Handbook on Computer Aided Design

- R.B. Agarwal

Handbook on Computer Aided Design

Handbook on Computer Aided Design

– Dr. Raghu B. Agarwal

Edited by: Ajay Yadiki

This page intentionally left blank

Contents

Finite Elemental Analysis

This page intentionally left blank

Note to readers

This book is dedicated to work of DR. Raghu B. Agarwal. The contents of the book are available over internet by name "lecture notes by R.B. Agarwal on Computer Aided Designing for Mechanical Engineering". The editor tried to collect all the available resources to make this a single book. The lecture notes was good but it was not a complete book but a collection of files in a scattered form. The editor collected all the information and made this book "Handbook on Computer Aided Design – R.B. Agarwal". In this way this book could easily reach students who want to learn fundamentals of Computer Aided Designing for free. This book is not copyrighted but the contents in the book are copyright protected.

About CAD:
Computer Aided Designing is a major study for engineers from different disciplines, it is widely used and adopted by many industries and companies and marketed their software and hardware globally. The fundamentals of Computer Aided Designing stands still and are very simple to understand the basics. This is the key subject to be studied by a design engineer irrespective of his field of work. Computer Aided Designing is used in many sectors like Automation & Industries, Digital Manufacturing, Digital Design of Electronic Circuits, Architecture & Structural Engineering, Electrical Systems & Wiring, Automotive and Vehicles, Consumer Products, Space etc.

Due to its wide area of applications, this subject is made mandatory in many colleges and universities worldwide but still there exists no book as simple as this book that deals with fundamentals concepts involved in CAD.

About Author:
DR. Raghu B. Agarwal
Professor & Graduate Program Advisor
Department of Mechanical Engineering
San Jose State University
One Washington Square
San Jose, California 95192
Email: raghu.agarwal@sjsu.edu
(Please mail in official hours only)

This page intentionally left blank

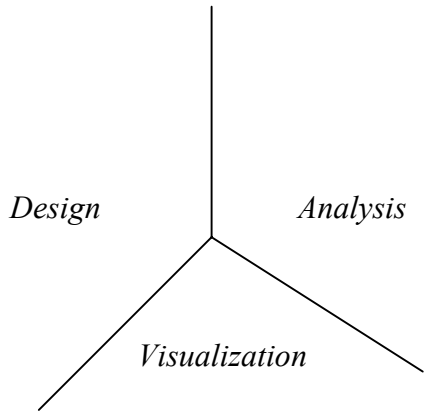
CHAPTER 1

INTRODUCTION

1.1 Introduction

In general, a Computer Aided Design (CAD) package has three components: a) Design, b) Analysis, and c) Visualization, as shown in the sketch. A brief description of these components follows.

a) **Design**: Design refers to geometric, i.e., 2-D and 3-D, including, drafting, part creation, creation of drawings with various views of the part, assemblies of the parts, etc.
b) **Analysis**: Analysis refers to finite element analysis, optimization, and other number crunching engineering analyses. In general, a geometric model is first created and then the model is analyzed for loads, stresses, moment of inertia, and volume, etc.
c) **Visualization**: Visualization refers to computer graphics, which includes: rendering a model, creation of pie charts, contour plots, shading a model, sizing, animation, etc.

