+cp_count)=f2_mat(i,j+cp_count)+temp1;
            if isfinite(temp1)==1
                min_mat(i,j)=min_mat(i,j)+temp1;
                min_mat(i,j+cp_count)=min_mat(i,j+cp_count)+temp1;
            end
        end

        time_f2=time_f2+toc;

    end



% last point is first point and delete extra rows and columns
% Addition of changes

min_mat(1,:)=min_mat(1,:)+min_mat(cp_count,:);
min_mat(:,1)=min_mat(:,1)+min_mat(:,cp_count);
min_mat(1,1)=min_mat(1,1)+min_mat(cp_count,cp_count);

min_mat(1+cp_count,:)=min_mat(1+cp_count,:)+min_mat(2*cp_count,:);
min_mat(:,1+cp_count)=min_mat(:,1+cp_count)+min_mat(:,2*cp_count);
min_mat(1+cp_count,1+cp_count)=min_mat(1+cp_count,1+cp_count)+min_mat(2*cp_co
unt,2*cp_count);

min_mat(cp_count,:)=[];
min_mat(:,cp_count)=[];
min_mat((2*cp_count)-1,:)=[];
min_mat(:,(2*cp_count)-1)=[];

right_side(1)=right_side(1)+right_side(cp_count);
right_side(1+cp_count)=right_side(1+cp_count)+right_side(2*cp_count);
right_side(cp_count)=[];
right_side(2*cp_count-1)=[];



% display results
min_mat
right_side
tic;
```

```
D=min_mat\right_side;
disp('The time required for Calculaton is');
toc
[temp,~]=size(D);
temp=temp/2;
% Ax=b - seperate Dx, Dy
    for l=1:temp
    Dx(l)=D(l);
    Dy(l)=(D(l+temp));
    end

    Dx=Dx'
    Dy=Dy'
    f1_mat
    f1_side
    f2_mat
    f2_side

disp('Total time for SDM');
time_sdm
disp('Total time for f1');
time_f1
disp('Total time for f2');
time_f2
disp('Exiting min_fun');
end
```

9.6 FIND_PIECE.M

```
function piece_num=find_piece(piece_mid_x,piece_mid_y,pcx,pcy)
% help to form groups
[~,mid_len]=size(piece_mid_x);

for i=1:mid_len
    dist(i)=sqrt(((piece_mid_x(i)-pcx)^2)+((piece_mid_y(i)-pcy)^2));
end
[~,piece_num]=min(dist);



end
```

9.7 XY_U.M

```
function [x,y]=xy_u(u,fx,fy,n,K,t)
% use 'u' to find (x,y) - for radius of curvature, foot point
syms Us
count=int16(fix(u))+1;
if(count>t(n+K))
```

```
    count=t(n+K);
end

x=subs(fx(count),Us,u);
y=subs(fy(count),Us,u);

end
```

9.8 TN.M

```
function T=tn(dfx,dfy,u,n,K,t,choice)
% calculate Vectors - tangents and normal unit vectors
% pre-rec - SDM fromulation
syms Us;
count=int16(fix(u))+1;
if(count>t(n+K))
    count=t(n+K);
end
T(1)=subs(dfx(count),Us,u)/sqrt(((subs(dfx(count),Us,u))^2)+((subs(dfy(count)
,Us,u))^2));
T(2)=subs(dfy(count),Us,u)/sqrt(((subs(dfx(count),Us,u))^2)+((subs(dfy(count)
,Us,u))^2));

% Unit Tangential Vector
if choice==1
T(1)=eval(T(1));
T(2)=eval(T(2));
end

% Unit Normal Vector
if choice==2
T(1)=-T(2);
T(2)=T(1);
end


end
```

9.9 F1_INT.M

```
function a=f1_int(basis,i,j,t,k)
% Regularization term - Aij
    syms Us;
    row_a=0;
    if i>1
    row_a=basis(i-1,:);
    end
    row_b=basis(j,:);
    prod=row_a.*row_b;
    [siz,~]=size(prod);
    suma=0;
    for c=1:siz
        suma=suma+int(prod(c),Us,c-1,c);
    end
    a=suma/(t(i+k-1)-t(i));


    row_a=basis(i,:);
    row_b=basis(j,:);
    prod=row_a.*row_b;
    [siz,~]=size(prod);
    suma=0;
    for c=1:siz
        suma=suma+int(prod(c),Us,c-1,c);
    end
    a=a-(suma/(t(i+k)-t(i+1)));
end
```

9.10 F2_INT.M

```
function fin=f2_int(basis,i,j,t,k)
%% Regularization term - Iij
   syms Us;
   i=i-2;
   ai2=1/((t(i+k)-t(i+2))*(t(i+k+1)-t(i+2)));
   i=i+1;
   ai1=1/((t(i+k)-t(i+2))*(t(i+k+1)-t(i+2)));
   ci1=1/((t(i+k)-t(i+2))*(t(i+k)-t(i+1)));
   i=i+1;
   ci=1/((t(i+k)-t(i+2))*(t(i+k)-t(i+1)));
   row_a=0;
   if i>2
   row_a=basis(i-2,:);
   end
   row_b=0;
   if j>0;
   row_b=basis(j,:);
   end
   prod=row_a.*row_b;
   [siz,~]=size(prod);
   suma=0;
   for c=1:siz
       suma=suma+int(prod(c),Us,c-1,c);
   end
   fin=suma*ai2;

   row_a=0;
   if i>1
   row_a=basis(i-1,:);
   end
   row_b=0;
   if j>0;
   row_b=basis(j,:);
```

```
    end
    prod=row_a.*row_b;
    [siz,~]=size(prod);
    suma=0;
    for c=1:siz
        suma=suma+int(prod(c),Us,c-1,c);
    end
    fin=fin+suma*(ai1+ci1);

    row_a=basis(i,:);
    row_b=0;
    if j>0;
    row_b=basis(j,:);
    end
    prod=row_a.*row_b;
    [siz,~]=size(prod);
    suma=0;
    for c=1:siz
        suma=suma+int(prod(c),Us,c-1,c);
    end
    fin=fin+suma*ci;
end
```