A Thesis Report on

DEVELOPMENT OF MACRO FOR 3D MACHINABLE PARAMETRIC FREEFORM SURFACES

submitted

on partial fullfillment of the requirement for the award of degree

MASTER OF ENGINEERING

in

CAD/CAM & ROBOTICS

Submitted by

VIDUR KOHLI

ROLL NO: 800981022

Under the guidance of

Mr. Ravinder K. Duvedi
Assistant Professor
MED, TU

Mr. A.S. Jawanda
Assistant Professor
MED, TU

MECHANICAL ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

2011

CERTIFICATE

This is to certify that the thesis titled " **DEVELOPMENT OF MACRO FOR 3D MACHINABLE PARAMETRIC FREEFORM SURFACES** ", being submitted by **Mr. Vidur Kohli**, Registration No. **800981022**, on partial fulfilment of the requirements for the award of degree of Master of Engineering (CAD/CAM & ROBOTICS) submitted in Mechanical Engineering Department of Thapar University, Patiala, is a bonafide work carried out by him under our supervision and guidance and no part of this thesis has been submitted to any other University or institute for award of any degree.

Mr. R. K. Duvedi
Assistant Professor, MED
TU, Patiala-147004

18/7/2011
Mr. A. S Jawanda
Assistant Professor, MED
TU, Patiala-147004

Dr. Ajay Batish 7/8/11
Professor and Head, MED
TU, Patiala-147004

Dr. S. K. Mohapatra
Dean of Academic Affairs
TU, Patiala -147004

i

ACKNOWLEDGEMENT

I express my sincere gratitude to my guide **Mr. R. K. Duvedi Assistant Professor , Mechanical Engineering Department , Mr. A. S. Jawanda Assistant Professor, Mechanical Engineering Department and, Dr. Ajay Batish, Professor Head Of Department, Thapar University**, **Patiala,** for their valuable guidance, proper advice and constant encouragement of my work in this thesis.

I do not find enough words with which I can express my feeling of thanks to the entire faculty and staff of **Mechanical Engineering Thapar university, Patiala,** for their help, inspiration And moral support which went a long way successful completion of this thesis.

(VIDUR KOHLI)

ABSTRACT

The work done here is focused on the development of MACRO for the machinable 3D freeform surfaces, which can be produced on a 3 axis NC machine. The tool path is generated using a programming language. The tool path file has its input data from the STL file generated by the MACRO. Tool path file is used by the NC machine to produce the final product. This is achieved by the means of a program which is integrated with the MACRO API. The work involves various aspects such as analysis, creativity, and development. To develop an approach for integration of NC machining into casual wood working thereby, developing strategies to generate optimal tool path to reduce machine time. The work here tends to development of unlimited number of machinable surfaces just by the variation of the 3D curvature. It takes considerable amount of time to model the required surface of the part on which finally machining is to be done to generate the final product. In the present work a attempt has been made to develop three dimensional surfaces of a specific variety of parametric nature. A generalized MACRO has been developed to create the parametric pattern surfaces and to allow the user to develop these types of design features with minimum requirement of expertise in this field. Solid modelling systems are design with an API (Application Programming Interface) which form the canvas for writing MACRO. The API of Solidworks™ solid modeller has been used for this purpose. API provides the entire tool required to write and test this MACRO. The MACRO issue geometric instruction to the solid modeller based on parameter provided by user. The geometric instructions are then used to form a solid model of 3D design. Thus the MACROS has been created with a view of providing simple numeric parameter input which could be given on the user and then developing the complete surface. Thus making the all the required calculations in the background and the modelled surface is finally generated on the computer screen. This reduces the time, as well as the brain application of the user, and further it provides ease to the user. The validation of the parameterized design has been done in SolidWorks™. The freeform surface patterns created and parameterized are checked for machinability on a 3-axis milling lathe. These parts are non-symmetric sculptured surfaces as well as symmetric on a cylindrical base or a plane base. The parametric variation of the freeform surface design patterns are checked for machining on a 3 axis NC machine available.

INDEX

CONTENTS **PAGE NO.**

LIST OF FIGURES

FIGURES **PAGE NO.**

LIST OF TABLES

CHAPTER 1
INTRODUCTION

At the threshold of the new century, technology can offer substantial support to all the activities of the Designers. Particularly in recent years, much research work has been carried out in the field of design/production and new sophisticated technology for manufacturing has been developed. Unhappily, the Designer does not always readily accept new technology. To give an example, investment casting was introduced in the late 1940s, but it was not accepted by the industry until the late fifties. Ten years elapsed before it was used in factories. Consequently, in spite of active research work for end product quality improvement, there are some production steps that seem unaffected by the lapse of time. In particular, there is a step in the production process where, even today, the designer is still unaware of new technology. It has always been believed that a compromise between aesthetic and production technology requirements is difficult to reach, but this has not actually been true for a long while, a statement proved by many examples of industrial design, such as Pininfarina design for the automotive industry. These demonstrate how creativity and imagination can perfectly be harmonized with computer calculations. The instrument enabling this small miracle is named CAD/CAM.

1.1 DEFINITION OF CAD

CAD stands for Computer Aided Design and CAM stands for Computer Aided Manufacturing through a computer. CAD denotes programs that replace the pencil and assist the designer in the representation of his ideas. These programs aid in the achievement of a quick development of the design and allow an immediate control. Moreover, they enable the transmission of the designed geometric patterns to other programs for machining. Initially, this software was mainly used for mechanical design. Later they underwent a true transformation and gave birth to two distinct branches: the first one led to the development of programs named CAE (Computer Aided Engineering), and the second one led to CAID (Computer Aided Industrial Design), both software for model development and study of style. The programs of the first kind are of a more technical nature. CAID programs don't require operations like dimensioning and patterning and give more emphasis to the creative phase. They enable a perfect simulation of the reality and give immediate concreteness to

ideas. Consequently, different kinds of CAD systems have been developed that are devoted to different design and modeling types.

1.2 DESCRIPTION OF TYPES OF MODELING :

1.2.1 Polygon Modeling

This is the most common CAD modeling. All models are created as a combination of small squares and triangle. In 3D computer graphics, polygonal modeling is an approach for modeling objects by representing or approximating their surfaces using polygons. Figure 1.1 shows the basic shapes in polygon modeling.

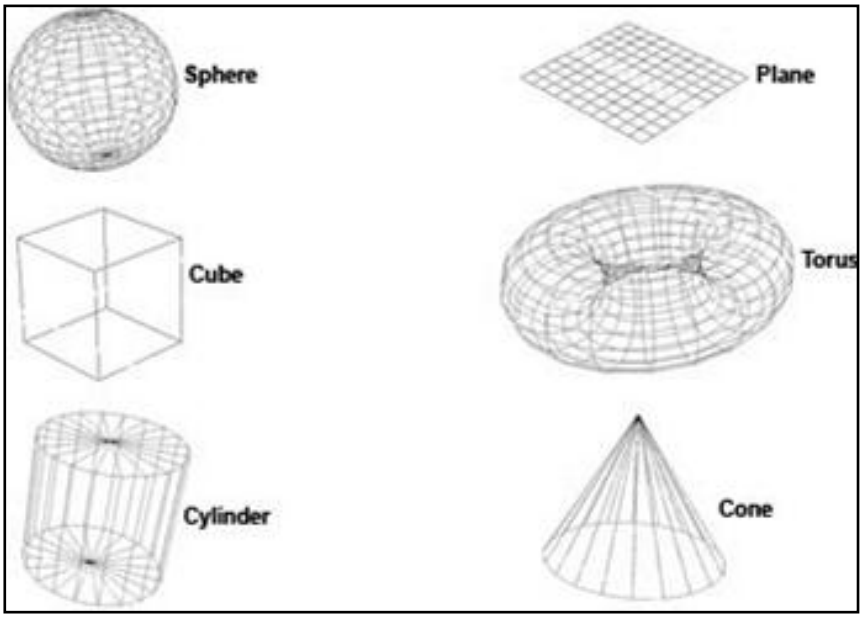


Figure 1.1 : Basic shapes in Polygon Modeling

1.2.2 Surface Modeling

This CAD modeling type shapes the surface, ie. the “skin” of the designed object, and enables the achievement of very complex shapes[1]. The various surfaces used can be B-Spline surface, Bezier surface, NURB, or Freeform surface. Figure 1.2 shows the surface used in surface modeling.

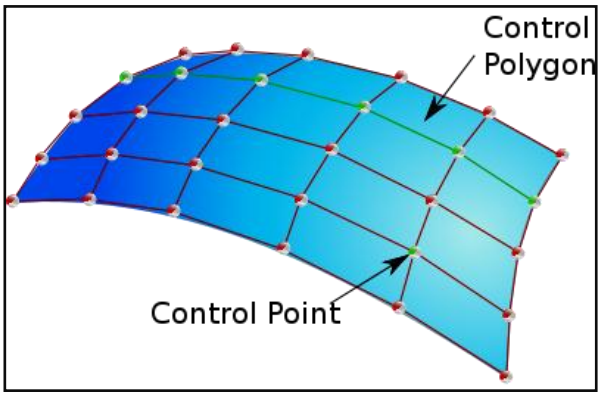


Figure 1.2 : Surface used in Modeling

1.2.3 Solid Parametric Modeling

This software is based on the geometric parameters defining the object. The ability to modify the design whilst keeping some parameters constant is typical of this kind of modeling. Figure 1.3 shows a model constructed using the Autodesk Inventor®.

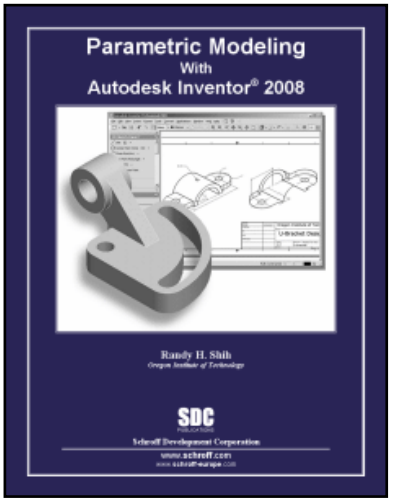


Figure 1.3 : Solid Parametric Modeling using Autodesk Inventor®

1.2.4 Hybrid Modeling

CAD systems that combine surface and solid modeling thus enabling the creation of complex patterns, starting from simple models. These classifications do not have a commercial purpose, like classifying the products from different software houses. Each type denotes a specialization, with a completely different approach to design, and has advantages and drawbacks. A design that is quite easy using one type of software can become very complex if another one is used. Figure 1.4 shows a complex model constructed using the hybrid modeling technique.

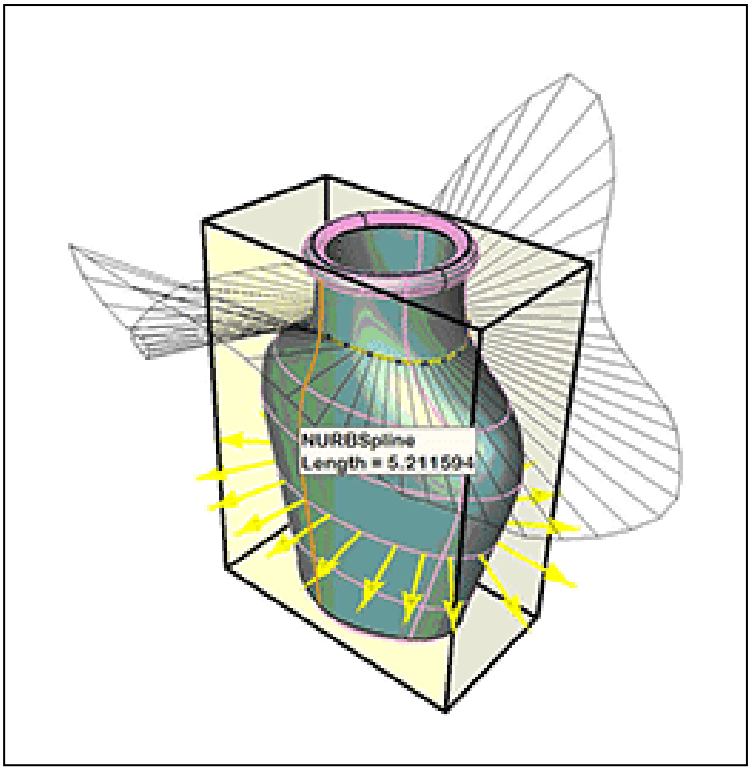


Figure 1.4 : Hybrid Model

The first phase concerns CAD modeling: The designer's idea (usually a sketch drawn on paper) is the starting point. The next stage is the realization of the drawing with the CAD program. Moreover, if we want to produce the same object in a different size, a solid parametric system will enable us to change the size of the object while keeping, for example, the height constant. The mathematical model is analyzed with the same program to verify the surfaces and control any possible errors. Finally, to get a better understanding, the finished object is simulated. This phase is named "rendering" by the technicians, but is not essential for the production of the model. It is used only to check the aesthetic characteristics of an object.

1.2.5 Freeform Surfaces

Freeform surface used in CAD and other computer graphics software to describe the skin of a 3D geometric element. Freeform surfaces do not have rigid radial dimensions, unlike regular surfaces such as planes, cylinders and conic surfaces. They are used to describe forms such as turbine blades, car bodies and boat hulls. Initially developed for the automotive and aerospace industries, freeform surfacing is now widely used in all engineering design disciplines from consumer goods products to ships. Most systems today use non-uniform rational B-spline

(NURBS) mathematics to describe the surface forms; however, there are other methods such as Gorden surfaces or Coons surfaces . Figure 1.5 shows a freeform surface.

The forms of freeform surfaces (and curves) are not stored or defined in CAD software in terms of polynomial equations, but by their poles, degree, and number of patches (segments with spline curves). The degree of a surface determines its mathematical properties, and can be seen as representing the shape by a polynomial with variables to the power of the degree value. For example, a surface with a degree of 1 would be a flat cross section surface. A surface with degree 2 would be curved in one direction, while a degree 3 surface could (but does not necessarily) change once from concave to convex curvature. Some CAD systems use the term order instead of degree. The order of a polynomial is one greater than the degree, and gives the number of coefficients rather than the greatest exponent.

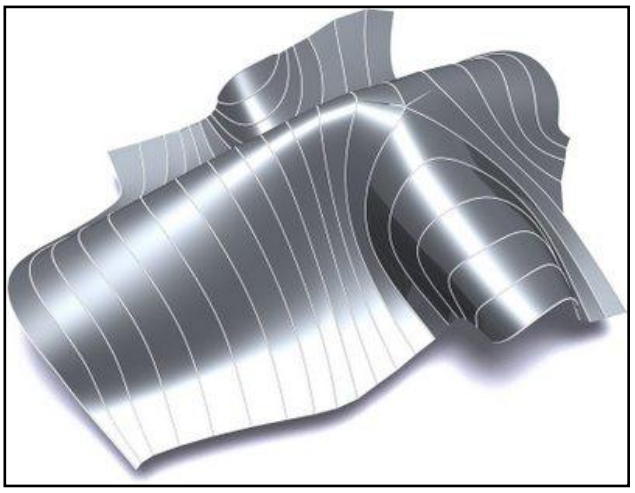


Figure 1.5 : Freeform surface

Freeform surfaces cannot be generated using the various CAD tools. So GL(Graphics Library) is a wide used technique for generating freeform surfaces. An example of freeform surface in shown in the figure 1.6, using the GL.

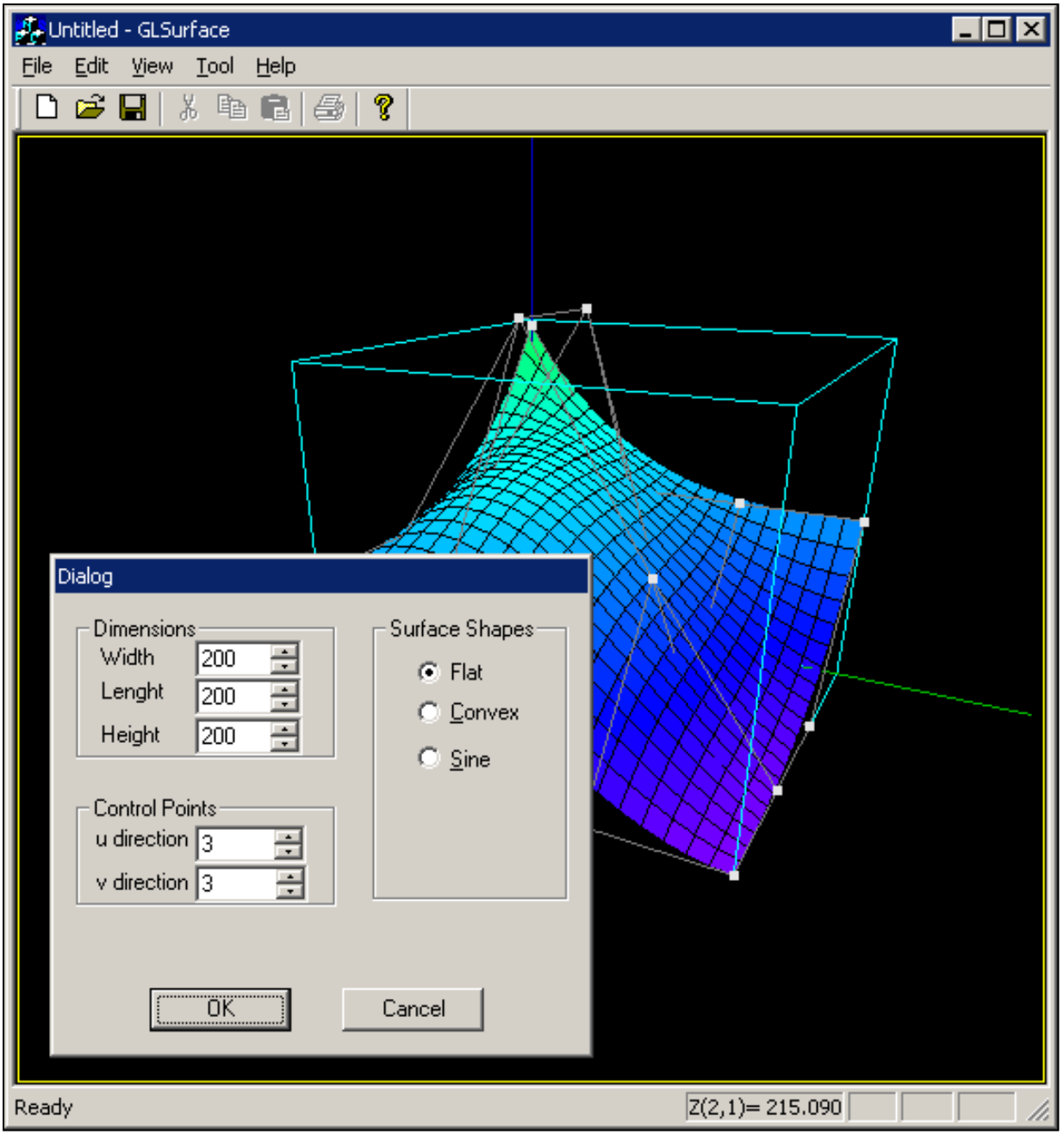


Figure 1.6 : An example of generating freeform surface using GL

1.3 CONTINUITY IN THE SURFACES

The poles (sometimes known as control points) of a surface define its shape. The natural surface edges are defined by the positions of the first and last poles. (Note that a surface can have trimmed boundaries.) The intermediate poles act like magnets drawing the surface in their direction. The surface does not, however, go through these points. The second and third poles as well as defining shape, respectively determine the start and tangent angles and the curvature. In a single patch surface (Bezier), there is one more pole than the degree values of the surface.

Surface patches can be merged into a single NURBS surface; at these points are knot lines. The number of knots will determine the influence of the poles on either side and how smooth the transition is. The smoothness between patches, known as continuity, is often referred to in terms of a C value:

1. C_0 Continuity.
2. C_1 Continuity.
3. C_2 Continuity.

1.3.1 **C_0 Continuity :** Figure 1.7 shows a curve with C_0 continuity. C_0 continuity has the following characteristics :

i. The endpoints of the two curves meet.
ii. The curves have positional continuity only.
iii. There may be a sharp point where they meet.