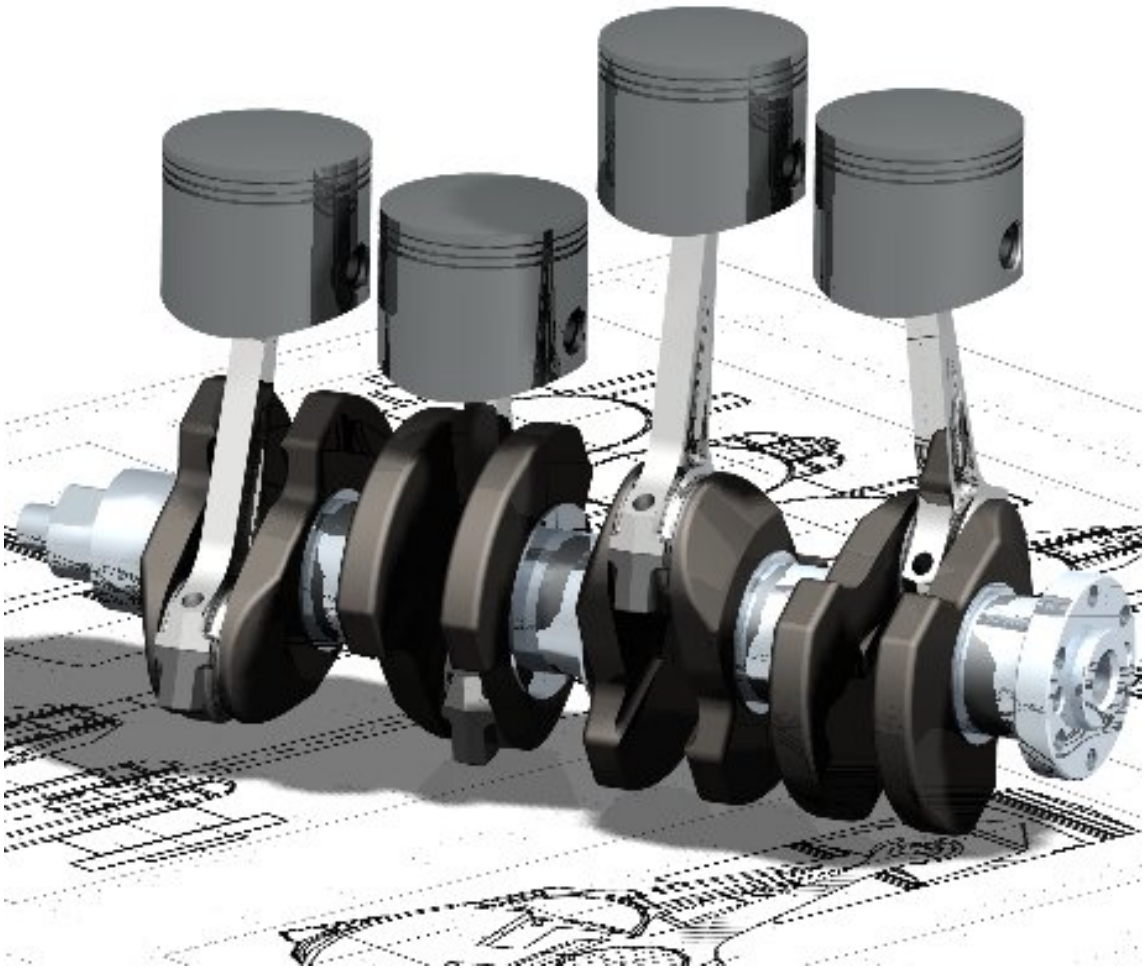


Components of Computer Aided Design

Each of these three areas has been extensively developed in the last 30 years. Several books are written on each of these subjects and courses are available through the academic institutions and the industry.

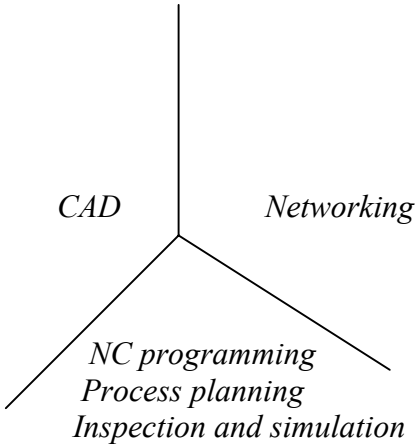
Most commercial CAD packages (software) consist of only a single component: design or analysis or visualization. However, a few of the vendors have developed an integrated package that includes not only these three areas, but also includes the manufacturing software (CAM). Due to the large storage requirement, integrated packages use either an UNIX workstation or a mainframe platform, and not the popular PC platform. With the improvement in PC computing speed, it's only a matter of time before we see an integrated package run on a PC. CAD has revolutionized the modern engineering practice; small and large companies use it alike, spending several billion dollars for the initial purchase or lease alone. CAD related jobs are high in demand and the new graduates have advantage over their senior colleagues, as they are more up to date and more productive.

An example of Computer Aided Design is shown below.



1.2 Computer Aided Manufacturing (CAM)

CAM is the next stage of CAD. A part created in CAD can be downloaded and manufactured, without a human hand touching the part. The process is called CAM, and involves CAD, Networking, and NC programming, as shown below.



Components of Computer Aided Manufacturing

1.3 Concurrent Engineering

Concurrent Engineering is another powerful CAD concept that has evolved in the 90's. According to this concept, there is an instantaneous communication between the designer, analyst, and manufacturing. Changes made at any of these work centers are immediately passed on to the others and the product is modified without delay. Often, the customer, management, and the marketing people join in and become part of the process. Concurrent engineering saves the valuable time and helps get the product out in the market quicker. Products that use to take years from the date of its concept to the actual production now take only a few weeks, and the final product is better and cost-effective.

Some large organizations have invested in Rapid Prototyping process. In this process, the part is created by a CAD package and downloaded into the rapid prototyping machine; the machine immediately manufactures the part, using a plastic material. This is a good example of concurrent engineering, sometimes referred as Art to Part concept.

1.4 CAD/CAM History

The concept of CAD and CAM is relatively new. The usage is linked with the development of computers. The actual application of CAD/CAM in industry, academia and government is only approximately 30 years old. Formal courses in CAD and Finite Element Analysis (FEA) were introduced in 1970's. The major application thrust of CAD came in 1980's, with the availability of PCs and workstations. In its early stage of usage, very few engineering companies could afford the expense of mainframe computers; however, PCs and workstations have evolved into affordable and adequate platform to support comprehensive CAD packages that initially were designed to run on the mainframe platform. A brief history of the evolution of CAD/CAM, according to the decade and the major CAD/CAM developments, is outlined below.

1960's

- Development in Interactive computer graphics research
- Sketchpad system developed by Ivan Sutherland in 1962
- CAD term coined
- First major commercial CAD/CAM software available: CADAM by Lockheed, in 1965
- Bell Telephone's - Graphics 1 remote display system developed

1970's

- Application of CAM in government, industry and academia
- National organization formed
- Beginning of usage of computer graphics
- Turnkey system available for drafting
- Wireframe and surface software became available
- Mass property calculation and FEA software became available
- NC tape generating, verification, and integrated circuit software became available

1980's

- CAD/CAM used for engineering research and development
- New CAD/CAM theories and algorithms developed
- Integration of CAD/CAM
- Solid software became available
- Use of PCs and workstation began

1990's

- Concept of concurrent engineering developed
- Increased use of CAD/CAM on PCs and worksations
- Improvements in hardware and software

1.5 CAD Hardware

There are basically two types of devices that constitute CAD hardware: a) Input devices, and b) Output devices. A brief description follows.

1.5.1 Input Devices

These are the devices that we use for communicating with computer, and providing our input in the form of text and graphics. The text input is mainly provided through keyboard. For graphic input, there are several devices available and used according to the work environment. A brief description of these devices is given here.

Mouse: This is a potentiometric device, which contains several variable resistors that send signals to the computer. The functions of a mouse include locating a point on the screen, sketching, dragging an object, entering values, accepting a software command, etc. Joystick and trackballs are analogous to a mouse device, and operate on the same principle.

Digitizers: Digitizers are used to trace a sketch or other 2-D entities by moving a cursor over a flat surface (which contains the sketch). The position of the cursor provides a feedback to the computer connected with the device. There are electrical wires embedded in orthogonal directions that receive and pass signals between the device and the computer. The device is basically a free moving puck or pen shaped stylus, connected to a tablet.

Light Pens: Lockheed's CADAM software utilized this device to carry out the graphic input. A light pen looks like a pen and contains a photocell, which emits an electronic signal. When the pen is pointed at the monitor screen, it senses light, which is converted to a signal. The signal is sent to the computer, for determination of the exact location of the pen on the monitor screen.

Touch Sensitive Screens: This device is embedded in the monitor screens, usually, in the form of an overlay. The screen senses the physical contact of the user. The new generation of the Laptop computers is a good example of this device.

Other Graphic Input Devices: In addition to the devices described above, some CAD software will accept input via Image Scanners, which can copy a drawing or schematic with a camera and light beam assembly and convert it into a pictorial database.

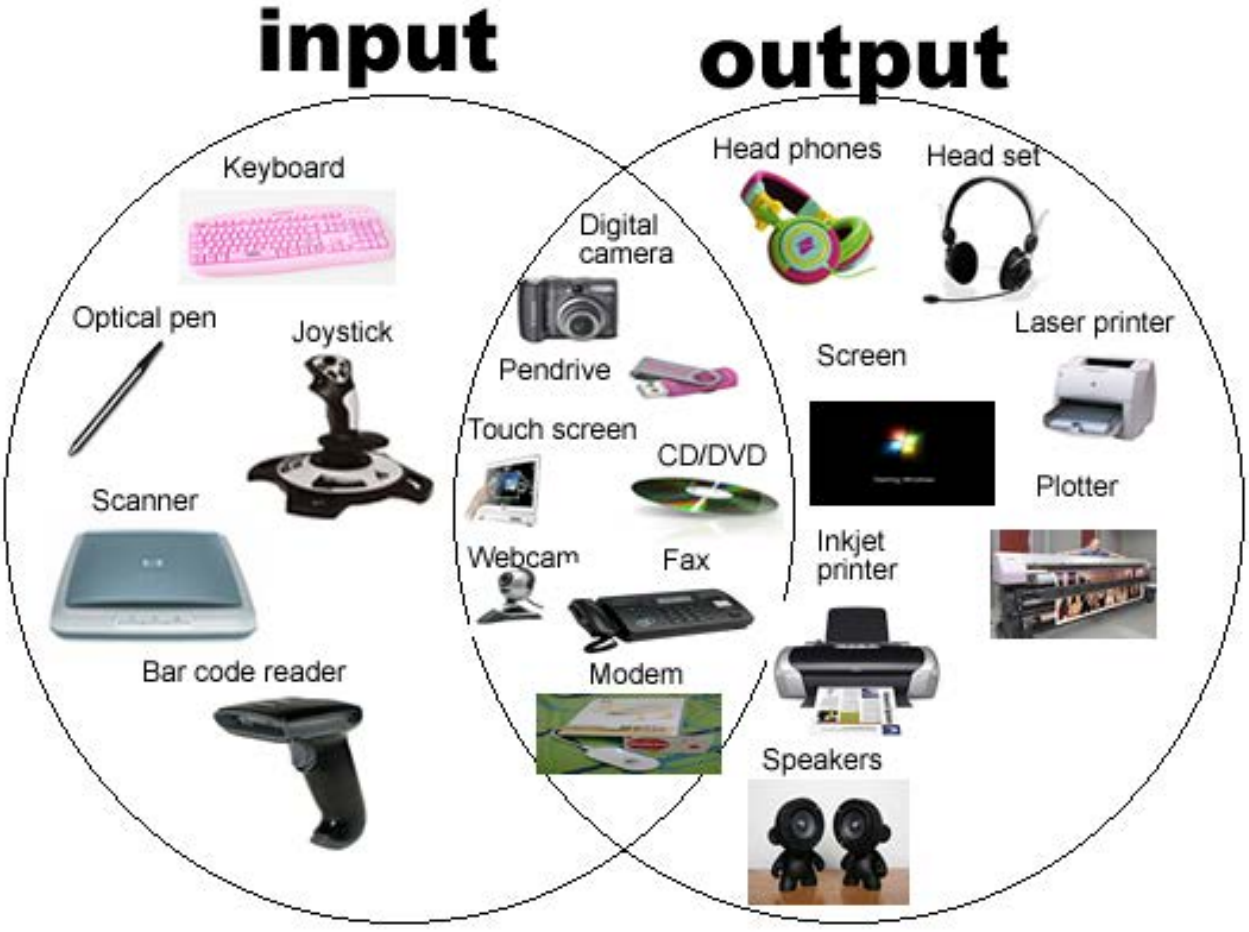
The devices just described are, in general, independent of the CAD package being used. All commercial CAD software packages contain the device drivers for the most commonly used input devices. The device drivers facilitate a smooth interaction between our input, the software, and the computer. An input device is evaluated on the basis of the following factors:

- Resolution
- Accuracy

- Repeatability
- Linearity

1.5.2 Output Devices

After creating a CAD model, we often need a hard copy, using an output device. Plotters and printers are used for this purpose. A plotter is often used to produce large size drawings and assemblies, where as, a laser jet printer is adequate to provide a 3-D view of a model. Most CAD software require a plotter for producing a shaded or a rendered view.