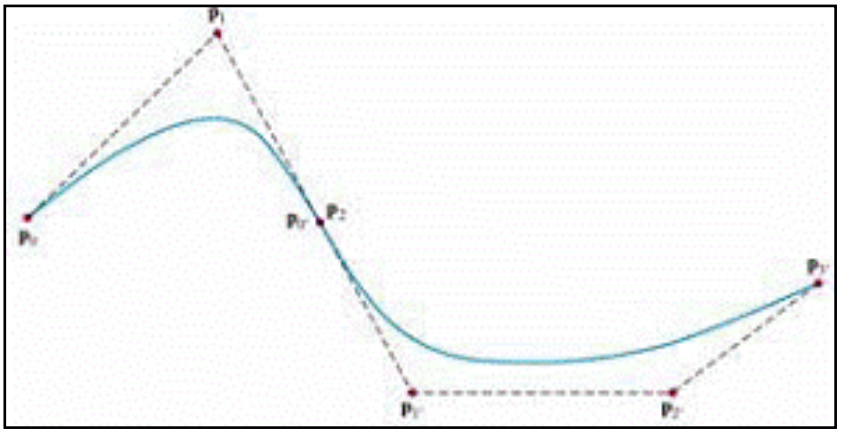


Figure 1.7 : Curve showing C_0 continuity

1.3.2 **C_1 Continuity :** Figure 1.8 shows a curve with C_1 continuity. C_1 continuity [2] has the following characteristics:

i. The curves have identical tangents at the breakpoint.
ii. The tangent is the slope at the breakpoint.
iii. The curves join smoothly.
iv. C_1 curves also have positional continuity.

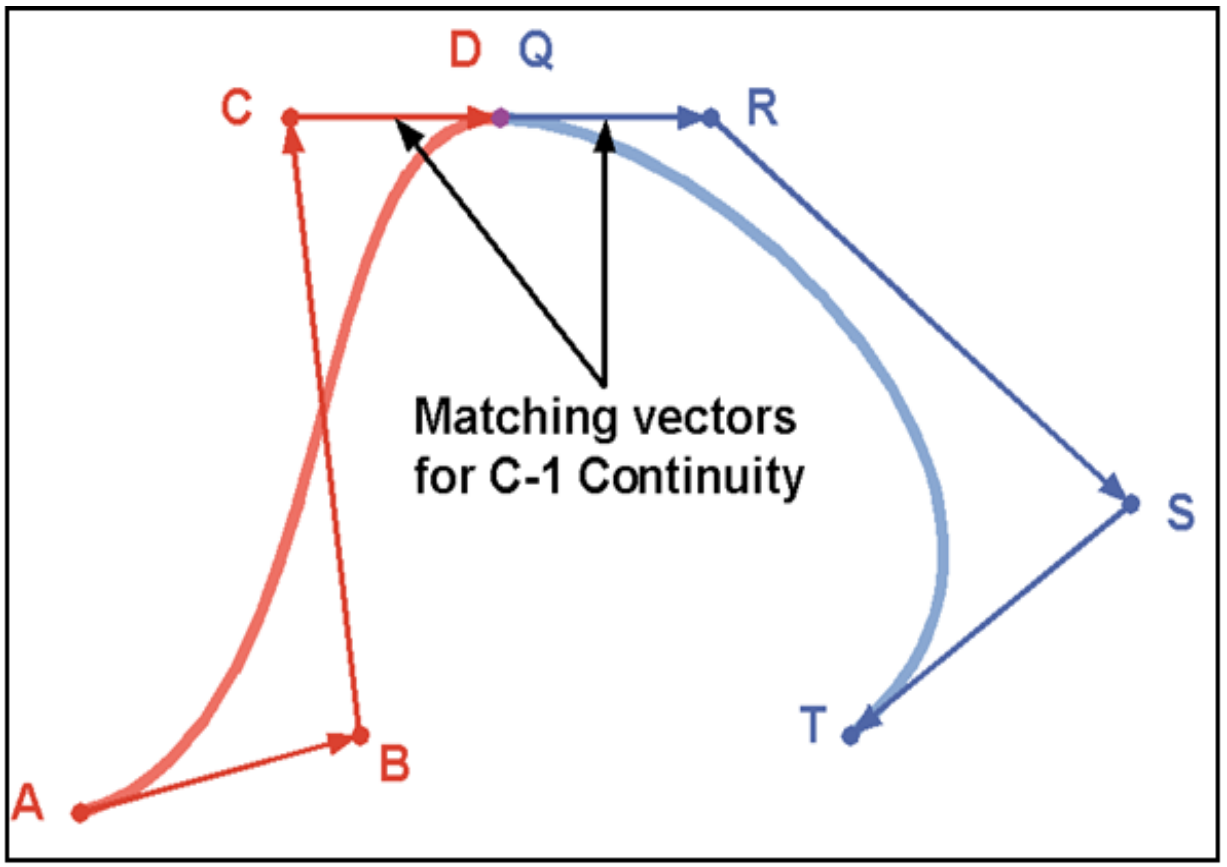


Figure 1.8 : Curve showing C_1 continuity

1.3.3 C_2 Continuity : Figure 1.9 shows a curve with C_2 continuity. C_2 continuity has the following characteristics :

i. The curves have identical curvature at the breakpoint (*Curvature* is defined as the rate of change of the tangents).
ii. Curvature continuity implies both tangential and positional continuity.

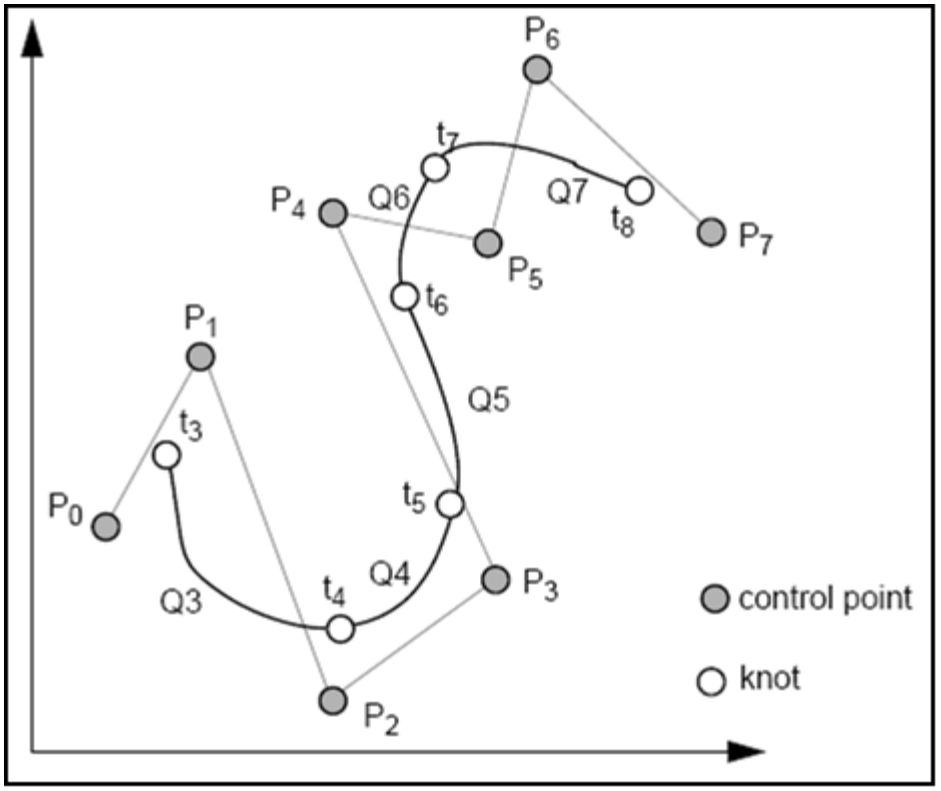


Figure 1.9 : Curve showing C_2 continuity

Two more important aspects are the U and V parameters. These are values on the surface ranging from 0 to 1, used in the mathematical definition of the surface and for defining paths on the surface: for example, a trimmed boundary edge. Note that they are not proportionally spaced along the surface. A curve of constant U or constant V is known as an isoperimetric curve, or U (V) line. In CAD systems, surfaces are often displayed with their poles of constant U or constant V values connected together by lines; these are known as control polygons.

Computing the convex hull [3] of a freeform surface is a challenging task in geometric modeling. Because of the difficulty in computing the exact convex hull of a spline surface, the convex hull of its control points is usually used as a simple, yet rough, approximation to the convex hull. When a tighter bound is needed for the convex hull, we can subdivide the surface into smaller pieces and then union the convex hulls of the control points of these small pieces. This simple approach will require a large number of subdivisions until the approximating convex hull converges to the exact convex hull within a certain reasonable bound.

Considering the convex hull of freeform surfaces. Although non-convex surface is used in the following discussion for clarity, the second surface point S(s,t) as a point on another surface is also considered. Let S(u,v) be a regular C_1-continuous rational surface. Consider the tangent

plane of S at S(u,v) as a moving plane while continuously touching the surface tangentially. Then, any surface point S(u,v)such that the surface is completely contained in one side of the tangent plane is on the boundary of the convex hull of the surface S, as shown in figure 1.10(a). On the other hand, if the tangent plane at S(u,v) intersects the surface transversally at any other surface point S(u,t), then the surface point S(u,v)cannot be on the boundary of the convex hull of the surface S, as shown in figure 1.10(b).

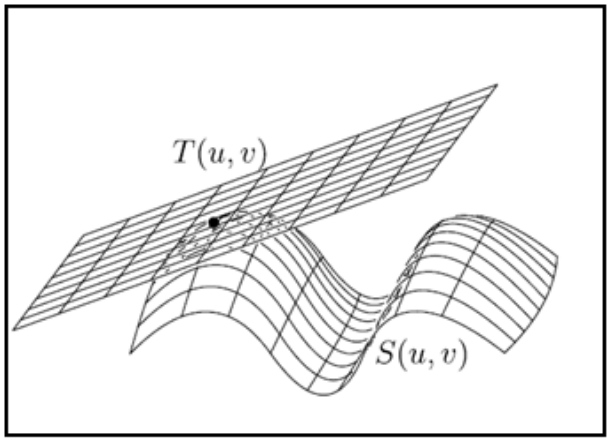


Figure 1.10(a)

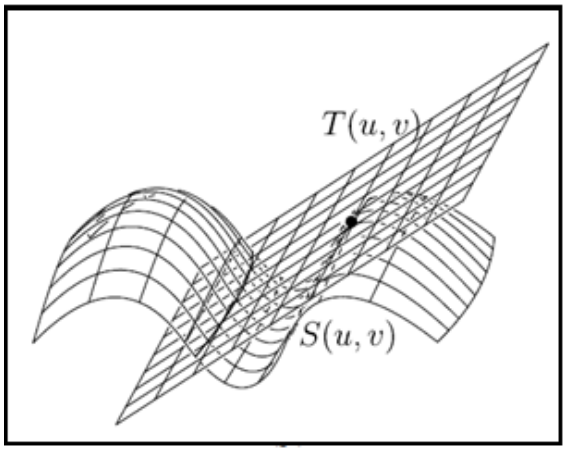


Figure 1.10(b)

Figure 1.10 : Explaining convex hull

However, there can be a case of common bitangent planes because a surface point at which the tangent plane is also tangent to some other surface point could be on the boundary of the convex hull, as shown in figure 1.11, on the next page.

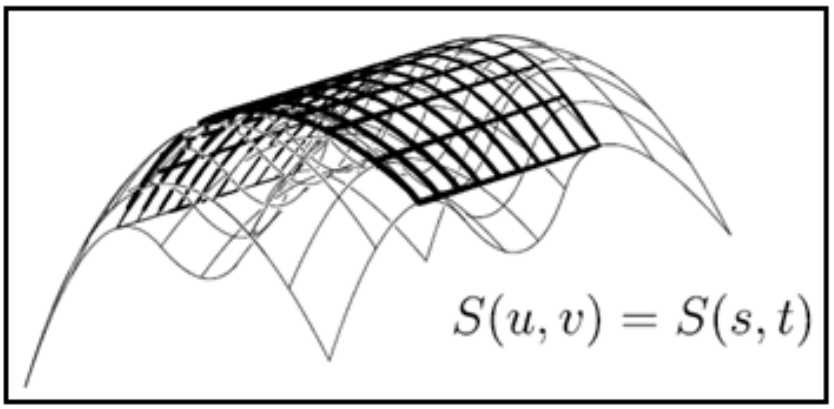


Figure 1.11 (a)

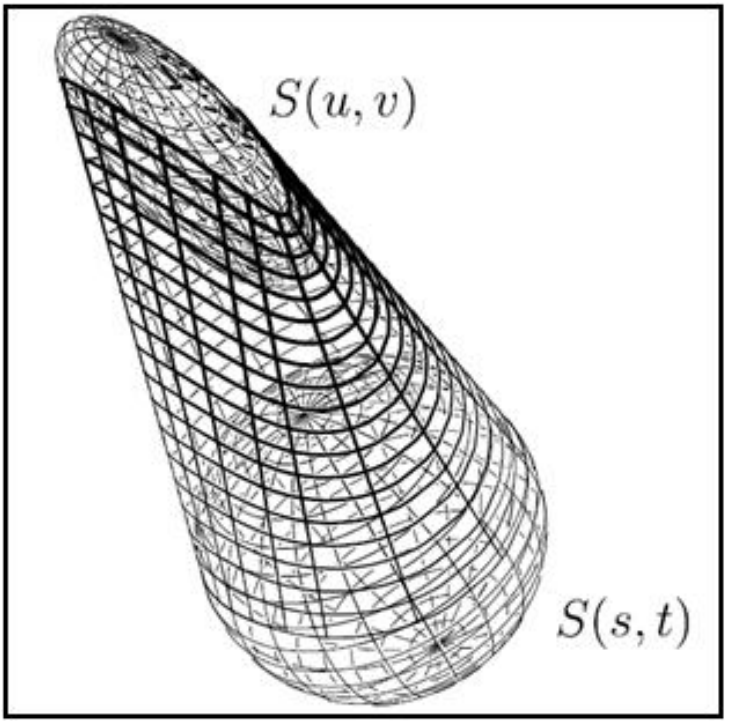


Figure 1.11 (b)

Figure 1.11 (a) and (b): Showing the cases of bitangent planes

1.4 IDENTIFICATION OF THE PROBLEM

The past work is aimed at the development of 3D surfaces without parameterization of the local and global factor. The complete surface is not to be selected as a whole, but only the control points are to be used for the purpose of 3D machining. The surface has 'u'-'v' parameters which vary from 0 to 1. These parameters give the shape of the figure in X direction and Y direction respectively. And the 'w' parameter gives the curve (or tilt) in Z direction. This parameter needs to be controlled for give the surface some curve in Z direction also.

Now, this surface needs to be machined on a flat canvas or a cylindrical canvas. For this purpose various mathematical and analytical calculations are done. So the surface is in the form of a paper that is wrapped around a cylinder or placed on a flat surface. Here the task gets complicated in taking the surface as a sheet and placing it on a curved or plane surface. For this purpose the individual control points are to be manipulated and controlled in such a way that they give the desired output. Furthermore, the surface when wrapped on cylinder or placed on a flat surface still remains 2D only. This is to be converted into 3D by varying 'w' parameter at the various control points. Along with, the shapes on the 2D canvas is also made in such a way that the various CAD transformations can be applied.

The various CAD transformations include:

a. Scaling.
b. Rotation.
c. Translation.
d. Mirror.

Now this 2D shape is to wrapped around the cylinder. After that it is undergone CAD transformation, about certain point or axis. Then the final curve is generated by varying the 'w' value in the parametric equation of the curve. Now this shape is completed, but in visualization only not in actuality. So to make this a reality, the various points are drawn out of the curve having some values in x-direction, y-direction and z-direction. These points are generated by the software tool only, which is used for the modeling of the 3D shape. Now another task is to machine such a complex shape. It is a 3D contoured shape having many concave and convex surfaces. So some proper tooling operation must be performed. For machining purposes, we neglect the convex shapes for the instance.

Now these points generated are then to be used for machining. The motion of tool used must be very precise manner, otherwise the desired shape cannot be obtained. For this purpose, the path of tool is designed. This path enables the tool to move at the desired points do perform whatever operation is required over there.

Now, for achieving all this human calculations are not enough. There is some need for using the automation facilities available. Some software tools are required to carry on with the implementation of the ideas. There are many sort of tools available in the market. The problem that prevails, is - indentifying the tool that will be beneficial in achieving this in the parameters of Complexity, data exchange with the other CAD SYSTEMS.

So, finally the tool which is used in achieving this is:

SolidWorks™: This tool has the best possible API interface as compared to the other tools available. It has the most user friendly interface includes the following characteristics:

I. API using MACRO.
II. Data exchanging tool(STL).

API is the tool which set the basis of this creativity on machining such a complex structured figure on a convex surface or a plane surface. Along with it STL format is used as a data exchanger for the NC tool path planning.

1.5 APPLICATION PROGRAMMING INTERFACE

API stands for APPLICATION PROGRAMMING INTERFACE which enables the user to interact with the model environment with the help of MACRO. Also, with the increasing need for sharing data across applications from various disciplines to provide solutions for real-world problems. In many cases there is also the need to implement specific algorithm to perform dedicated and accurate computation which cannot be found in any commercial software. These are the main motivations to use Application Programming Interface within commercial software. This allows the use of predefined native geometrical entities and operations together with computational algorithms.

The API contains functions that ase used, which are predefined in various programming languages. These functions provide direct access to tools such as SolidWorks™ functionality such as creating a line, inserting an existing part into a part document, or verifying the parameters of a surface.

1.5.1 PROGRAMMING LANGUAGES USED FOR API

There are many programming languages used for the API. Following is the list of a few programming language generally used for API:

i. Visual C++ 6.0
ii. Visual C# .NET
iii. Visual C++ .NET
iv. Pascal
v. Java
vi. Visual Basic 6.
vii. Visual Basic .NET
viii. FORTRAN

Now the need it to use the API interface of mechanical softwares. So that the modeling as well as the programming interface can be dealt with various mechanical-software packages. A few

mechanical-software packages available in market which have the facility of API support are listed on the next page :

i. SolidWorks™
ii. Unigraphics
iii. Proengineer
iv. Catia
v. Mechanical desktop

The package that is taken up in this research work is SolidWorks™, which is discussed further.

1.5.2 SOLIDWORKS™ AND APPLICATION PROGRAMING INTERFACE

Recent SolidWorks™ releases have improved the methods supported by native object and they have been interlaced with a very powerful Mathematical Utility. This feature allows to easily perform basic and advanced point, vector and matrix operations.

Using API into SolidWorks™ we can manipulate three kinds of objects: those coming from SolidWorks™ (model native entities), those coming from math utility database (math entities) and user defined entities as shown in figure 1.12. The native geometrical objects concern the sketch entities (point, line, circle, spline, etc.) and their constraints, the features (extrusion, revolution, loft, etc.), the assembly management (mating, inserting, moving, etc.). The math native objects concern points, vectors and transformations for manipulate entities (projecting from model space to sketch space and vice versa, performing basic operation on vectors, etc.).

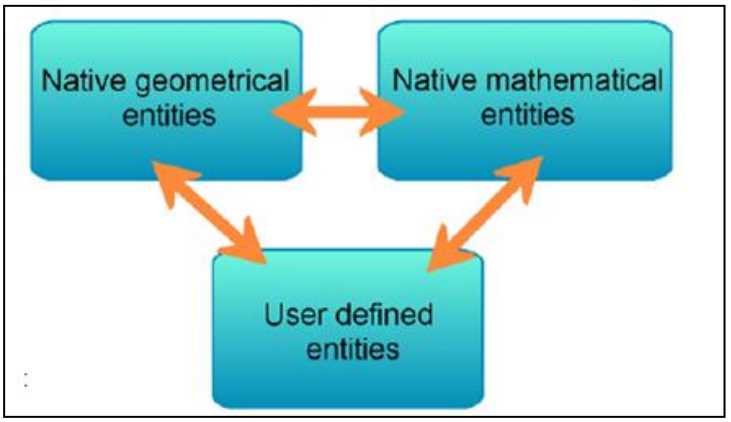


Figure 1.12 : SolidWork™ API entity scheme

Using the software without API the user can only access to single model entity and the direct access to internal database is not permitted [4]. Using API the database of entity can be directly accessed saving time to execute command and model entities can be interlaced with math and user defined ones.

SolidWorks™ provides various application programming interfaces. Following is the list and explanation of the API within the SolidWorks™ :

1.5.2.1 eDrawings API :

The eDrawings application gives you the power to create, view, and share your 3D models and 2D drawings. The eDrawings Application Programming Interface (API) is an OLE(Object Linking and Embedding programming) interface to eDrawings.
Use the eDrawings API to:

i. customize the eDrawings Viewer.
ii. create interactive web pages.
iii. translate files.

1.5.2.2 FeatureWorks API:

The FeatureWorks software recognizes features on an imported solid body in a SolidWorks part document. Recognized features are the same as features that you create using the SolidWorks software. The FeatureWorks Application Programming Interface (API) is an OLE programming interface to FeatureWorks, which allows you to automate FeatureWorks.

1.5.2.3 PDMWorks API:

PDMWorks is a project-data management application designed to run both natively inside of the SolidWorks environment and as a standalone application. The PDMWorks Application Programming Interface (API) is an OLE programming interface to PDMWorks, which allows you to automate and customize PDMWorks.

1.5.2.4 PhotoWorks API:

Using PhotoWorks, you can create photo-realistic images of exceptional quality of SolidWorks models. PhotoWorks:

i. provides many professional rendering effects.
ii. is based on the mental ray rendering engine.

The PhotoWorks Application Programming Interface (API) is an OLE programming interface to PhotoWorks, which allows you to automate and customize PhotoWorks. These functions provide direct access to the PhotoWorks environment.

1.5.2.5 SolidWorks™ Routing API:

SolidWorks™ Routing allows you to create a special type of sub-assembly that builds a path of pipes, tubes, or electrical cables between components.

The SolidWorks™ Routing Application Programming Interface (API) :

i. Allows you to automate and customize only SolidWorks™ electrical routes.
ii. Is an OLE programming interface to only SolidWorks™ Routing's electrical routing functionality

1.5.2.6 SolidWorks™ API:

This online reference guide documents the SolidWorks™ APPLICATION PROGRAMMING INTERFACE (API), which you can use to automate and customize the SolidWorks™ software. These functions provide direct access to SolidWorks™ functionality such as creating a line, inserting an existing part into a part document, or verifying the parameters of a surface.

1.5.2.7 SolidWorks™ Utilities API:

SolidWorks™ Utilities contains tools that let you examine the geometry of a solid model and make comparisons to other models.

1.5.3 Advantages of using MACRO

Using the software without API MACRO the user can only access to single model entity and the direct access to internal database is not permitted . Using API the database of entity can be

directly accessed saving time to execute command and model entities can be interlinked with math and user defined ones. By automating common and repetitive task, it reduces the time consuming in doing a process again and again.

CHAPTER 2

LITERATURE REVIEW

The standard of programming NC machine tools has remained fundamentally unchanged since the early 1950s when the first NC machine was developed at M.I.T. (Massachusetts Institute of Technology), USA. The early NC machines and today's CNCs utilize the same standard for programming, namely G&M codes formalized as the ISO 6893 standard [5]. Starting in the 1970s, significant development has been made towards automatic and reliable CNC machines with new processes such as punching & nibbling, laser cutting, and water jet cutting, which are now commonplace.

The invention of minicomputers, and later microcomputers, has brought a massive improvement in the capabilities of CNC machines with the ability of multi-axis, multi-tool, and multi-process manufacturing. These ever-growing capabilities have made the programming task more and more difficult and complex.

The complexity of required programs to utilize the capabilities of the latest machines requires off-line software tools for CAD, computer aided process planning (CAPP) and CAM for efficient and proven NC code generation.

The Software evolution for NC programming has seen a number of generations, from the beginning of hardwired machines, with manual block to block programming, to automated programming tool (APT), ADAPT [6], AUTOMAP, COMPACT II, and UNIAPT [7] and extensions of APT such as EXAPT, EXAPT II, and EXAPT III [6] to modern graphical interactive CAM systems.

Kaplan [6] they have developed a research in the area of computer-generated geometric art and ornament. Focus on two projects in particular. Develop a collection of tools and methods for producing traditional Islamic star patterns. He has examined the tessellations of M. C. Escher, developing an "Escherization" algorithm that can derive novel Escher-like tessellations of the plane from arbitrary user-supplied shapes. Throughout, showing how modern mathematics, algorithms, and technology can be applied to the study of these ornamental styles.

Today, the software and hardware available at machine tools make it possible to graphically simulate the tool motion and the material removed, and to use adaptive control for on-line improvement of manufacturing process parameters.

There is also trend towards open architectures such as OSACA [7] and OMAC [8], where third party software is being used at the controller, working within a standard PC operating system. One further industrial development is the application of software controllers, where PLC logic is captured in software rather than hardware. Such systems, for example the commercial MDSI CNC architecture [9], provide many opportunities to implement open control capabilities, but they are predominantly used in retrofitting applications for older CNC and conventional NC machines.

Although these developments have improved software tools and the architecture of CNC machine tools, vendors and users are still seeking a common language for CAD, CAPP, CAM, and CNC, to integrate the knowledge generated at each stage. With the current range of proprietary standards for computer aided systems, translating a program written for a specific computer aided systems resource to work on another computer aided systems resource requires tremendous effort. As a result, an enterprise's responsiveness to market changes and its ability to cope with resource relocation or alterations are severely hindered. Since the inception of CAD and CAM software, the problem of a model's portability from system to system was one of the key issues in the use of these tools.

Many solutions have been proposed to standardize the exchange of data. These standards include SET, VDA, and IGES, which have been successful in specific domains but have failed to address the industry-wide CAD/CAM data portability issues [10]. Thus, the international community has developed the ISO10303 set of standards [11], better known as STEP, which have their foundations in many of the earlier aforementioned standards

The Erep (Editable Representation) is the result of a research project carried out at the Purdue University, under the guidance of Prof. Hoffmann. The Erep is a high-level product representation scheme independent of the underlying geometric modeller [12, 13]. Such a representation, the author say, should satisfy the requirements of archivability (capability of being recorded in a machine-readable and easily reusable form), fidelity (capability of complying with the structure created by the designer) and constraints (capability of representing algebric and geometric constraints). The declared aim of Erep is to completely and unambiguosly specify the geometry withuout referencing any particular method through which geometry is implemented. . The authors also discuss the problem of how to support graphical pick operations that the designer issues to generate a feature [14]. This problem, which involves the defintion of a mapping between the high level entities of the Erep and the low level entities of the underlying geometry (Boundary-representation), is strictly related to the persistent naming, which plays a central role in the topic discussed in this paper.

The Erep project represents a milestone in all research activities related to the defintion of a procedural CAD model, since the theoretical aspects of it are well addressed and explained. Unfortunately, to the best of our knowledge, it is not detailed how the proposed approach can be implemented on the top of the existing modeling kernels, and there is a lack of experimental activities on common mechanical parts. As a consequence the applicability of the method to data exchange between commercial CAD packages cannot be evaluated.

The ENGEN (Enabling Next Generation mechanical design) was a research project jointly sponsored by DARPA (Defense Advanced Research Projects Agency) and PDES Inc. The overall aim of the project was the development of a data model capable of exchanging key aspects of design intent information between diverse CAD systems [15]. In particular the ENGEN team worked on the formalization of geometric constraints at sketch level The resulting Engen Data Model (EDM), developed on the STEP/EXPRESS platform, provides a formal specification to handle geometric constraints such as parallelism, perpendicularity, concentricity, tangency, etc. Experimental activity was carried out in order to test the feasibility of the approach for data exchange among different commercial packages. In particular the parametric design of a Ford connecting road was considered as case study. The authors reported successful data exchange among I DEAS, CADDS5 and Pro/Engineer [17]. The results of the projects seem to be confined in 2D sketching, and do not cover the definition of entities for design history. The persistent naming problem was not addressed.

The Working Group 12 (Parametrics Group) of ISO TC184/SC4 has been working for some years with the overall aim of bringing the capabilities of the ISO 10303 (STEP) standard into line with the procedural modeling capabilities of commercial CAD packages [18]. The effort of the Parametrics Group is currently directed on two fronts: the enhancement of the explicit model exchange to include information that is currently lost and the development of an approach for the exchange of procedural models. Regarding the first front a STEP standard has been recently released: the ISO 10303-108:2005 specifies the resource constructs for the representation of model parameters and constraints in CAD or other kinds of models.

A significant step towards the definition of a procedural CAD model is represented by the recently released ISO 10303:55 (Integrated generic resource: Procedural and hybrid representation) [39]. The main purpose of this standard is to provide a means to represent, at a high level of abstraction, models defined in terms of the operations unused to construct them. Though this standard was originally intended to capture and exchange CAD procedural and hybrid models, it can be intended as a general-purpose system for the transfer of any type of procedural model, either geometric or non-geometric.

Gattamelata et al [20] presentation of an application of Visual Basic Application Programming Interface (API) to develop numerical and procedural algorithm into CAD software. The paper focuses on Reverse Engineering embedded into Solidworks. In many RE applications there is the need to remodel the tessellated surface into an editable solid feature, to analyze it and to manipulate it. For this purpose they can be programmed numerical procedures which interact with native geometrical entities in order to improve the modeling capability using automation protocols.

Parametric surfaces are surfaces that are parameterized (usually) by two independent variables.

By doing this it is relatively easy to represent surfaces that are self-intersecting, such as Enneper's surface, and surfaces that are non-orientable, such as the M¨obius strip. Many of these surfaces are impossible to represent by using implicit functions. Even where implicit functions exist for these surfaces, the tessellated representation is often incorrect. This is because the surfaces are self intersecting and the tessellation generated from the implicit function does not reflect the nearness of points on the surface. Historically, these surfaces played a powerful and important role in Differential Geometry. They were often used as tools to determine what properties of surfaces are invariant under various transforms. They were also used as tools for investigating the properties of surfaces in higher-dimensions for example the Klein bottle[21]. Thus they played an important role in the foundation of, and development of Geometry and General Relativity, underlying modern Mathematics and Physics. The surfaces have an inherent beauty and were often named after their discoverers. This class also has the nice property in that it unifies the spline functions found in VTK, since these are parametric by nature. Thus we can derive from this class and provide the equations that generate the surface in the derived class. To show how this is done, a series of classes have been provided. Once we have created the class defining the surface, we need to generate it. To do this, we pass the output from the derived class to an instance.. This will produce the tessellation of the parametric function[22].

2.1 WORKS DONE ON STL

For surface offset method STL file format is used which can be saved easily from most of the CAD modeling software such as PRO-E, CATIA, and SOIDWORKS etc. Since data is in a definite order as shown below the data extraction is easy as compared to other neutral format like STEP, IGES etc.

STL (Stereo lithography) format is rather conceptually simple and sufficiently accessible as it repetitively describes every normal and vertex of triangular facets built for object approximation. Facet data are saved in computers in two types of data format: text (ASCII) format and binary format. The text STL format is a set of facet descriptions in the form of ASCII containing its unit normal vector and 3D coordinates of three vertices. An STL file in text format is obviously redundant for computer storage as it records every character and digit of items. Although the content of a text STL file is readable, its file size is so large that it is generally used as a testing tool. The structure of text STL format is as follows:

solid solid_name

<facet list>

facet normal $NxNyNz$

outer loop

vertex $X1$ $Y1$

$Z1$

vertex $X2$ $Y2$

$Z2$

vertex $X3$ $Y3$

$Z3$

endloop

endfacet

...

endsolid

Where (Nx, Ny, Nz) is a normal vector, and $(X1, Y1, Z1)$, $(X2, Y2, Z2)$ and $(X3, Y3, Z3)$ are

coordinates of vertices. Unlike text STL format, binary STL format is more compact and therefore more efficient for data processing because vectors and coordinates are saved as floating-point numbers, each of which occupies four bytes of computer memory. The binary STL format contains file head, facet number and facet list. STL is widely accepted by most commercial CAD software and rapid prototyping equipment due to the obvious advantage of its topologically simple and robust nature. STL format is composed of only one type of element, a triangular facet, which is defined by its normal and three vertices. All the triangular facets described in a STL format file constitute a triangular mesh to approximate modeling surfaces.

However, drawbacks exist in STL format. Flaws may appear in the process of creating triangular facets, although they can be checked and corrected afterwards. Incorrect normal and inconsistent normal are two cases of inconsistency problems in STL. The former problem happens when the facet normal generated by the CAD system is different from that calculated from facet vertices, and the latter is owing to inconsistent orientation of the normals of adjacent triangular facets. Another kind of flaw is malformation, for instance, once an STL triangular facet is too thin to keep its triangular shape, it may collapse to be a gap, crack or hole. Illegal overlap is the third type of problem. When a facet vertex is located on the edge of another facet or when two facets intersect with each other, the two facets are partly overlapping, which breaks the STL rule that each triangular facet must share two vertices with every adjacent facet.

In addition, STL has some disadvantages in both its format and applications. Redundant depiction of geometric elements in STL format, i.e., each vertex of a triangular facet is recorded at least four times brings extra computational memory occupation and time consumption. Another shortcoming is that STL file size is incommensurate with its approximation accuracy. When the required approximation accuracy and pronounced curvature of an object surface increases in case of complex surface and, the size of the generated STL file is dramatically enlarged. Finally, STL format records only the geometry of object surface and lacks object attributes.

STL format is widely used as a de facto industry standard in the rapid prototyping industry due to its simplicity and robustness. However, on account of its shortcomings and inadequacy in applications, many interface alternatives have been brought forward. Wu et al [23] proposed a new scheme to enhance the approximation accuracy and to extend functions of STL by means of introducing additional feature and attribute codes into STL format. The geometry feature code describes a tetrahedron based on the STL triangular facet, which provides better approximation to the object surface covering. The attribute code attaches attributes of object surfaces such as colours and markers to STL triangular facets. Moreover, the enhanced STL also shares the structure of binary STL format by filling feature and attribute codes into its blanks, and therefore is compatible with STL. Compared with STL, the enhanced STL provides not only higher accuracy with the same file size and compatible format, but also colour and marker functions for rapid prototyping.

2.2 Literatures available on surface modeling :

A shell map [24] is a bijective mapping between shell space (the space between a base surface and its offset) and texture space. It can be used to generate small-scale features on surfaces using a variety of modeling techniques. In this paper, an efficient algorithm which reduces distortion by construction, for the offset surface generation of triangular meshes is given. The basic idea is to independently offset each triangle of the base mesh, and then stitch them up by solving a Poisson equation.

Qu and Stucker [25] presented a new 3D offset method for modifying CAD model data in the STL format. In this method, vertices, instead of facets, are offset. The magnitude and direction of each vertex offset is calculated using the weighted sum of the normals of the facets that are connected to each vertex. To facilitate the vertex offset calculation, topological information is generated from the collection of unordered triangular facets making up the STL file. A straightforward algorithm is used to calculate the vertex offset using the adjoining facet normals, as identified from the topological information. This newly developed technique can successfully generate inward or outward offsets for STL models.

Malosio et al [27] described a new geometric algorithm to offset CAD objects, described as surfaces tessellated with triangular facets, transforming the original geometry in a new smoothed offset model. Different approaches to cope with convex and concave geometries and to prevent overlapping cases are suggested and investigated. Furthermore, an STL-file preprocess algorithm is proposed in order to obtain an errors-free final surface modifying starting tessellated model. The developed algorithm has been named Offset Weighted by Angle (OWA).

Work has been done in parametric and non-parametric tool paths. A brief history has been discussed by Dragomat et.al.[28]. The tool path is defined by two or three parameters which represents the complete surface. Catania[29] gives the estimation of the cutting depth. The computer representation of the designed surface is usually of parametric form; common forms include Bezier, B-helical, and Non-uniform rational B-helical (NURB) surfaces [30, 31, 32].The die or mould is made from the stock material in three steps, namely, rough, finish machining and polishing. In the roughing, the stock material is machined rapidly into an approximated shape. Finish machining is performed on the remained stock to bring it to within a specified surface tolerance of the die. In order, to reduce cost and time for mould and die production, the finish machining and polishing steps should be improved. In industry 3-axis CNC machines are used for both the rough and finish machining steps. Commonly, 3-

axis machining methods for machining complex surfaces use a ball-end mill cutter. An inherent problem in using ball-end mills is a zero cutting speed at the bottom of the ball. This problem can be overcome either by using other cutter geometries, such as radiused-end or fiat-end mills, or by using 4/5-axis CNC machines to avoid cutting with the bottom of the ball. The first approach has difficulty in machining the bottom of cavities and the crests of convex shapes; in addition, a non uniform scallop height is left over the entire surface. The second approach uses expensive machines and achieves little gain in production time. To overcome the above problem, 5-axis machines have been used to machine dies and moulds with flat-end mills. Choi et al. [33] used a 5-axis CNC machine to mill sculptured surfaces using a method called Sturz milling. Vickers et al. [34] used a similar method to machine compound curvature surfaces. In this method, a flat-end 17 mill is inclined in the direction of feed by a fixed angle, defined between the tool axis and the surface normal. Vickers et al.[35] show how the inclination of a fiat-end mill changes the effective cutting radius. In Sturz milling the inclination angle is chosen somewhat arbitrarily and remains constant throughout machining. Furthermore, because machining is done with the edge of the cutter, material is left in the wake of the tool. Cho et al.[36] showed this problem to be a concern as it created surface roughness in the direction of feed. To address the fixed inclination angle Jensen and Anderson [37] used the effective cutting radius of a flat-end mill to vary curvature of the cutter's swept silhouette curve to match surface curvature. This technique adapts the fixed Sturz angle to best fit the local surface topology. The curvature matching was done by projecting a 3D cutting edge onto the osculating plane at the point of contact, and matching it with the surface curvature. To realise the projection requires the cutter to move linearly along a perpendicular to the osculating plane. 5-axis machines in general cannot perform linear interpolation with 5-axis simultaneous motion, thus the projection method is approximate. The significant work has been done by Choi [38] and Saitho [39]. Saito presented the surface machining with G-Buffer method. other work has also been done by Choi [38].

CHAPTER 3

SolidWorks™ MACRO

The quickest and easiest way to start programming with the SolidWorks™ API is to record a SolidWorks™ macro, which contains the SolidWorks™ API calls that correspond to the actions performed in the user interface. You can modify the macro in Microsoft Visual Basic for Applications (VBA) or Microsoft Visual Studio Tools for Applications (VSTA) to fit your work site's needs.

1. Microsoft VBA is a toolset based on Microsoft Visual Basic for Applications (VBA) and is embedded in the SolidWorks™ software. Microsoft VBA lets you record, run, and edit Microsoft VBA macros in the SolidWorks™ software. Recorded macros are saved as .swp files.
2. Microsoft VSTA is a toolset based on Microsoft VB.NET and C# and is embedded in the SolidWorks™ software. Microsoft VSTA lets you record, run, and edit VB.NET and C# code in the SolidWorks™ software. Recorded Microsoft VSTA macros are saved as either Microsoft VB.NET or C# projects. Building or debugging a Microsoft VSTA macro creates an executable file that ends in the filename extension .dll, which can be used by the SolidWorks™ software in the same manner as a .swp file.

   While debugging Microsoft VSTA macros with user-interface components such as PropertyManager pages, manipulators, or other objects that use events or handler objects, the debugger must continue to run after the main() method of the VSTA macro exits. Either deselect the user-interface system option Stop VSTA debugger on macro exit (located on the Tools > Options > System Options dialog) or set swUserPreferenceToggle_e.swStopDebuggingVstaOnExit to false to keep the debugger running after the main() method of the Microsoft VSTA macro exits.

Typically these steps are followed to create a SolidWorks™ API application using SolidWorks™ macros:

1. Plan the user-interface actions before recording them.
2. Record the user-interface actions.
3. Run the macro to test it.