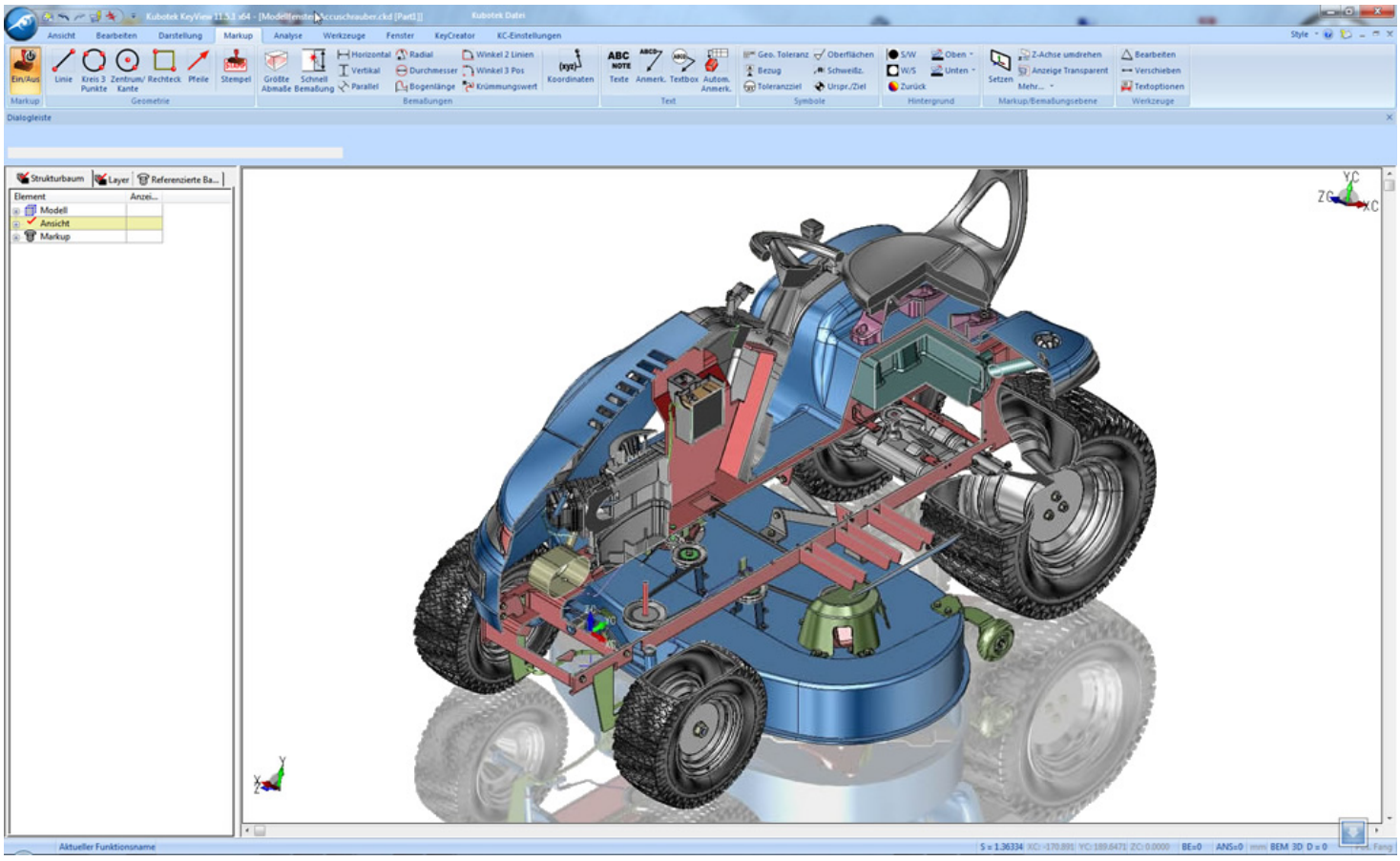


1.6 CAD Software

CAD software are written in FORTRAN and C languages. FORTRAN provides the number crunching, where as, C language provides the visual images. Early CAD packages were turnkey systems, i.e., the CAD packages were sold as an integrated software and hardware package, with no flexibility for using second vendor hardware (1970s and 80s). These systems were based on 16-bit word, and were incapable of networking. The modern CAD software utilizes the open architecture system, i.e., software vendors do not design and manufacture their own hardware. Third party software can be used to augment the basic CAD package. Most popular CAD package will facilitate integration of the Finite Element Analysis and other CAD software from more than one vendor. For example, IDEAS preprocessor can work with almost all the FEA packages for pre and post analyses.

Networking is an important consideration in applications of CAD software. A model created by one engineer must be readily accessible to others in an organization, which is linked by a LAN or other means. The designer, analyst, management, marketing, vendor, and others generally share a model. This is the concurrent engineering in action, mentioned earlier.

an example of CAD software is shown below,



1.7 CAD Platform

In general, we can run CAD software on three different CAD platforms: Mainframe, Workstation, and PC. When the CAD programs first became available, they could only be run on a mainframe computer. However, as the PCs have become faster and cheaper, almost all the CAD vendors have introduced a version of their CAD software that will effectively run on a Pentium or higher computer. Currently, the most popular platforms are PCs and Workstations. Popularity of Workstations stems from their ability to network easily with other computers, and also, due to their large memory storage capability. However, PC platform is still the most preferred medium for most engineers. Increasing popularity of the PC platform can be attributed to several factors, including, total user control, the speed, capability of storing large memory, ease of hardware upgrading and maintenance, and the overall reasonable cost.