4. Debug the macro and test it again.
5. If the application has a user interface, create it in Microsoft VBA or Microsoft VSTA and then modify the macro to work with the user interface, run the modified macro, and debug it.

When done recording, testing, and debugging the macro, you can assign the macro to button.

3.1 Record SolidWorks MACRO

To record a SolidWorks macro:

1. Click Record\Pause Macro  on the Macro toolbar, or click Tools, Macro, Record.
2. Perform the actions that you want to record.
3. When you are done recording, click Stop Macro  on the Macro toolbar or click Tools, Macro, Stop.
4. In the dialog, type a name for the macro in File name, select the type of macro in Save as type, and click Save.

    1. If SW VBA Macros (*.swp) is selected , then the .swp extension is automatically added to the filename.
    2. If SW VSTA VB Macro (*.vbproj) is selected , then a Visual Basic .NET project is created. Building or debugging the project creates an executable file with an filename extension of .dll.
    3. If SW VSTA C# Macro (*.csproj) is selected , then a C# project is created. Building or debugging the project creates an executable file with an filename extension of .dll.
    4. If SW All Macros Types (*.swp, *.csproj, *.vbproj) is selected , then a VBA macro file is created and VB.NET, and C# projects are created. A .swp extension is automatically added to the VBA macro's filename. Two folders are created for the VB.NET and C# projects.1.
        I. The VB.NET project folder's name is the name of the macro plus -vbnet.
        II. The C# project folder name is the name of the project plus -csharp.

        For example, if the recorded macro is saved as Macro1 and selected SW All Macros Types (*.swp, *.csproj, *.vbproj) in Save as type, then the folder

where the MACRO is saved would contain a file called Macro1.swp and two project folders, Macro1-vbnet and Macro1-csharp. The VB.NET and C# projects reside in their respective folders.

The automatically edit macro after recording system option has no effect when you select to save a just recorded macro as SW All Macros Types (*.swp, *.csproj, *.vbproj).

To pause while recording a macro, click Record\Pause Macro  or Tools, Macro, Pause. Click Record\Pause Macro  again to continue recording.

3.2 Run SolidWorks MACRO:

Opens a dialog that lets you select the macro that you want to run.

To run a macro:

1. Click Run Macro  on the Macro toolbar, or click Tools > Macro > Run.
2. In the dialog, locate the macro (.swp or .dll) file that you want to run and click Open.

3.3 Edit and Debug the MACRO:

To edit or debug a SolidWorks macro:

1. Click Edit Macro  on the Macro toolbar, or click Tools > Macro > Edit.

2. In the dialog, select a macro file and click Open.
   I. To edit a Microsoft VBA macro, select SW VBA Macros (*.swp) in Files of type (selected by default).
   II. To edit a Microsoft VSTA macro, select SW VSTA VB Macro (*.vbproj) or SW VSTA C# Macro (*.csproj).

   NOTE: You can also edit .swb files, which are older-style SolidWorks macro files. When you run or edit a .swb file, it is automatically converted to a .swp file.
3. Edit or debug the macro.

I. Delete extra lines of code automatically inserted in the macro when it was created:
    a. For example, the following variables are declared automatically in a SolidWorks VBA macro. Delete any variables not used in the macro.

```
Dim swApp As Object

Dim Part As Object

Dim boolstatus As Boolean

Dim longstatus As Long, longwarnings As Long

Dim FeatureData As Object

Dim Feature As Object

Dim Component As Object
```

    b. Delete all lines of code that change the view.
    c. Delete all IModelDocExtension::SelectByID2 calls appearing immediately before IModelDoc2::ClearSelection2 calls. However, do not delete IModelDocExtension::SelectByID2 calls appearing immediately after ModelDoc2::ClearSelection2 calls.
    d. Delete all IModelDoc2::ClearSelection2 calls appearing immediately before IModelDocExtension::SelectByID2.
II. Explicitly declare all variables in a Microsoft VBA macro.
III. Early bind all variables.

To assign a macro to a button:

1. With a document open, click Tools > Customize.
2. In the dialog box, on the Commands tab:
    a. Select Macro in Categories.
    b. Under Buttons, drag the Macro  button to any toolbar in the SolidWorks window.
3. In the Customize Macro Button dialog box:
    a. Under Appearance:

1. Click Choose Image.
2. In the Icon path dialog box, select a bitmap image (*.bmp), then click Open.

   NOTE: The SolidWorks software provides bitmap images to use as custom buttons. These are located in <*install_dir*>/data/usermacro icons. Select Thumbnail to see the image in the Icon path dialog box.
3. Type a Tooltip and Prompt message, which should provide a brief description of the function of the tool on the status bar. Both the ToolTip and message are displayed when the pointer is on the bitmap.

b. Under Action:

1. Click  and navigate to the folder where the macro is stored.
2. In the Macro Path dialog:

a. Select SW VBA Macros (*.swp) if adding a VBA macro (selected by default).
   - or -
   Select SW VSTA Macros (*.dll) if adding a VB.NET or C# macro.
b. Select the macro.
c. Click Open.

3. Select the macro that you want to assign to the button.
4. Click Open.

c. Click OK.

4. Click OK again to close the Customize dialog box.

To edit a macro button:

1. In an open SolidWorks document, click Tools > Customize.
2. Right-click the macro button in the toolbar that you want to edit and select Properties.
3. In the Customize Macro Button dialog, edit the macro button and click OK.
4. Click OK again to close the Customize dialog.

To delete a macro button:

1. In an open SolidWorks document, click Tools > Customize.

2. Right-click the macro button in the toolbar that you want to edit and select Delete.
3. Click OK to close the Customize dialog.

3.4 SolidWorks API Standalone And Add-in Applications:

The programming languages used, that supports COM to create SolidWorks standalone API (.exe files) and add-in (.dll files) applications. The programming languages most commonly used are:

- Visual Basic .NET (VB.NET)
- Visual C++/CLI
- Visual C# .NET
- Visual C++ 6.0

Visual Basic .NET Standalone and Add-in Application:

Standalone Applications (.exe files)

To create an instance of the SolidWorks software, the project should contain lines of code as follows:

```
Sub Main

    Dim swApp As SldWorks.SldWorks

    swApp = New SldWorks.SldWorks()

    swApp.ExitApp

     swApp = Nothing

    End Sub
```

Additionally, references to the SolidWorks type libraries must have added.

CHAP
TER 4

MACRO DEVELOPED IN SolidWorks™

The MACRO developed is aimed at increasing the performance, efficiency and the effectiveness of the user. The MACRO is basically a collection of VB codes sequenced together to form a meaningful output in the modelling environment. The VB codes are modified in such a way to display the user forms so that it is very convenient for the user to work on them effectively. All the mathematical relations are considered in the VB code to get the geometrically correct body in the modelling environment.

4.1 Algorithm for the working of MACRO for freeform

change The MACRO functioning is discussed in the

Algorithm below: **STEP 1:** Play button is clicked to start

the MACRO

A VB form is popped up on the screen , which enable the user to perform various actions. The form is shown in the figure 4.1 as shown on next page.

This Form has the options of controlling the curvature of the spline, which is responsible for making a petal of the flower, then making a complete flower out of it. After the required curvature flower is made, it is Embossed on a cylindrical solid, where its 3D curvature can be changed (controlled by a scroll bar). And finally a complete pattern is generated on the cylindrical body.

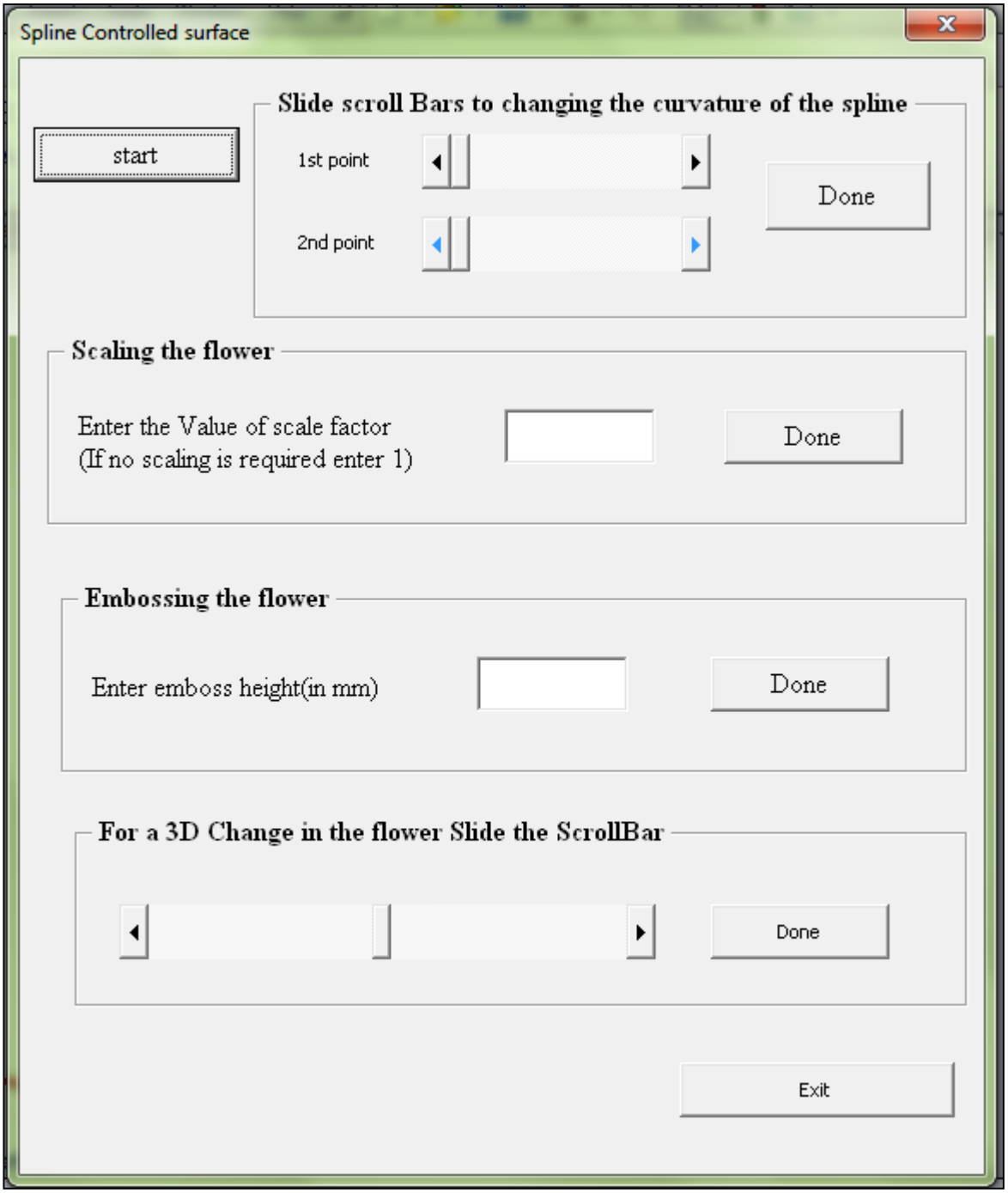


Figure 4.1 : UserForm

STEP 2: Pressing the Start Button.

When the start button is pressed, a default leaf petal is generated as shown in the figure 4.2.

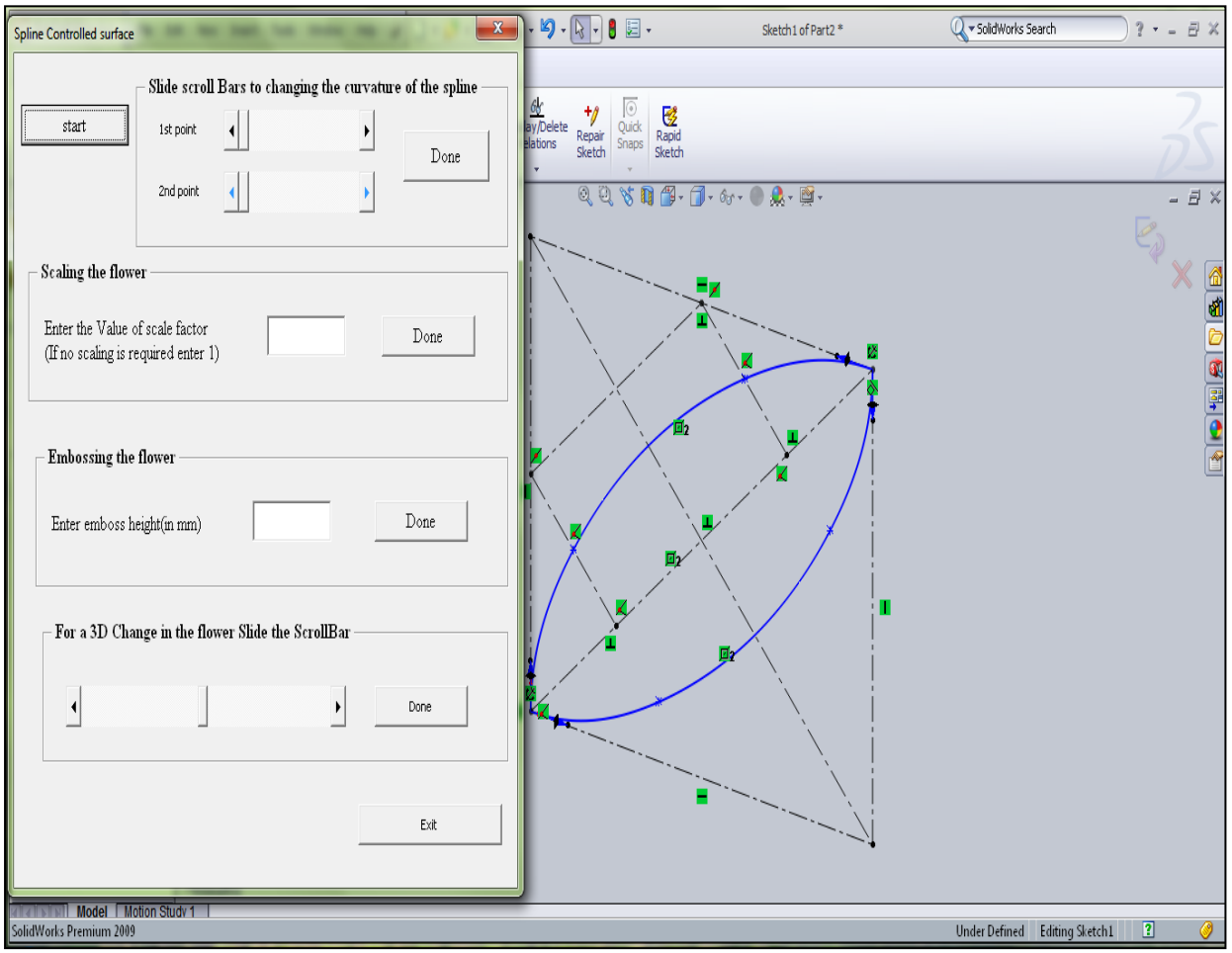


Figure 4.2: MACRO UserForm1

Then, the curvature of the two curves can be controlled by the means of the two scroll bars provided under the heading of 1st point and the 2nd point. Hence getting the final shape for the petal of the complete flower.

STEP 3: Enter the value of scale factor required.

The user is required to enter the value of scale factor and clicking on the done button. The sketch is scaled to the level entered by the user. After that , the emboss height is needed to be entered. After entering the emboss height, the user needs to click on the Done botton, and the shape generated by the modeler is shown in the figure 4.3.

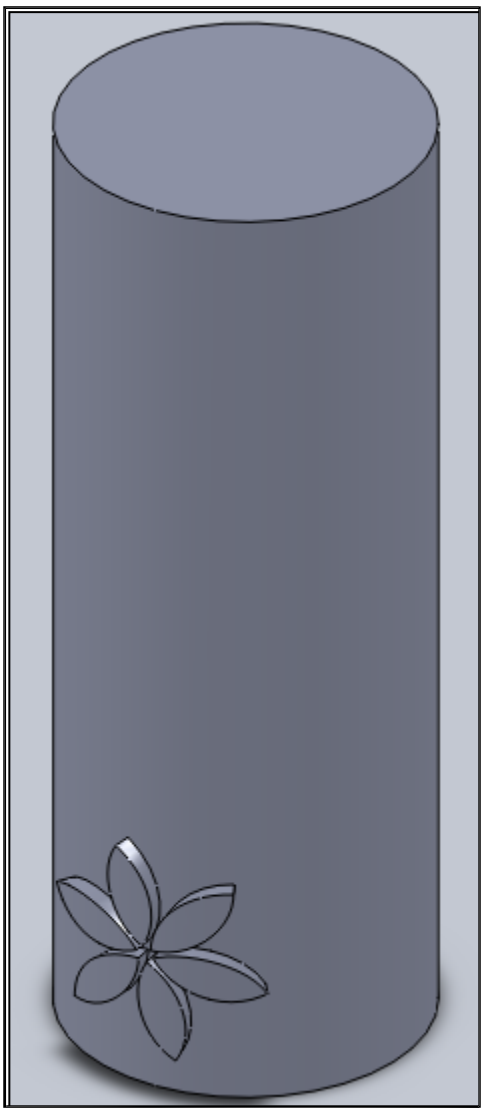


Figure 4.3: Embossed flower

STEP 4: Adjusting the scrollbar and getting the 3D curvature of the final flower.

By sliding the scroll bar, under the heading of 3D change in flower, the user is able to get various

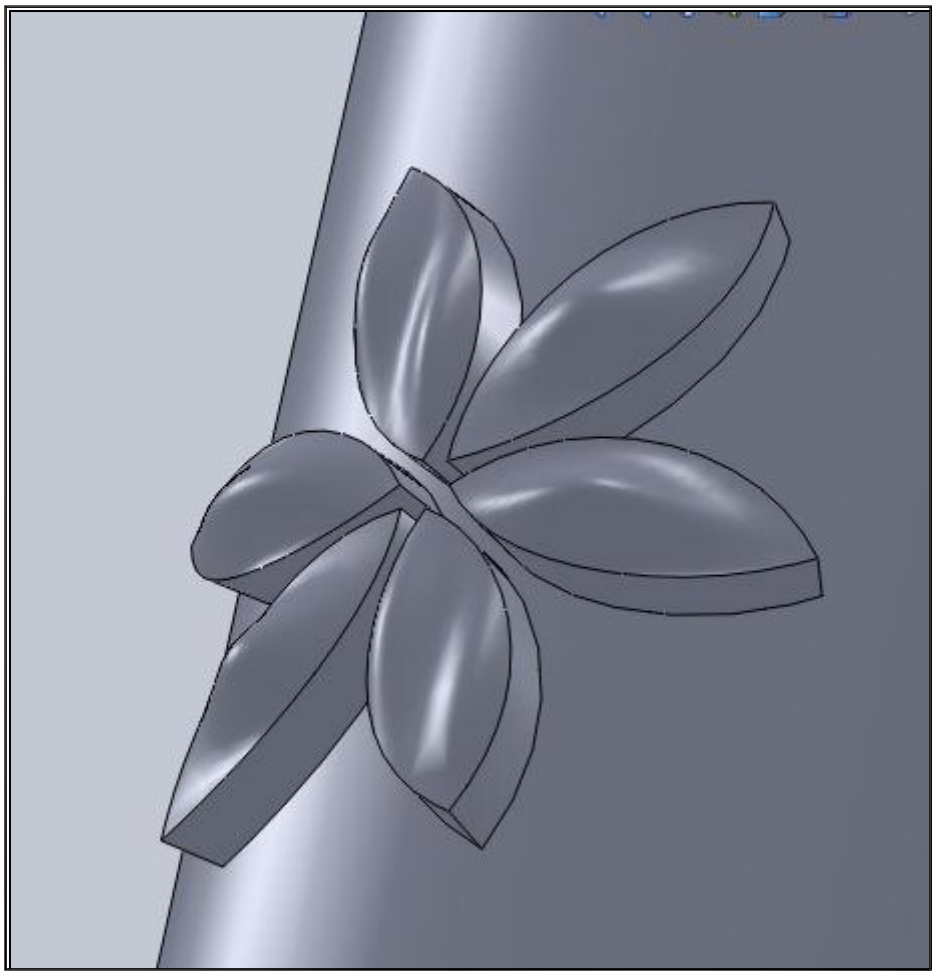


Figure 4.4 : 3D flower

STEP 5: Click on the Exit button

When the desired model is achieved. Exit button is pressed. And the solid generated as a result is shown in the figure 4.5.

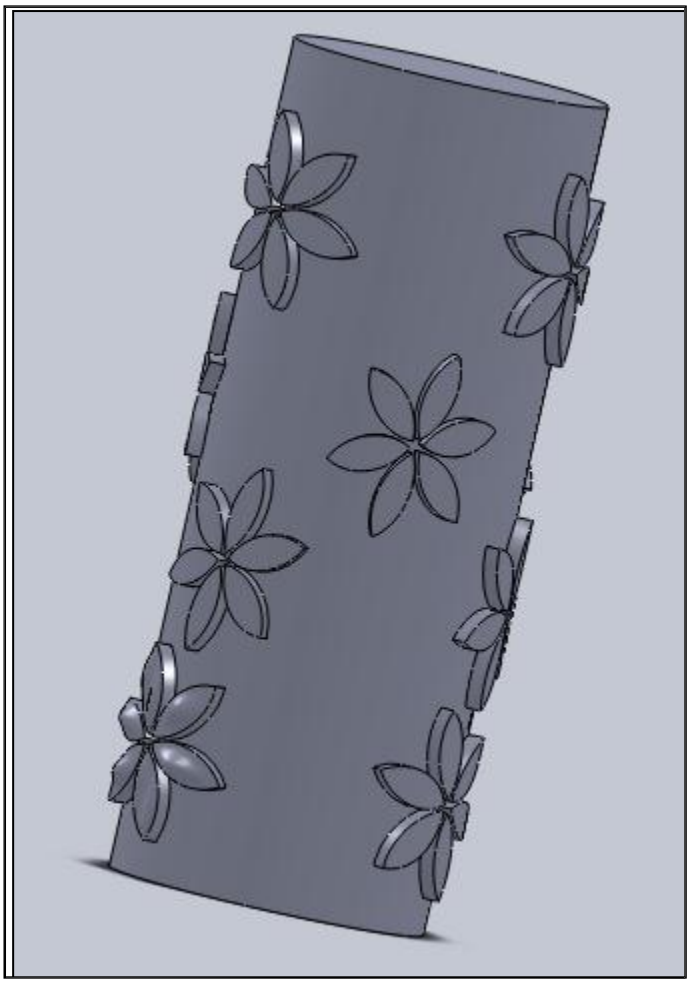


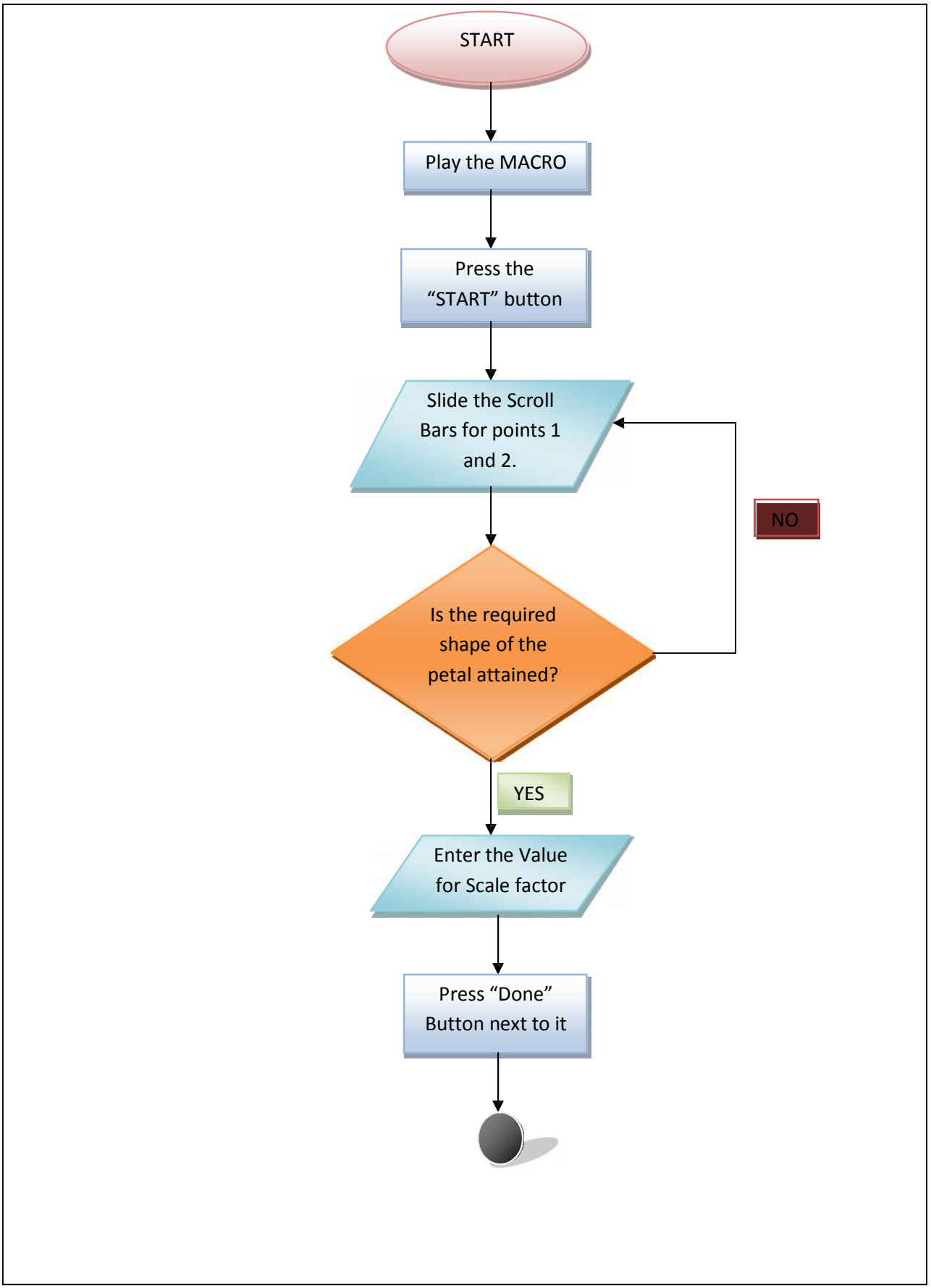
Figure 4.5: Final Output

The figure 4.5 shows the final solid generated by using the developed MACRO.

The algorithm shows the working of the developed MACRO designed out of VB forms, helping the user to increase the performance of the work, along with that creating infinite number of shapes of a flower pattern.

Figure 4.6 : 5 Petal flower Embossed on cylinder

4.2 Flowchart showing the control of the developed MACRO



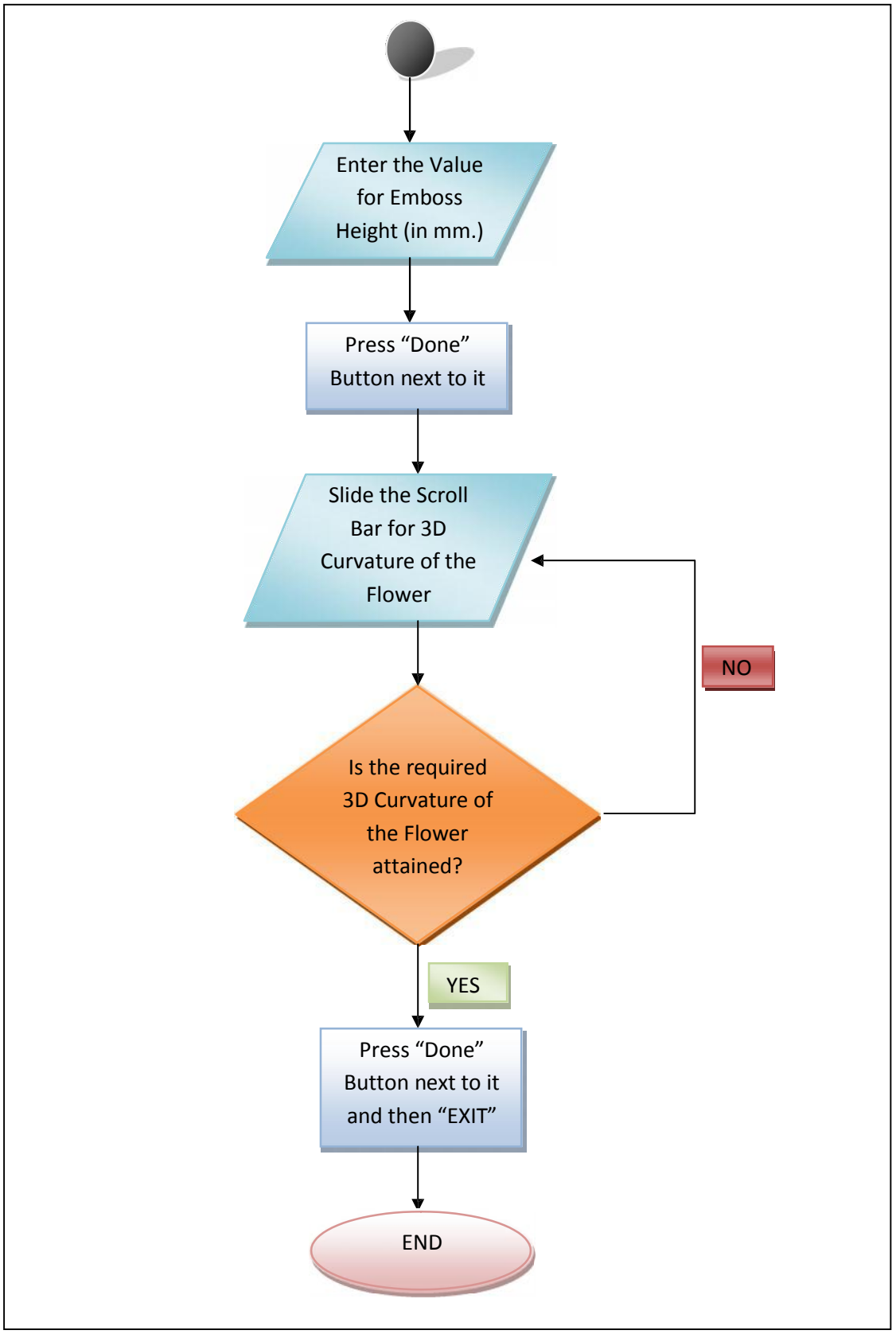


Figure 4.7: Flowchart explaining the flow of control of MACRO 1

4.2 Working of MACRO for UserForm2

VB form popped up on the screen enables the user to perform various actions. The form is shown in the figure 4.6 as shown on next page.

This Form has the options of selecting the base surface, whether flat, or cylindrical , or even elliptical. After the basic shape is selected, curvature of the flower is maintained by the means of scroll bars. Then the scale factor is selected. After the required curvature flower is made, it is Embossed on a cylindrical solid, where its 3D curvature can be changed (controlled by a scroll bar). And finally a complete pattern is generated on the initial basic shape selected.

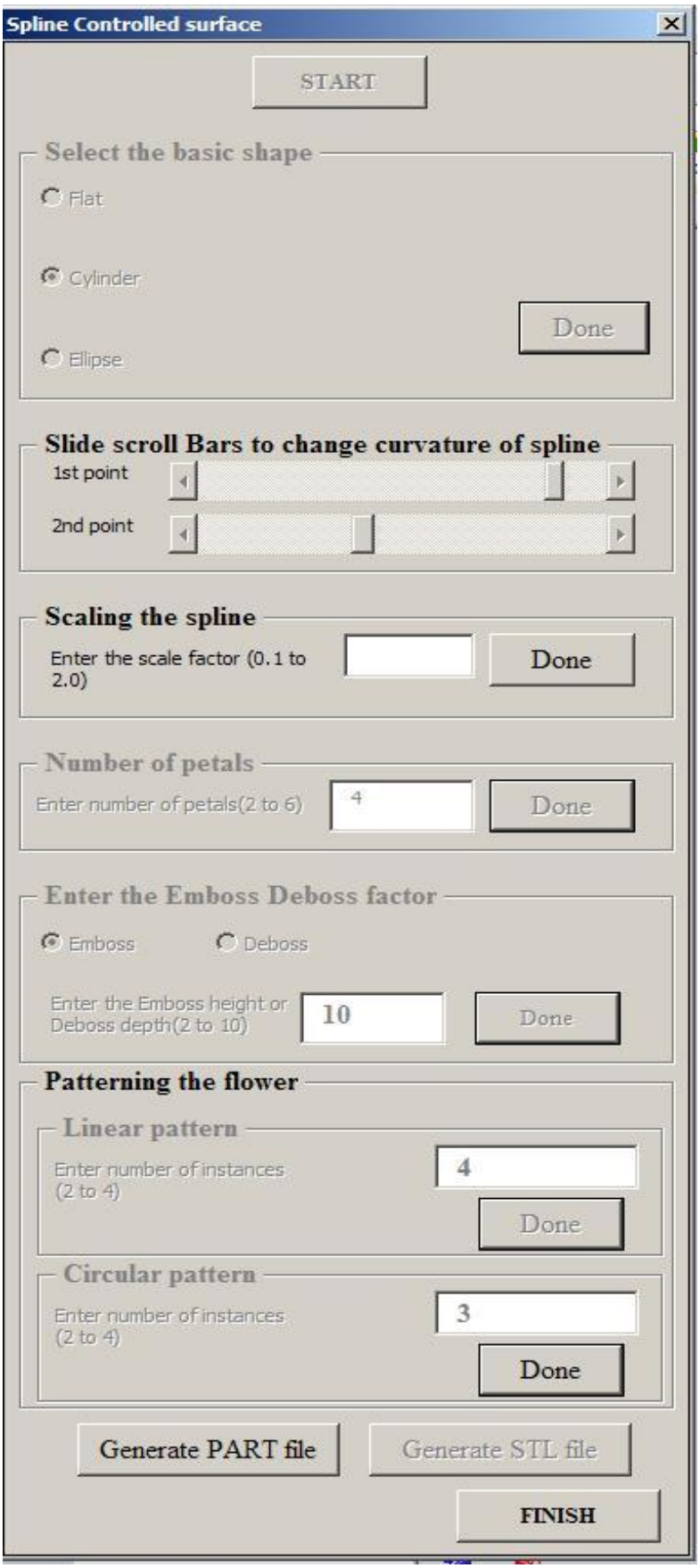


Figure 4.8 : MACRO UserForm 2

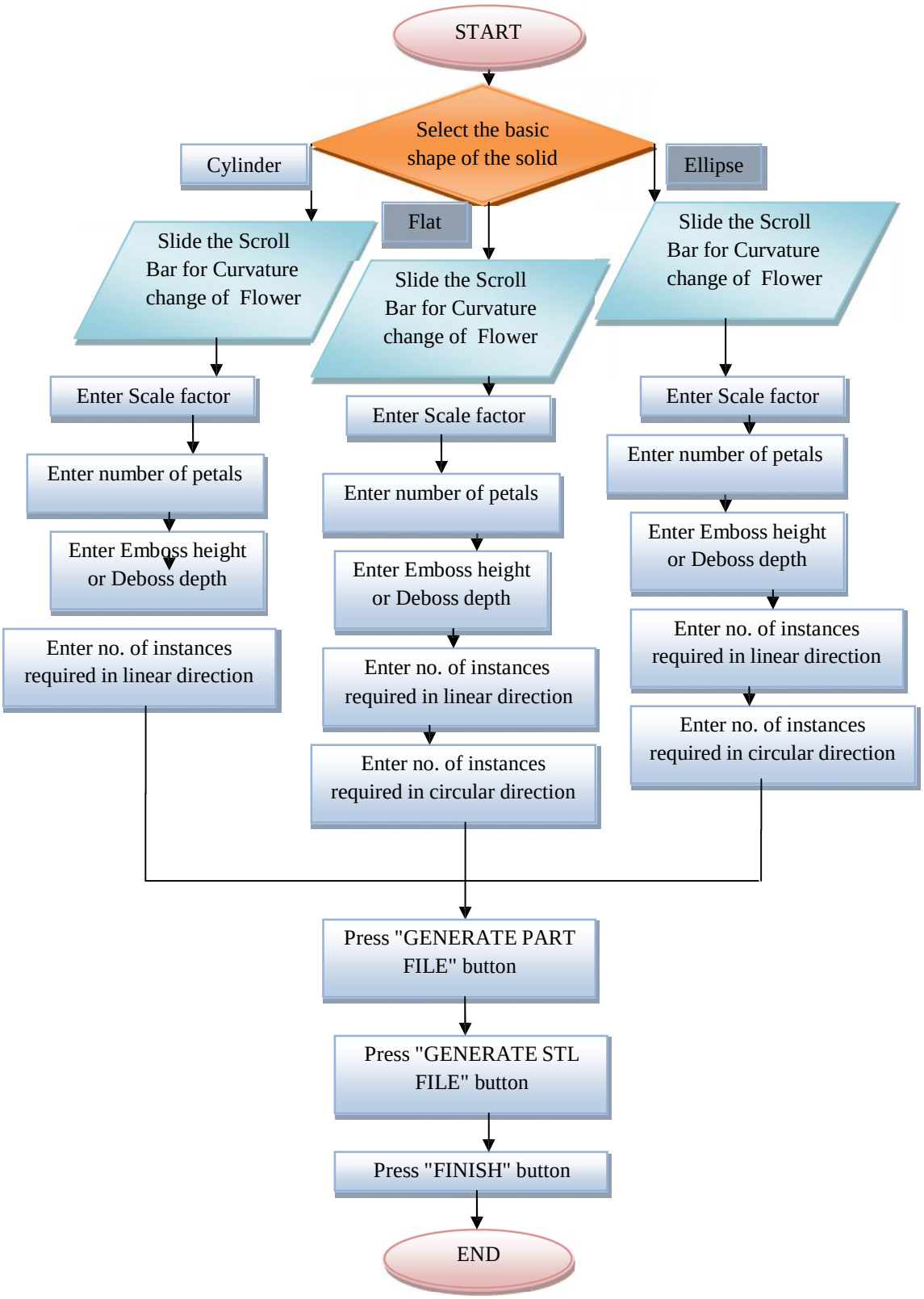


Figure 4.9 : Flowchart explaining the flow of control of MACRO 2

CHAPTER 5

Logic and Mathematical control

The 2D sketch, shown in figure 5.1 shows two curves C1 and C2. C2 curve is the mirror image of the first curve C1. The curve is basically a Bezier curve with curve with cubic degree. The Analysis of the curve involves the positioning of the control points. Control points are the guiding constraints of a curve. The control points predict the flow of curve. Thus, by manipulating these control points the curvature of the curve can be changed and infinite number of differently curved splines can be formed , keeping the first and the last control point constant and manipulating the in between control points. In SolidWorks™ Bezier spline with cubic degree is used.