CAD PLATFORMS

MAINFRAME	WORKSTATIONS	PCs
Large Data storage	Medium size data storage	Limited data storage
Networked	Networked	Can be networked
Expensive	Relatively inexpensive	Inexpensive
Need interface language	Runs on Unix	Run on MS-Windows
No user control	No user control	User controlled
Good for Large corporations	Good for small companies	Good for all users

1.8 CAD Evaluation Criteria

In the current CAD market, ProE and AutoCAD are arguably the most dominating CAD software. AutoCAD is basically a 2-D program, with some capability to create 3-D models, where as, ProE is a truly 3-D CAD package. Besides these software, there are several other CAD software, listed in the previous section (Sec 1.3), that have sales exceeding $100 millions. No one CAD package is suitable for all the CAD users in the world. The product we are designing dictates the type of CAD package we need. A good CAD package includes good software, as well as, a compatible hardware. Following is a brief description of the general criteria for evaluating a CAD package.

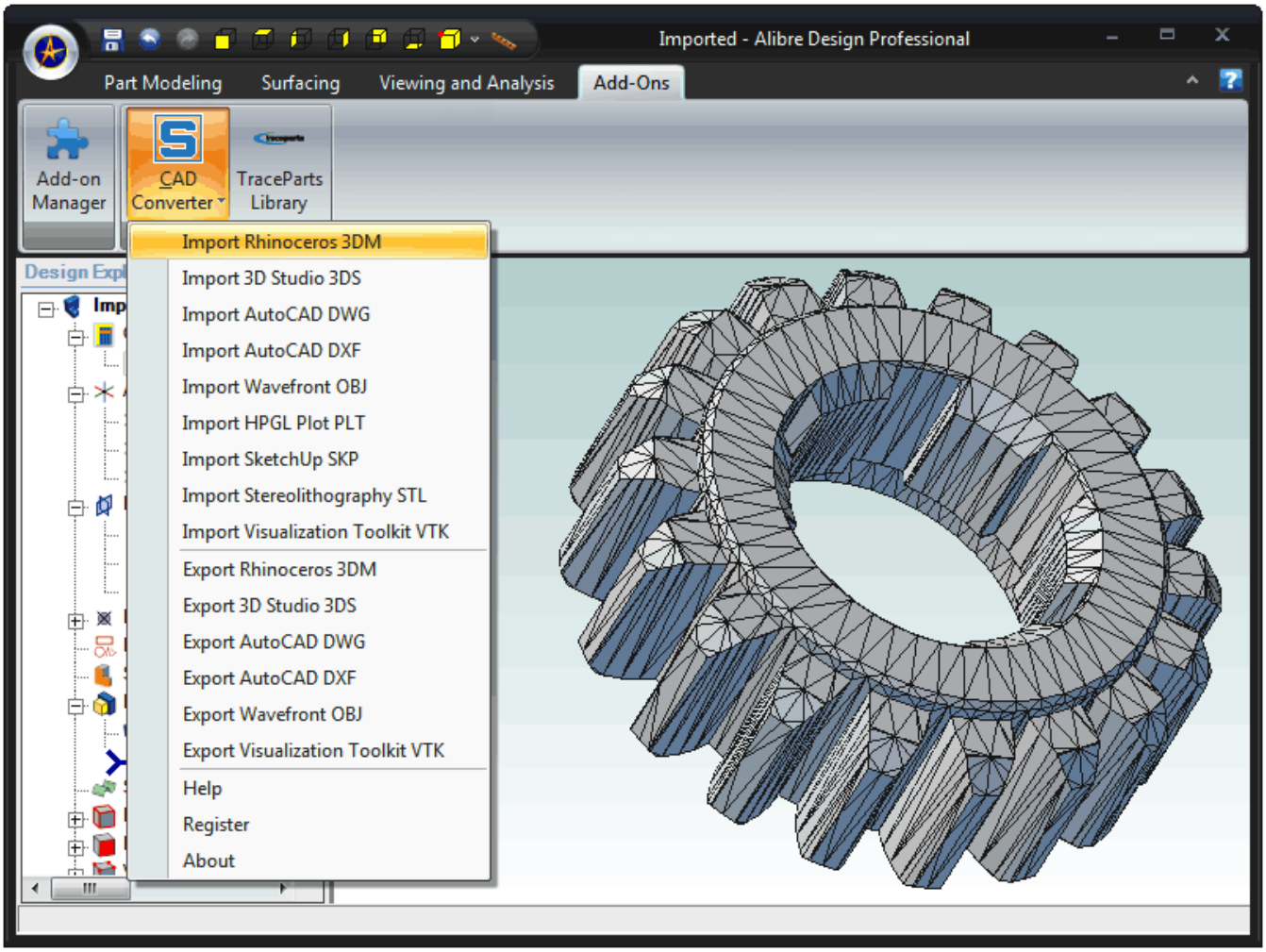
Hardware: Most desirable features in a good hardware are:

- Open architecture
- High speed, large storage
- Compact size
- Inexpensive components
- Inexpensive upgrading

Software: In general, the most comprehensive software are written to satisfy almost all the modelling needs of a modeler, consequently, the software tend to be very complex and hard to learn. To create a simple model, we go through several unnecessary steps, and lack the intuitiveness of a simple, straightforward program. ProE is a good example, where we have to go through several layers of menus to create a simple solid. On the other hand, if we were to use a simpler CAD program, the same solid can be created by only a few simple commands. There are several other factors that we should consider when evaluating software. Following is a brief description of these factors.