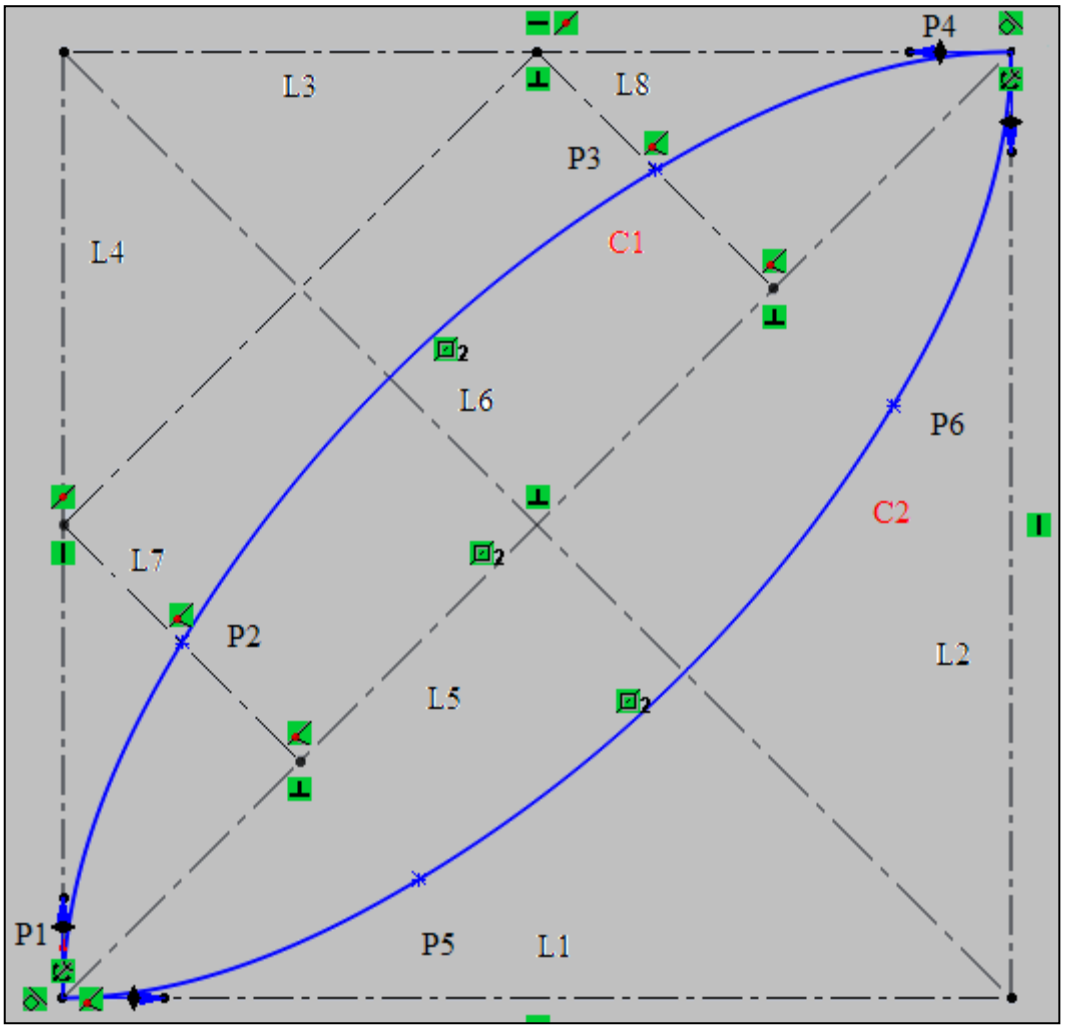


Figure 5.1 : 2D sketch of canvas

The Mathematical control of the Bezier involves the following formulation:

$$\mathbf{P(u)} = \sum_{\mathbf{i=0}}^{n} P_i \, \mathbf{B_{i,n}(u)}$$

Where, $\mathbf{0 \leq u \leq 1}$

The factor is controlled here. This factor is responsible for the change in the curvature of curve. Its limiting conditions are at 0 and 1.

$\mathbf{B_{i,n}(u)}$ is the Bernstein function or the Blending function. This parameter is responsible for the continuity of the curve along different control points.

And, $\mathbf{B_{i,n}(u) = C(n,i)u^i(1-u)^{(n-i)}}$

And, $\mathbf{C(n,i) = \frac{n!}{i!(n-i!)}}$

Now, for Bezier curve of cubic degree, we have the values of n =3. Therefore the number of control points = **(n+1)** = 4.

The values of i are taken up as 0,1,2,3.

Substituting these values in the above Bernstein formulation, we get:

$$\mathbf{B_{0,3} = (1-u)^3}$$

$$\mathbf{B_{1,3} = 3\,u\,(1-u)^2}$$

$$\mathbf{B_{2,3} = 3\,u^2\,(1-u)}$$

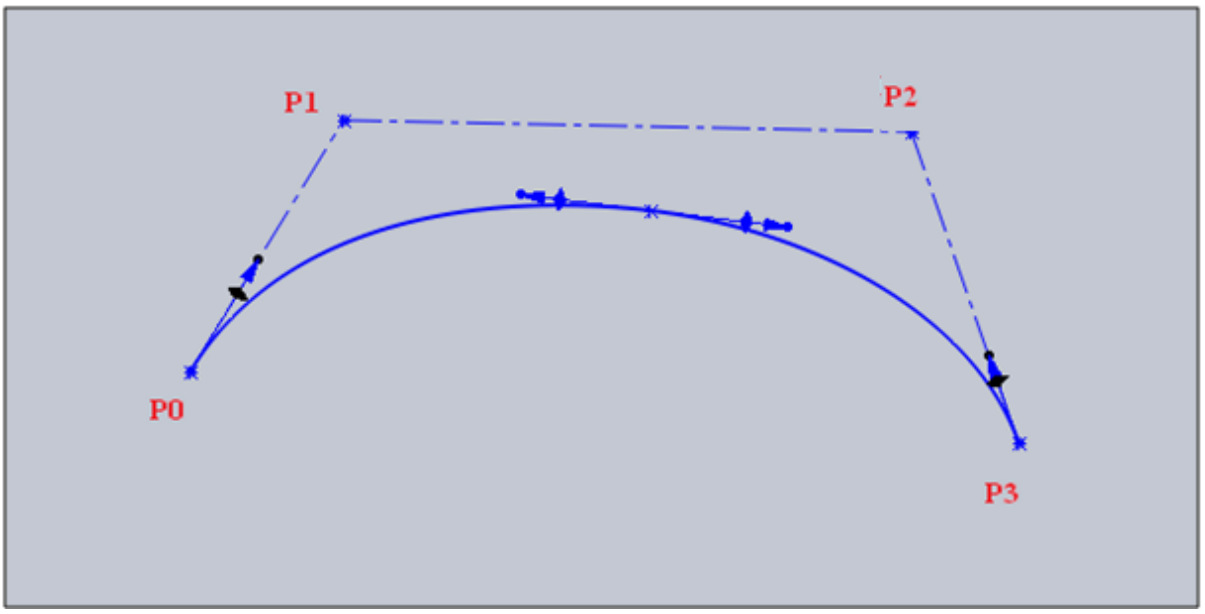
$$\mathbf{B_{3,3} = u^3}$$

So, the final Bezier formulation sums up to:

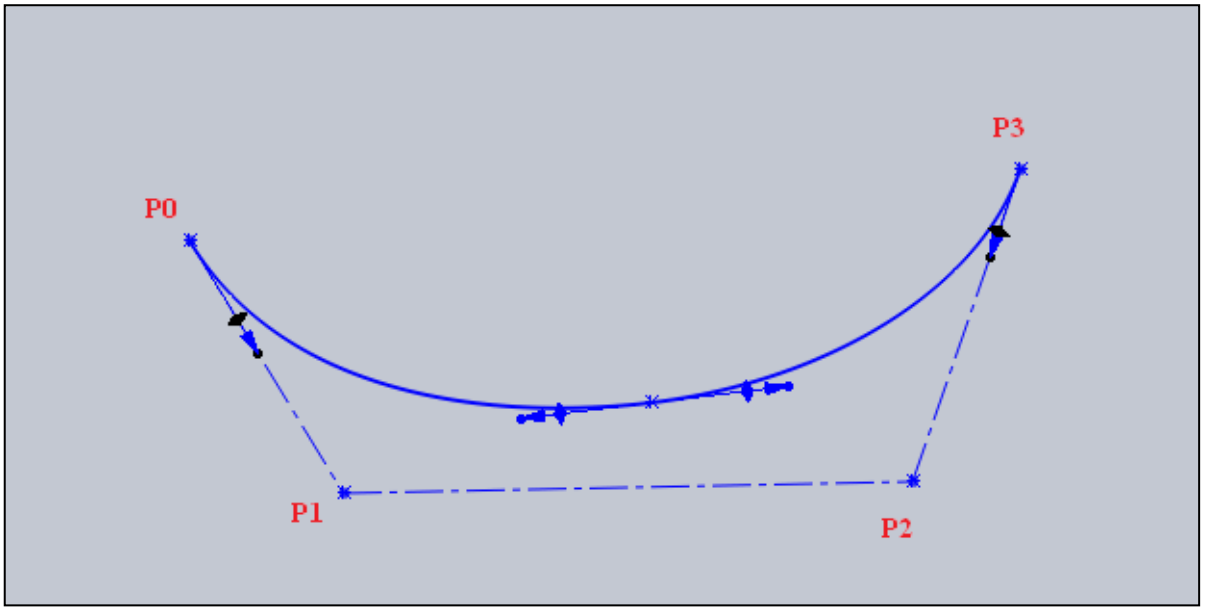
$$\boldsymbol{P(u) = P_0(1-u)^3 + 3P_1\,u\,(1-u)^2 + 3\,P_2\,u^2(1-u) + P_3u^3}$$

Now, the variable factors left over are the control points P_0 , P_1 , P_2 and P_3 along with the parameter "**u**". These control points are given values at the run time by changing the parameter u using the UserForm.

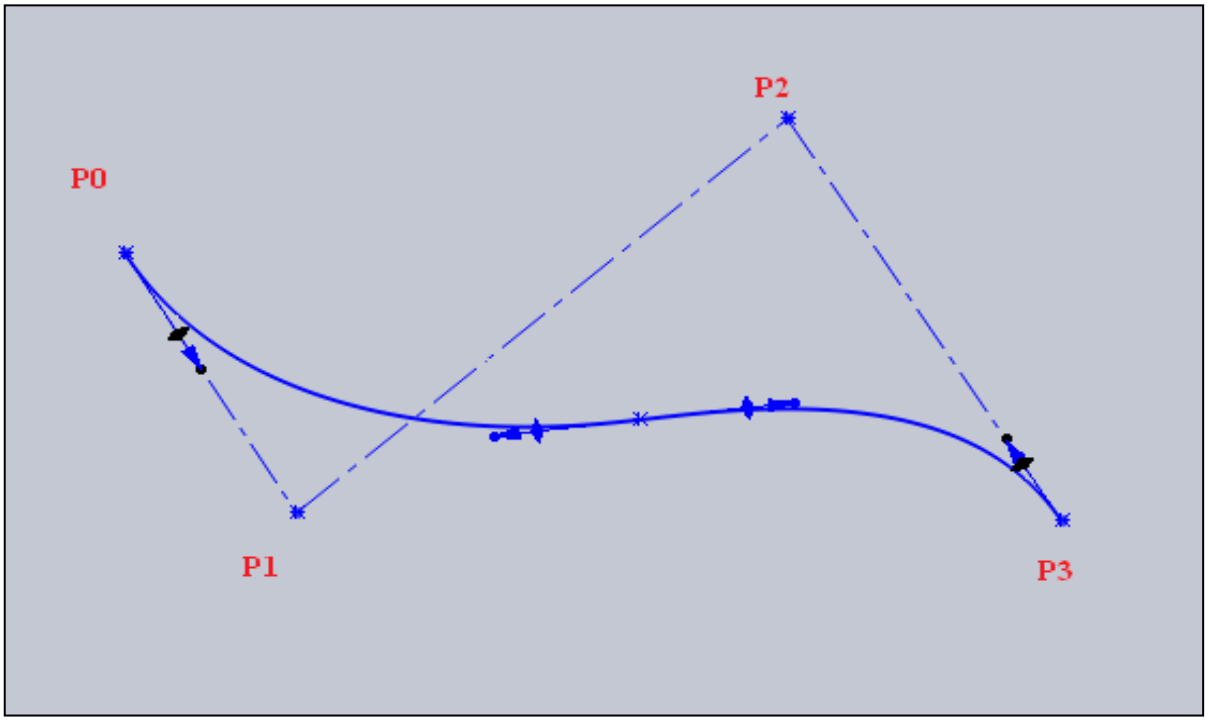
The Various shapes of the Cubic Bezier for differently placed Control Points are shown below in the figure 5.2



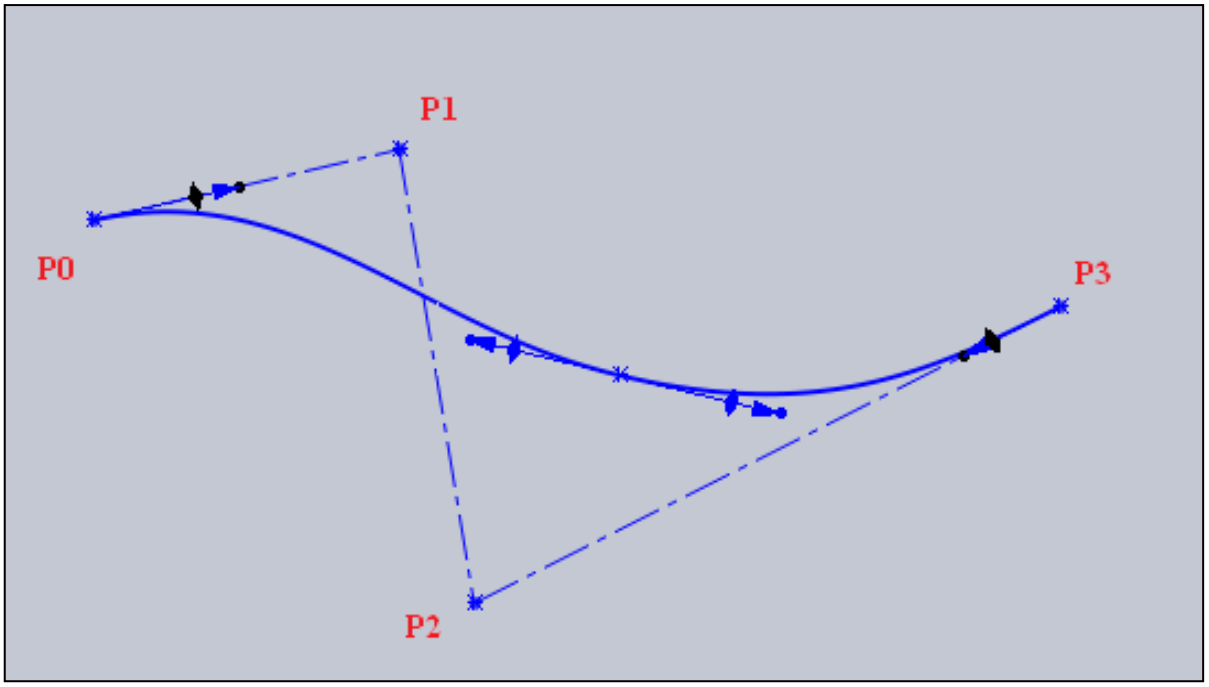
.Figure 5.2 (a) : Cubic Bezier shape for specific location of control points



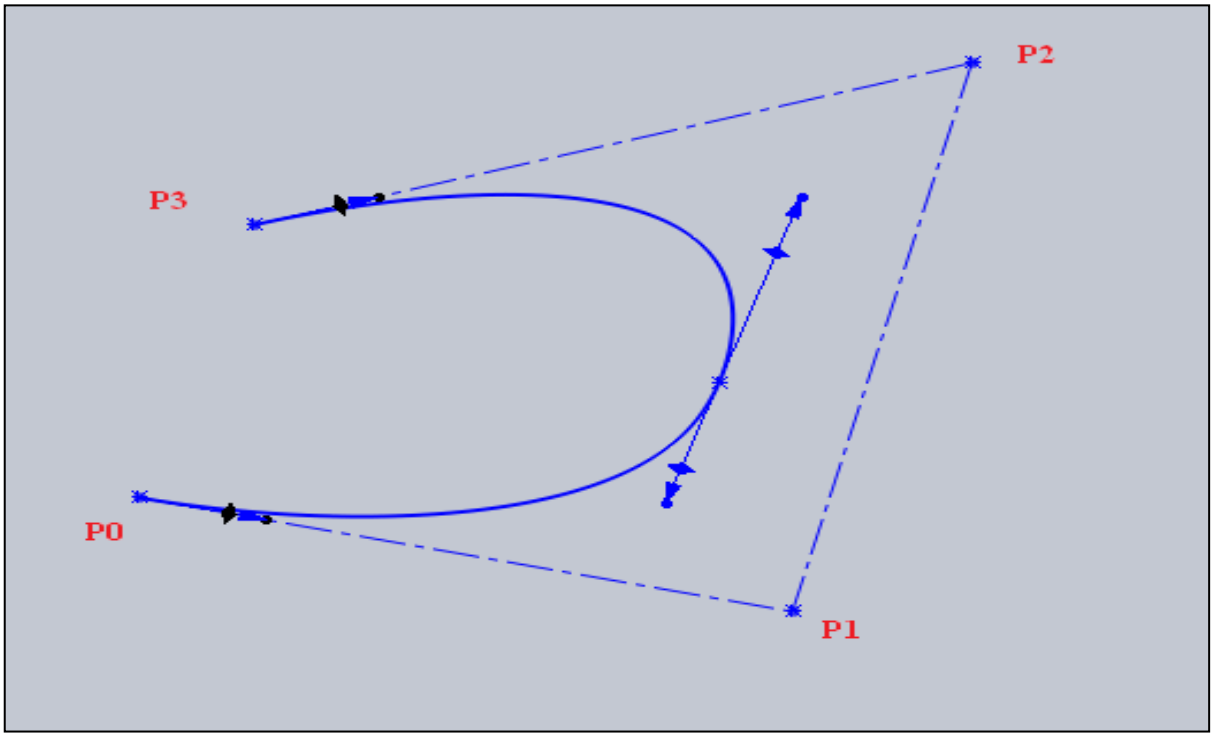
.Figure 5.2 (b) : Cubic Bezier shape for specific location of control points



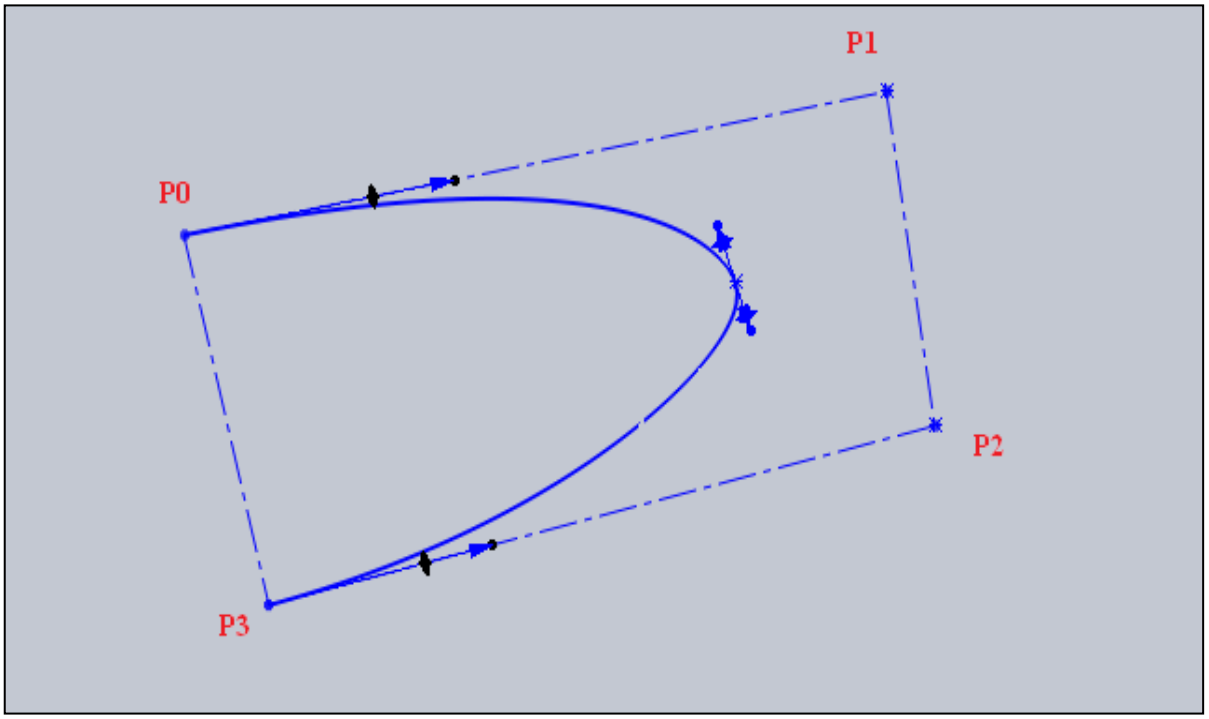
.Figure 5.2 (c) : Cubic Bezier shape for specific location of control points



.Figure 5.2 (d) : Cubic Bezier shape for specific location of control points



.**Figure 5.2 (e) : Cubic Bezier shape for specific location of control points**



.**Figure 5.2 (f) : Cubic Bezier shape for specific location of control points**

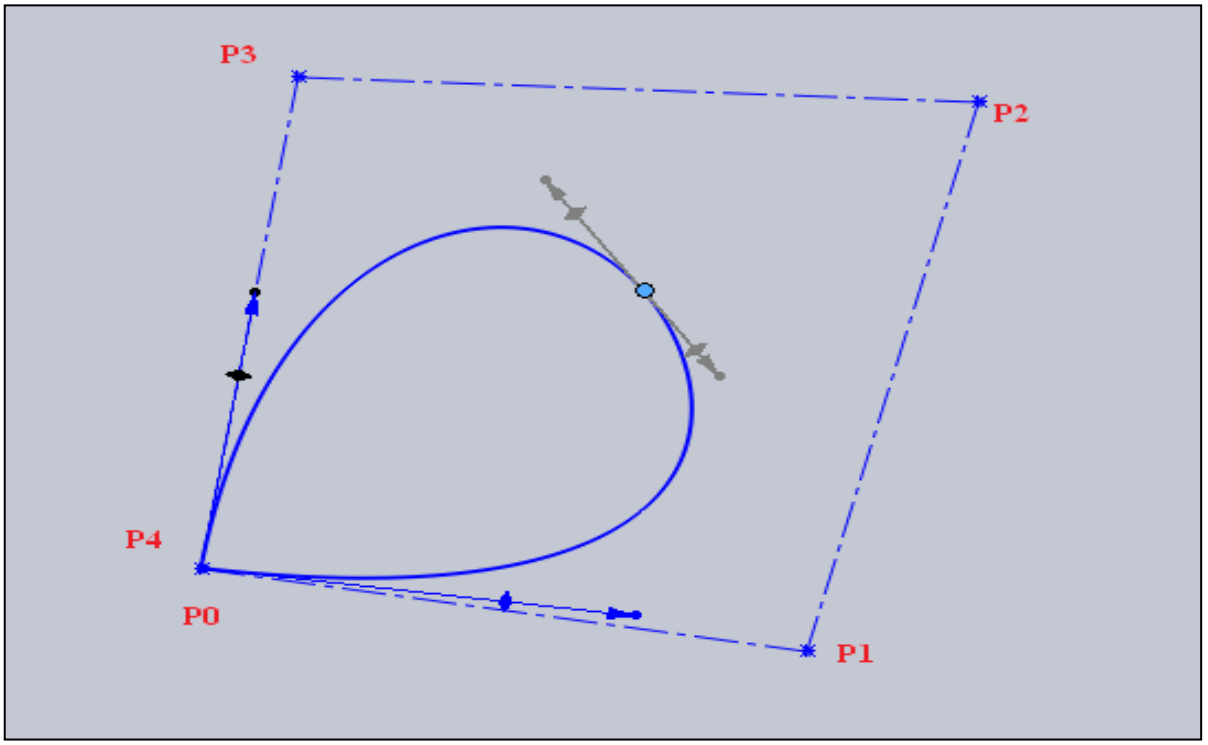
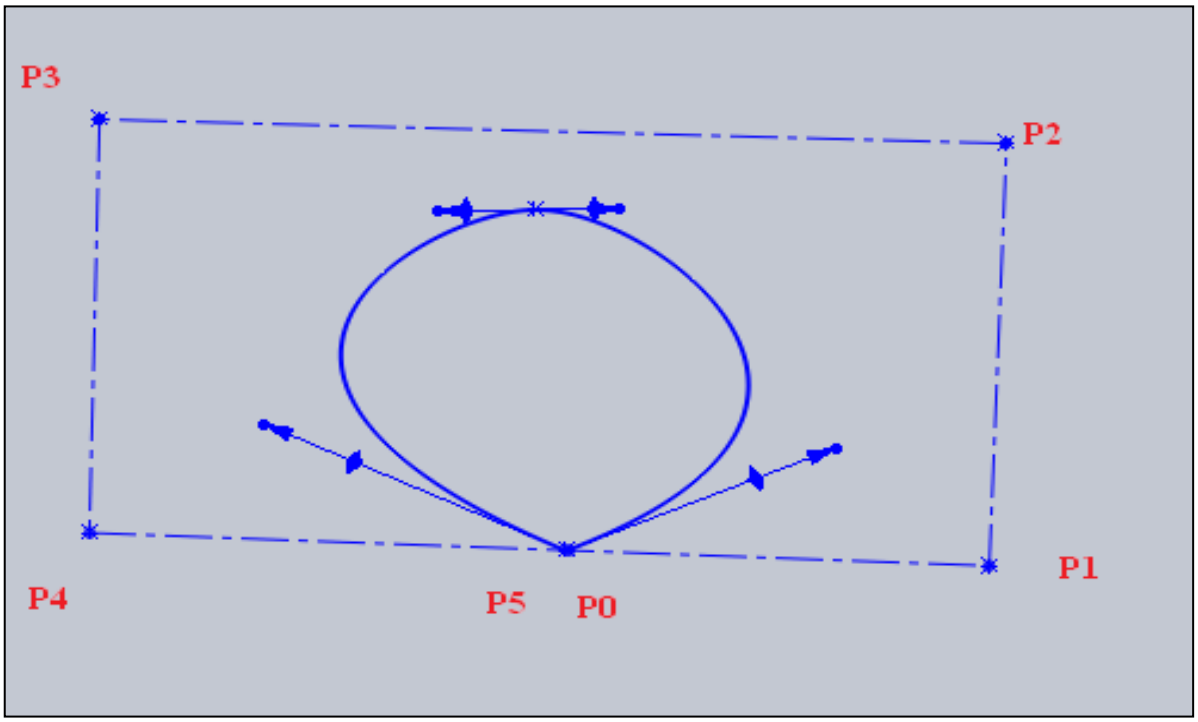


Figure 5.2 (g) : Cubic Bezier shape for specific location of control points



.Figure 5.2 (h) : Cubic Bezier shape for specific location of control points

Finally, in the 2D sketch developed in figure 5.1, this Bezier curve is supposed to be tangent to the Lines L3 and L4, thereby coincident Points P2 and P3 with the Lines L7 and L8.

For these Points of the curve to lie on the line, the intersection of the Lines is considered with the Curve C1.

The UserForm controls the curvature of the splines by the means of a scrollbar. The 2D sketch of the splines is shown below in the figure 5.1.

The variable positions of the points P2 and P3 is controlled by the scrollbar. These positions or the instananeous points all lie on the line L7 and L8. So, using the standard equation of the line,

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$

So the equation of Line L7 (for a canvass size of 100mm × 100mm) is given by:

$$x + y = 50$$

So the equation of Line L8 (for a canvass size of 100mm × 100mm) is given by:

$$x + y = 150$$

By Generating the intersection points of the Lines L7 and L8 with the equation of Curve C1, we can get the following shape, which can be instantaneously controlled by the scrollbar at the run time. The following figure 5.1 exactly describes the application of these mathematical formulation and getting the final output in the SolidWorks™.

The figure 5.1 shows the 2D sketch of the canvass in which there are following entities :

1. 8 Construction lines (that is, L1, L2, L3, L4, L5, L6,L8)
2. 6 Points (that is, P1, P2, P3, P4, P5, P6)
3. 2 Spline Curves (that is, C1,C2)

Now, the constraints which completely define the sketch are discussed in table 5.1.

Entity	Constraint
1. Line L1	Horizontal
2. Line L2	Vertical
3. Line L3	Horizontal
4. Line L4	Vertical
5. Line L5 and L6	Perpendicular to each other
6. Curve C1 and Line L4	Tangent
7. Curve C1 and L3	Tangent
8. Point P1 and Point P4	Fixed
9. Line L5	Mirror Axis
10. Point P2	Coincident on Line L7
11. Point P3	Coincident on Line L8
12. Point P2 and P5	Symmetric
13. Points P3 and P6	Symmetric

Table 5.1 : Table of Constraints

Now, the requirement is that the spline should change its curvature. It is achieved only when the point P2 and P3, being coincident on the line L7 and Line L8 should scroll over these line thus leading to a change in curvature of the spline.

In this equation, we have x, y as unknown parameters and rest as known parameters.

Now , when the user slides the scroll bar, an instantaneous value is given as a feedback by the scroll bar. The value of the scrollbar is given to the equation of the line for calculating the intersection point of the curve C1 and the line L7. And hence the value of Point P2 is obtained.

Similarly, When the second scrollbar is used to get second instantaneous value, which will be responsible for the generation of the Point P3. The value of the second scrollbar is then used to find the intersection point of line L8 and curve C1, thereby generating the point P3.

After getting the points P2 and P3 , the curve C1 is generated with first point P1(fixed at the start of the canvas) and end point as P4(fixed at the end diagonal of the canvas). So by using the scroll bar , any number of shapes of the curve C1 cab be generated.

Curve C2 is generated as a result of mirroring of the final curve C1 about the mirror axis, line L5, thereby, leading to formation of a complete petal of a flower.

1.1 Various other shapes that can be generated

While sliding the scrollbar infinite number of shapes of the curve can be generated. A few of them are shown in the figures 5.3(a), 5.3(b) and 5.3(c).

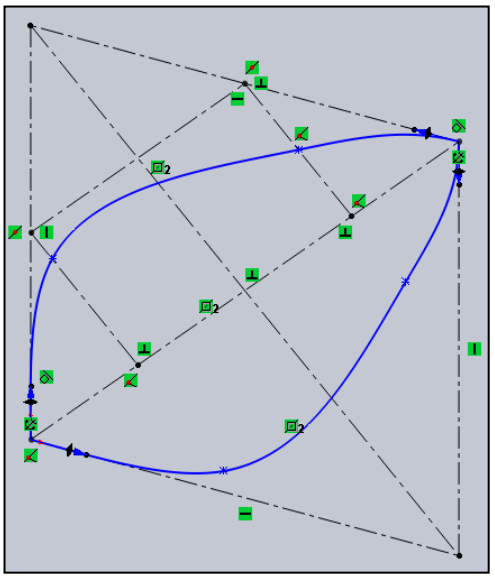


Figure 5.3(a) : Spline controlled petal shape

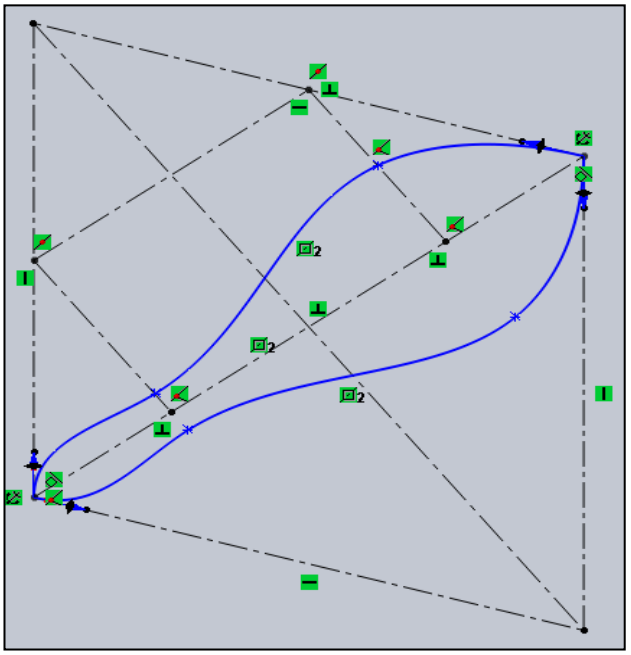


Figure 5.3(b) : Spline Controlled petal shape

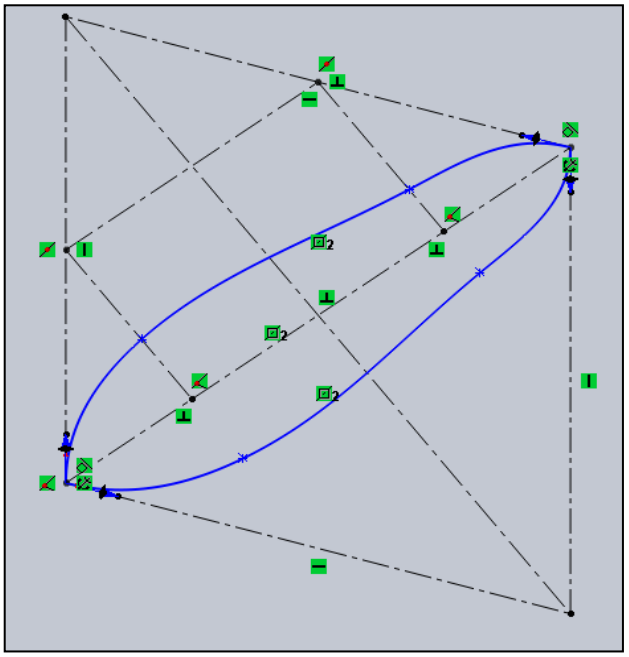


Figure 5.3(c) : Spline controlled petal shape

The various shapes thus attained of the 2D sketch of the flower can then be embossed on the cylindrical body of desired height. And finally 3D curvature can be controlled of the final flower before making artistic patterns of the flower.

CHAPTER 6

Results and Validation

6.1 Results and Validation of the MACRO

The validation of the work can be discussed by the screen shots of the part that is actually developed. The user defined values given at the run time are stored in the form of variables in the background VB environment. Further the user defined values pass into the functions that are responsible for the specific actions, which they are meant for. When all the desired inputs are taken from the user , the final user defined object is projected on the screen. The figure 6.1(a), shows the final shapes attained when the user had given inputs:

Base Shape = Cylinder,

Number of petals = 5,

Emboss Height = 15mm,

Number of instances in linear direction = 4,

Number of instances in circular direction = 3,

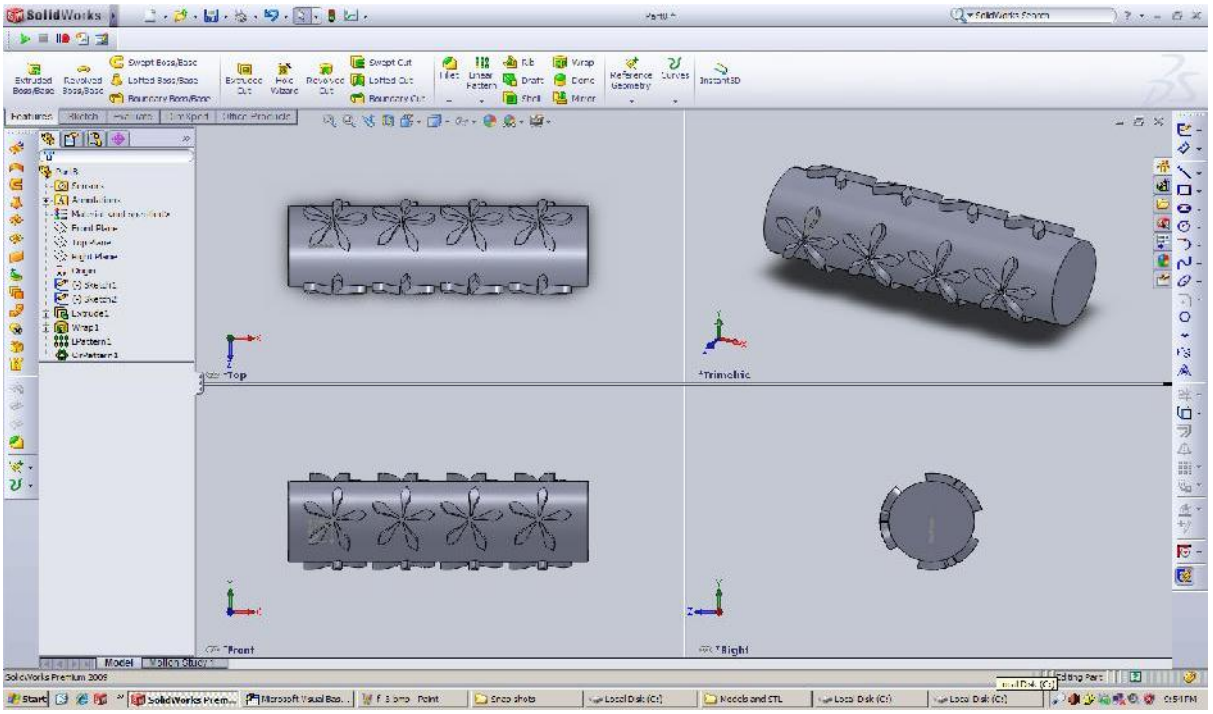


Figure 6.1 (a) : Validated output of user defined values.

Figure 6.1(b) shows the validated output of the user defined values as:

Base shape = Ellipse,

Number of petals = 4,

Emboss Height = 18mm,

Number of instances in linear direction = 4,

Number of instances in circular direction 3,

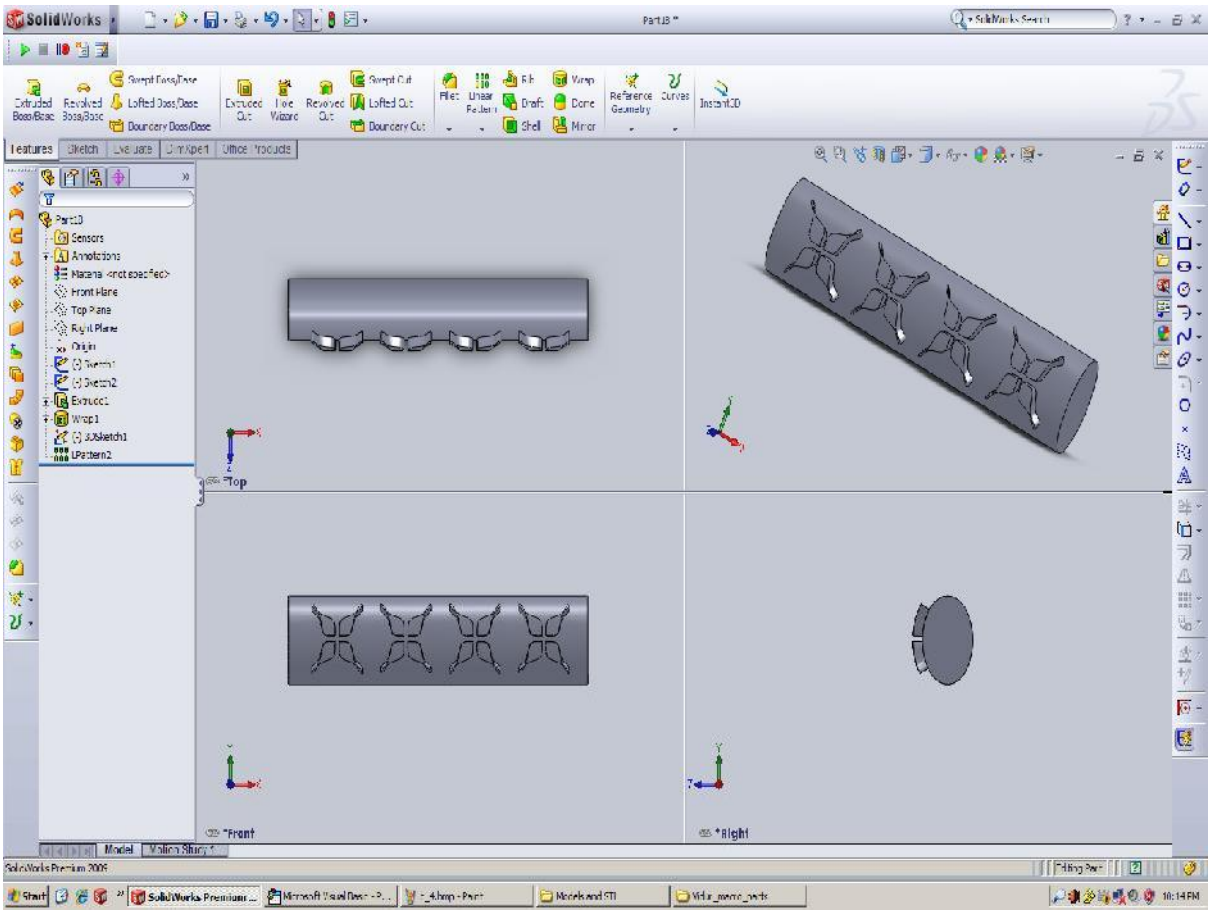


Figure 6.1 (b) : Validated output of user defined values.