- **Operating System**: Unix or Windows/NT. PCs in general use Microsoft Windows, where as, operating system for Workstations is Unix. For a large organization, Workstations are preferable.
- **User Interface**: Most popular CAD software have menu driven commands, which is preferable to the old system of non-menu driven, where user interface was completely by responding to software commands. The most popular CAD programs work with menu driven interface, with some input/action required through command prompts.
- **Documentation and Support**: Learning a software can be very difficult if the software lacks good documentation. Documentation usually comes in the form of a user's manual, a tutorial book, commands manual, and on-line help. The recent trend is to provide access to the above-mentioned documentation through the Internet, or provide the manuals on a CD ROM. Some CAD vendors provide additional technical support help through phone – ProE is a very good example of this type of support.
- **Maintenance**: Cost of the hardware and software upgrades can significantly impact the small and medium size companies' decision to choose one software over the others. Most CAD vendors go through an upgrade, on the average, every two years. Usually, hardware upgrade is not as frequent.

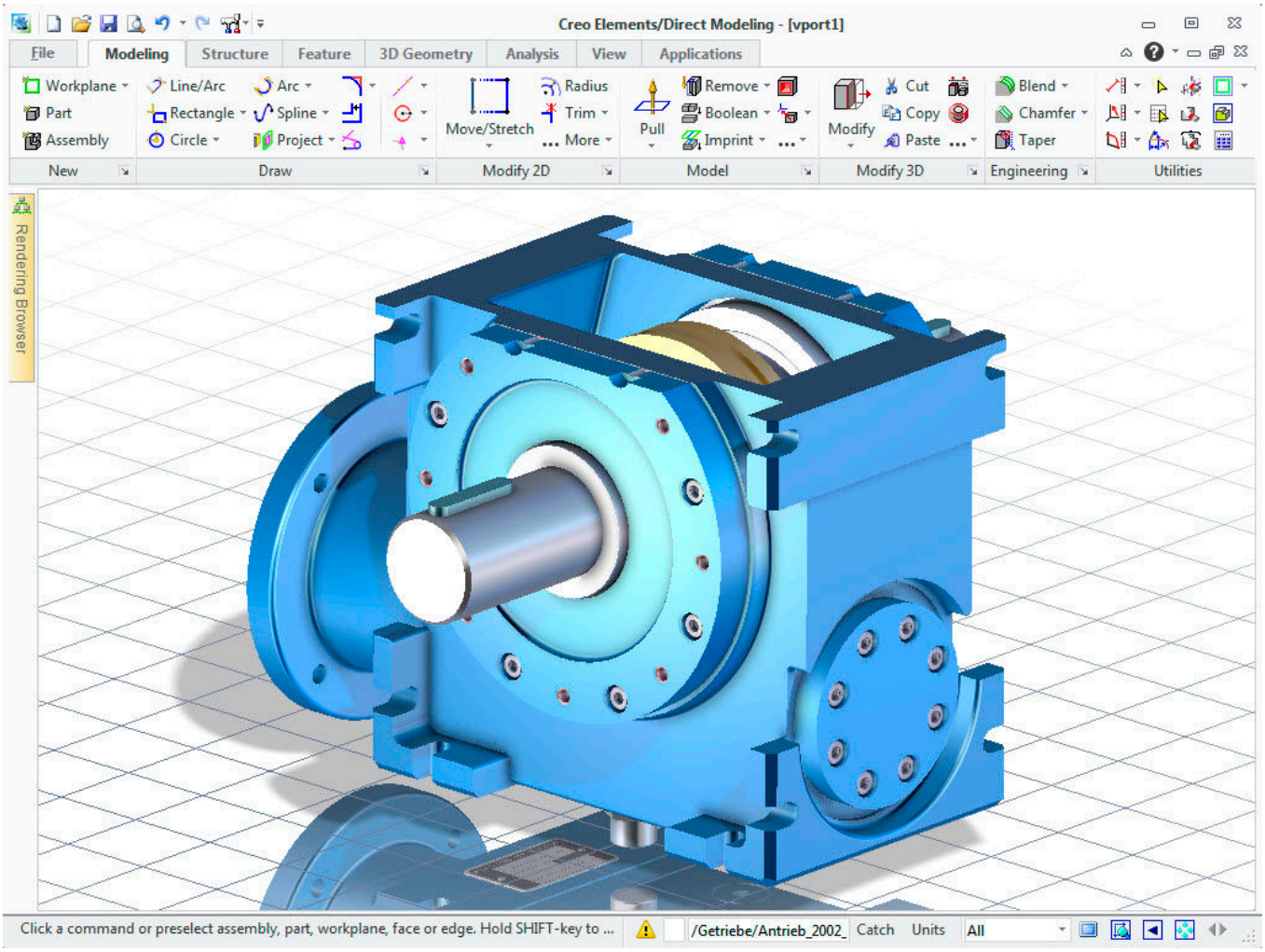
- **modelling Capabilities**: In, general, a CAD software can be classified as either a 2-D or a 3-D program. If we were basically involved in 2-D drawings, any well established 2-D software, similar to AutoCAD would suffice our needs. On the other hand, if we need to create 3-D models and assemblies, we will be better off with a 3-D molder – ProE, SOLIDWORKS, etc.
- **Ease of modelling**: As a rule-of-thumb, a general, all-purpose type CAD software is much more complex and difficult to learn than a special purpose CAD package.
- **Interface with other CAD Packages and Data Transferability**: A CAD package is used to create models that will be used for analysis, manufacturing, or some other applications. Therefore, a CAD software should be capable of transferring and accepting files from other CAD or CAM programs, without this provision, the CAD program has only a very limited use.
- **Design Documentation**: Besides creating a model, the software should be capable of creating drawings, assemblies, dimensioning, various views (isometric, orthogonal, etc.), labels and attributes, etc.