Figure 6.1(c) shows the validated output of the user defined values as:

Base shape = Ellipse,

Number of petals = 6,

Emboss Height = 12mm,

Number of instances in linear direction = 3,

Number of instances in circular direction = 1,

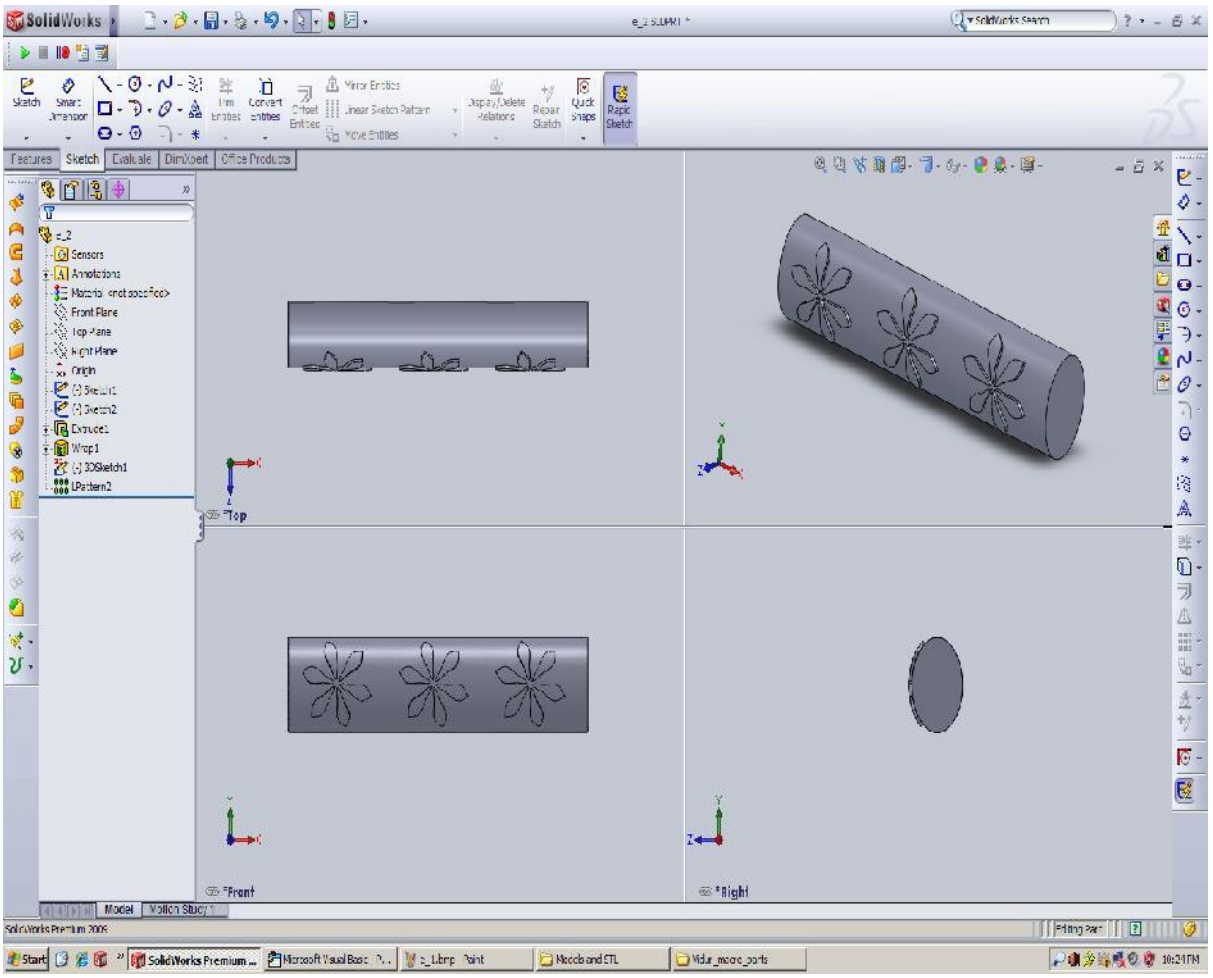


Figure 6.1 (c) : Validated output of user defined values.

Figure 6.1(d) shows the validated output of the user defined values as:

Base shape = Flat,

Number of petals = 5,

Deboss Height = 12mm,

Number of instances in first linear direction = 4,

Number of instances in second linear direction = 2,

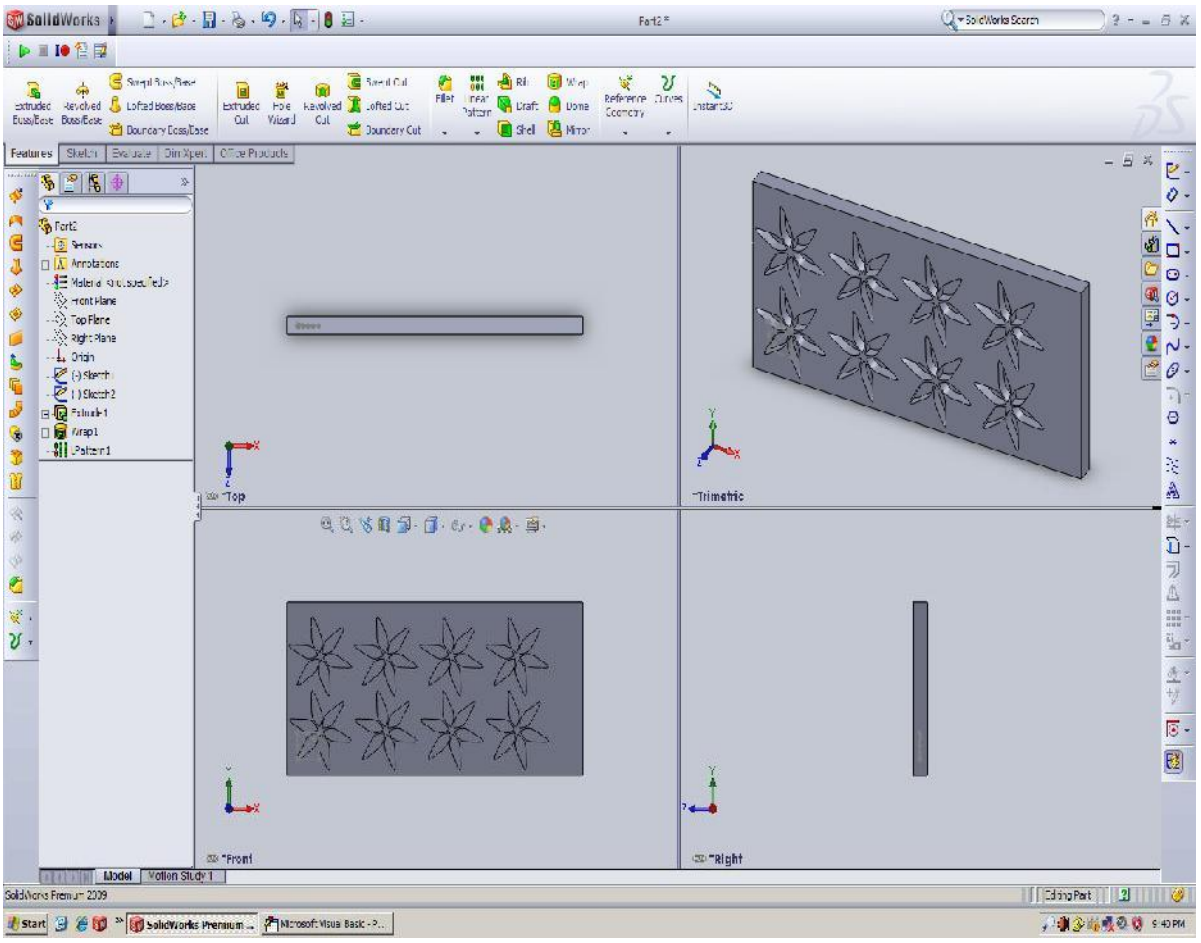


Figure 6.1 (d) : Validated output of user defined values.

Figure 6.1(e) shows the validated output of the user defined values as:

Base shape = Flat,

Number of petals = 6,

Emboss Height = 12mm,

Number of instances in first linear direction = 4,

Number of instances in second linear direction = 2,

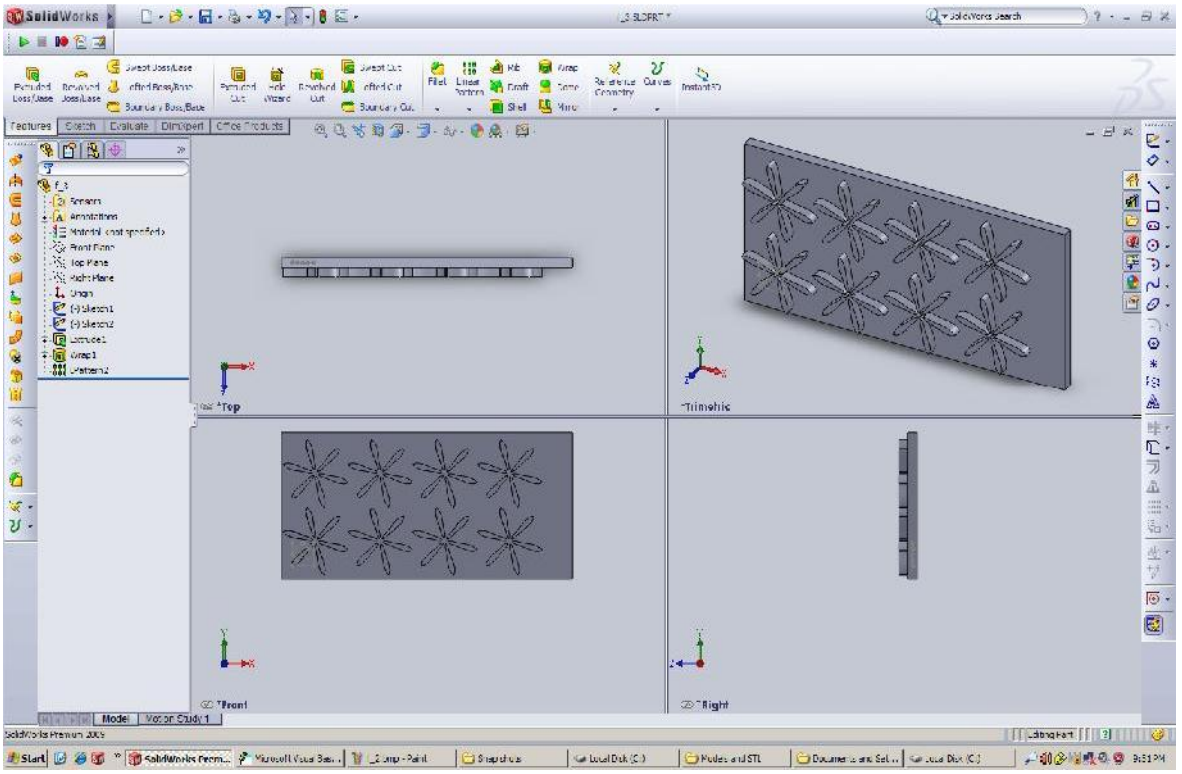


Figure 6.1 (e) : Validated output of user defined values.

The solids thus generated can be sent for machining using the CAM Module. STL file of the body is generated as soon as the MACRO ends. This STL file converts the complete body into triangular facets . STL file serves as an input to the tool path planning program, thereby creating a "tp" extension file. This file thus goes to the CNC machining for the actual machining purpose.

Chapter 7

Conclusion of the Work and Future Scope

The project is intended to help and simplify the work of a non trained user, so after the successful completion of this research work the user can get the desired sculptured product of cylindrical cross-sectional product.

The work is to simplify the user's labor. User just needs to fill up the FORMS generated in the API which is linked to Microsoft Visual basic. The form contains the details of what all are requirements of the user. The form is a graphical interface between the user and the background coding. The values filled on the form have a significant meaning in the background coding. So when the user finishes the form the actual desired shape of the product is displayed on the computer screen. After taking confirmation from the user that the final product is according to the requirement of the user, the complete modeled part is converted in STL format. This STL file is sent to the program where tool path is generated, for the actual machining purpose.

The further scope of the presented work will lead to integration of the design process with the manufacturing process. The project will provide a new tool based on CNC technology that will add to the capability of the wood handicraft/Artisans and allow them to build furniture likes of which are not possible today. Various complicated shapes of the flower which were earlier not possible will be achieved by this approach. The work of the manual woodworking will be completely replaced by this CAD/CAM tool .

So in brief this research work will prove to be a very efficient tool in the days to come.

REFERENCES:

[1] R. Brumana, L. Fregonese, C. Monti, C.C. Monti, G. Monti, E. Vio,"Complex analyses of surface, modelling and comparison of the 3D ortho photo to the real scale with historical cartography:mosaic surface of basilica of San Marco in Venice", e-Perimetron, Vol. 2, No. 4, pp[224-244],Autumn 2007.

[2] Zorin Denis, "a method for analysis of *c*1-continuity of subdivision surfaces", siam j. Numer. Anal, vol. 37, no. 5, pp 1677–1708, 2000.

[3] J.-K. Seong, Seoul, G. Elber, Haifa, J. K. Johnstone, Birmingham and M.-S. Kim, Seoul, "The Convex Hull of Freeform Surfaces", Springer-Verlag, 2004.

[4] Pezzuti, E., Piscopo, G., Ubertini, A., Valentini, P.P., Milana, M., Di Leginio, R., Una metodologia perl'analisi e l'archiviazione di reperti archeologici basata sul rilievo mediante scanner laser tridimensionali a non contatto, Archiviazione e Restauro di Reperti Archeologici Mediante Tecniche CAD-RP (in italian), Napoli, 2004.

[5] M. Leyton, Symmetry, Causality, Mind. Cambridge, Mass.: MITPress, 1992

[6] A.R. Rao and G.L. Lohse, "Identifying High Level Features of Texture Perception," CVGIP: Image Processing, vol. 55, pp. 218-233,1993.

[7] International Standards Organization, ISO6983-1—Numerical control of machines—program format and definition of address words—Part 1:data format for positioning, Line motion and contouring control systems,1982.

[8] Groover MP.Automation, production systems, and computer-integrated manufacturing. Prentice Hall Int. Publ; 2001.

[9] Seames WS. Computer numerical control: concepts and programming. 3rd ed.Albany: Delmar Publishers; 1995.

[10] Lutz P, Sperling W. OSACA—The vendor neutral control architecture. In:Fichtner D, et al., editors. Facilitating deployment of information and communications technologies for competitive manufacturing. Proceedings of the European conference on integration in manufacturing IiM'97. Dresden: Selbstverlag der TU Dresden; 1997.

[11] OMAC, Open Modular Architecture Controls, /http://www.omac.org/S, 2007.

[12] MDSI, Open CNC, /http://www.mdsi2.com/S, 2007.International Standards Organization. ISO10303-1—Industrial automationsystems and integration—Product data representation and exchange, Part 1.Overview and fundamental principlen s, 1994.

[13] Kemmerer SJ, editor. STEP The Grand Experience Manufacturing EngineeringLaboratory. Gaithersburg (MD): National Institute of Standards and Technology;1999.

[14] International Standards Organization. ISO10303-1—Industrial automationystems and integration—Product data representation and exchange, Part 1. Overview and fundamental principlen s, 1994

[15] Juan R. Hoffmann M.C. Erep - An Editable High-Level Representation for Geometric Deisgn and Analysis. In Pratt M. Wilson P, Wozny M., editor, Geometric and Priduct Modeling, pages 129–164. North Holland, 1992.

[16] Hoffmann C.M. Chen X. Design Compilation for Feature-Based, Constraint-Based CAD. In Proceedings of the Third ACM Symposium on Solid Modeling and Applications, page13–19, Salt Lake City (USA), December1995

[17] Anderson B. Shih C. A Design/Constraint Model to Capture Design Intent. In Proceedings of Solid Modeling'97, pages 255–264, Atlanta (USA),December 1997.

[18] Various authors. ENGEN Case Study: Improvements in the Design Process. Technica report, Advanced Technology Institute, June 1998

[19] Pratt M.J. A Shape Modeling API for the STEP Standard. Technical report, National Institute of Standards and Technology, September 2000.

[20] Gattamelata, Davide, Pezzuti, Eugenio; Valentini, Pier Paolo , "using application programming interface to integrate reverse engineering methodologies into solidworks", University of Rome Tor Vergata

[21] Maclean Andrew J.P., "Parametric Equations for Surfaces", pp 29, 2006

[22] Maclean Andrew J.P. , "Parametric Equations for Surfaces", Centre for Autonomou

[23] Tong Wu, Edmund H and M. Cheung "Enhanced STL" Int J Adv Manuf Technol 29, pp. 1143–1150,2006.

[24] S. D. Porumbescu, B. Budge, L. Feng, and K. I. Joy. "Shell maps" ACM Trans. on Graphics, 23(3), pp. 626-633,2005,

[25] Xiuzhi Qu and Brent Stucker, "A 3D surface offset method for STL-format models", Rapid Prototyping Journal Volume 9, Number 3, pp. 133–141,2003.

[26] Su-Jin Kim and Min-Yang Yang, "Offset Triangular Mesh Using the Multiple Normal Vectors of a Vertex", Journal of the Korean Society of Precision Engineering, Vol. 22, No. 9, September 2005.

[27] Matteo Malosio, Nicola Pedrocchi and Lorenzo Molinari Tosatti, "Algorithm to Offset and Smooth Tessellated Surfaces", Computer-Aided Design and Applications , pp. 351-363, 2009.

[28] D. Dragomatz and S. Mann, "A Classified Bibliography of Literature on NC Tool Path Generation", Journal of Computer-Aided Design, vol. 27, pp. 239–247 March 1997.

[29] Catania, G ‘A computer-aided prototype system for NC rough milling of freeform shaped mechanical part-pieces’ Comput. in Indust. Vol 20 No 2 , pp 275–93, 1992.
[30] Joe Fields, Constructing boy’s surface, Maclean Andrew J.P., “Parametric Equations for Surfaces”, 2006
[31] Amy Hawthorne and Jenny Posson-Brown, Boy’s surface Maclean Andrew J.P., “Parametric Equations for Surfaces”, 2006.,
[32] P. Gray, S. Bedi, F. Ismail, “Arc-intersect method for 5-axis tool positioning”, Computer-Aided Design 37 (2005).
[33] D. Roth, S. Bedi, and F. Ismail, “Surface Swept by a Toroidal Cutter during 5- Axis Machining of Curved Surfaces”, Computer-Aided Design, vol. 33, no. 1 pp. 57-63,2001.
[34] B. K. Choi, D. H. Kim and R. B. Jerard, “C-space approach to tool-path generation for die and mold machining”, Computer-Aided Design, vol. 29, pp. 657–669, 1997.
[35] G. W. Vickers and K. Quan, "Ball-mills versus end-mills for curved surface machining", ASME Journal of Engineering for Industry, (22),pp 111,1989.
[36] D. Roth, S. Bedi, and F. Ismail, “Surface Swept by a Toroidal Cutter during 5- Axis Machining of Curved Surfaces”, Computer-Aided Design, vol. 33, no. 1,pp. 57-63, 2001.
[37] B. K. Choi, “Surface Modeling and Machining”, Advances in Industrial Engineering, Elsevier, New York, vol. 11, 1991.
[38] Bedi, S. Gravelle and Y. H. Chen, "Principle curvature K alignment technique for machining complex surfaces", ASME Journal of Engineering for Industry, 2001.
[39] T. Saito and T. Takahashi, “NC machining with G-buffer method”, Journal of Computer Graphics, vol. 25, pp. 207–216,1991.