1.9 Mechanical Engineering Applications of CAD

Following is a brief description of the applications of CAD in mechanical engineering.

- **Two Dimensional Drafting**: This is the most common use of a CAD package. 2-D drawings are used for manufacturing a product.
- **Report Generating**: To generate reports and bill of materials. Spreadsheets and word-processors can be linked to provide a report writing facility.
- **3-D modelling**: To create the wireframe, surface and solid models. The 3-D models are for concept verification, manufacturing, FEA, etc.
- **Finite Element Analysis**: FEA package is used for pre-processing, analysis, and post-analysis of structures. For this application, a CAD package contains both the modeling and analysis modules.
- **Manufacturing**: manufacturing software is usually called CAM, and contains CAD software as one of the components. CAM software provides capabilities of carrying out 2 and 3-axes machining.



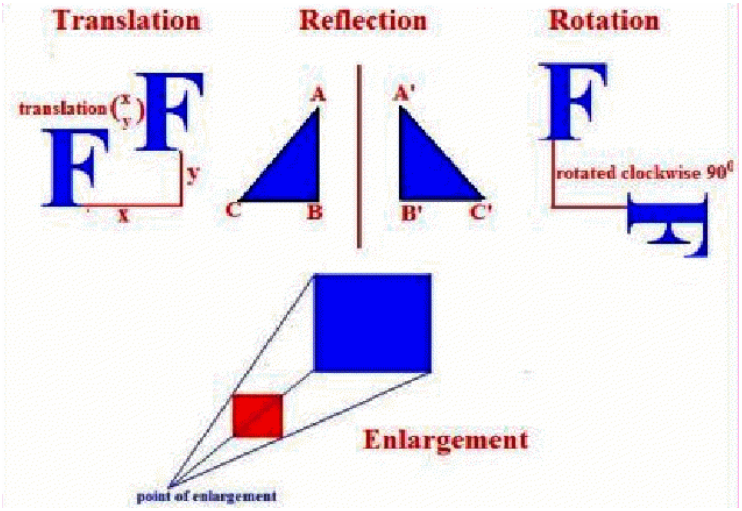
CHAPTER 2

TWO-DIMENSIONAL TRANSFORMATION

2.1 Introduction

As stated earlier, Computer Aided Design consists of three components, namely, Design (Geometric), Analysis (FEA, etc), and Visualization (Computer Graphics). Geometric provides a mathematical description of a geometric object - point, line, conic section, surface, or a solid. Visualization dea .ls with visual effects of e.eg., creation of pie charts, contol plots, shading, animation, etc. Computer graphics provides visual displays and manipulations of objects, e.g., transformation, editing, printing, etc. Fortran and visual C languages are used to effect these operations. Transformation is the backbone of computer graphics, enabling us to manipulate the shape, size, and location of the object. It can be used to effect the following changes in a geometric object:

- Change the location
- Change the Shape
- Change the size
- Rotate
- Copy
- Generate a surface from a line
- Generate a solid from a surface
- Animate the object



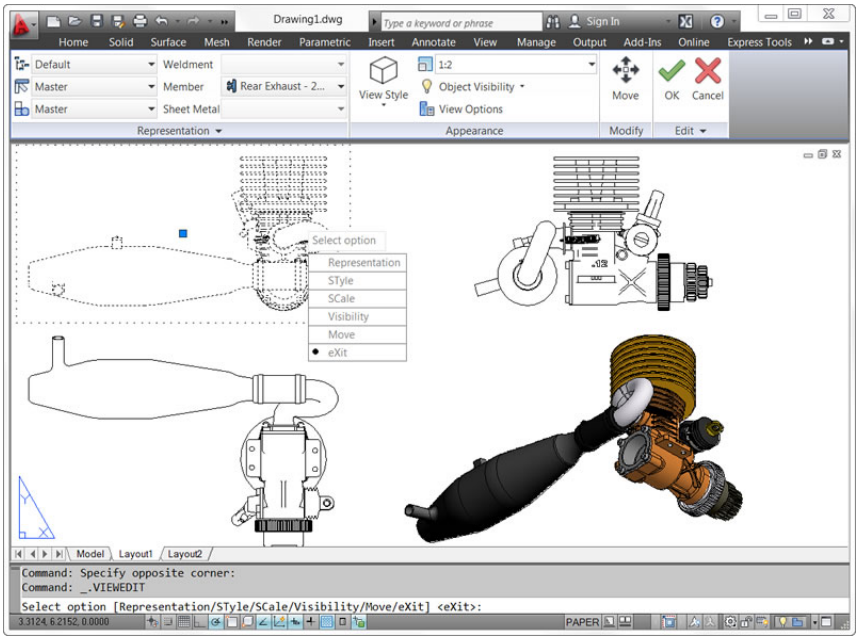
2.2 Two-Dimensional Transformation

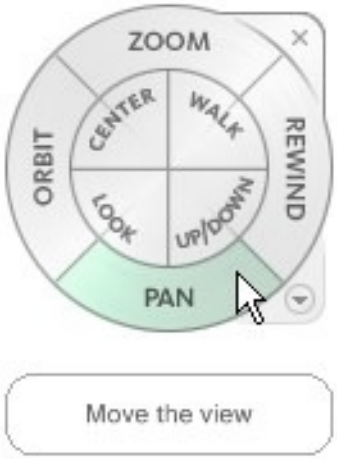
Geometric transformations have numerous applications in geometric, e.g., manipulation of size, shape, and location of an object. In CAD, transformation is also used to generate surfaces and solids by sweeping curves and surfaces, respectively. The term 'sweeping' refers to parametric transformations, which are utilized to generate surfaces and solids. When we sweep a curve, it is transformed through several positions along or around an axis, generating a surface. The appearance of the generated surface depends on the number of instances of the transformation. A parameter t or s is varied from 0 to 1, with the interval value equal to the fraction of the parameter. For example, to generate 10 instances, the parameter will have a value t/10 or s/10. To develop an easier understanding of transformations, we will first study the two-dimensional transformations and then extend it to the study of three-dimensional transformations. Until we get to the discussion of surfaces and solids, we will limit our discussion of transformation to only the simple cases of scaling, translation, rotation, and the combinations of these. Applications of transformations will become apparent when we discuss the surface and solid modelling.

There are two types of transformations:

Modelling Transformation: this transformation alters the coordinate values of the object. Basic operations are scaling, translation, rotation and, combination of one or more of these basic transformations. Examples of these transformations can be easily found in any commercial CAD software. For instance, AutoCAD uses SCALE, MOVE, and ROTATE commands for scaling, translation, and rotation transformations, respectively.

Visual Transformation: In this transformation there is no change in either the geometry or the coordinates of the object. A copy of the object is placed at the desired sight, without changing the coordinate values of the object. In AutoCAD, the ZOOM and PAN commands are good examples of visual transformation.





2.3 Basic modelling Transformations

There are three basic transformations: Scaling, Translation, and Rotation. Other transformations, which are modification or combination of any of the basic transformations, are Shearing, Mirroring, copy, etc.

Let us look at the procedure for carrying out basic transformations, which are based on matrix operation. A transformation can be expressed as

$$[P^*] = [P]\,[T]$$

where, $[P^*]$ is the new coordinates matrix
$[P]$ is the original coordinates matrix, or points matrix
$[T]$ is the transformation matrix

With the z-terms set to zero, the P matrix can be written as,

$$[P] = \begin{pmatrix} x_1 & y_1 & 0 \\ x_2 & y_2 & 0 \\ x_3 & y_3 & 0 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 0 \end{pmatrix} \qquad (2.1)$$

The size of this matrix depends on the geometry of the object, e.g., a point is defined by a single set of coordinates (x_1, y_1, z_1), a line is defined by two sets of coordinates (x_1, y_1, z_1) and (x_2, y_2, z_2), etc. Thus a point matrix will have the size 1x3, line will be 2x3, etc.

A transformation matrix is always written as a 4x4 matrix, with a basic shape shown below,

$$[T] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (2.2)$$

Values of the elements in the matrix will change according to the type of transformation being used, as we will see shortly. The transformation matrix changes the size, position, and orientation of an object, by mathematically adding, or multiplying its coordinate values. We will now discuss the mathematical procedure for scaling, translation, and rotation transformations.

2.4 Scaling

In scaling transformation, the original coordinates of an object are multiplied by the given scale factor. There are two types of scaling transformations: uniform and non-uniform. In the uniform scaling, the coordinate values change uniformly along the x, y, and z coordinates, where as, in non-uniform scaling, the change is not necessarily the same in all the coordinate directions.

2.4.1 Uniform Scaling

For uniform scaling, the scaling transformation matrix is given as

$$[T] = \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

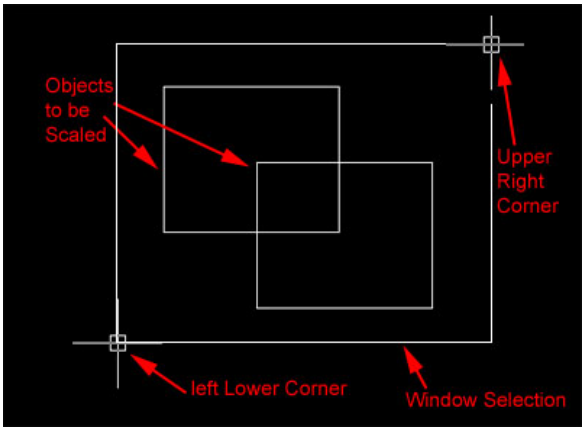
Here, s is the scale factor.

2.4.2 Non-Uniform Scaling

Matrix equation of a non-uniform scaling has the form:

$$[T] = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.4}$$

where, s_x, s_x, s_x are the scale factors for the x, y, and z coordinates of the object.



2.5 Homogeneous Coordinates

Before proceeding further, we should review the concept of homogeneous coordinate system. Since the points matrix has three columns for the x, y, and z values, and a transformation matrix is always 4x4 matrix, the two matrices are incompatible for multiplication. A matrix multiplication is compatible only if the number of columns in the first matrix equals the number of row in the second matrix. For this reason, a points matrix is written as,

$$[P] = \begin{pmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & z_n & 1 \end{pmatrix} \qquad (2.5)$$

Here, we have converted the Cartesian coordinates into homogeneous coordinates by adding a 4^{th} column, with unit value in all rows. When a fourth column, with values of 1 in each row, is added in the points matrix, the matrix multiplication between the [P] and [T] becomes compatible. The values (x_1, y_1, z_1, 1) represent the coordinates of the point (x_1, y_1, z_1), and the coordinates are called as homogeneous coordinates. In homogeneous coordinates, the points (2,3,1), (4,6,2), (6,9,3), (8,12,4), represent the same point (2,3,1), along the plane z = 1, z = 2, z = 3, and z = 4, respectively. In our subsequent discussion on transformation, we will use homogeneous coordinates.

Example 1: If the triangle A(1,1), B(2,1), C(1,3) is scaled by a factor 2, find the new coordinates of the triangle**.**

Solution: Writing the points matrix in homogeneous coordinates, we have

$$[P] = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 \\ 1 & 3 & 0 & 1 \end{pmatrix}$$

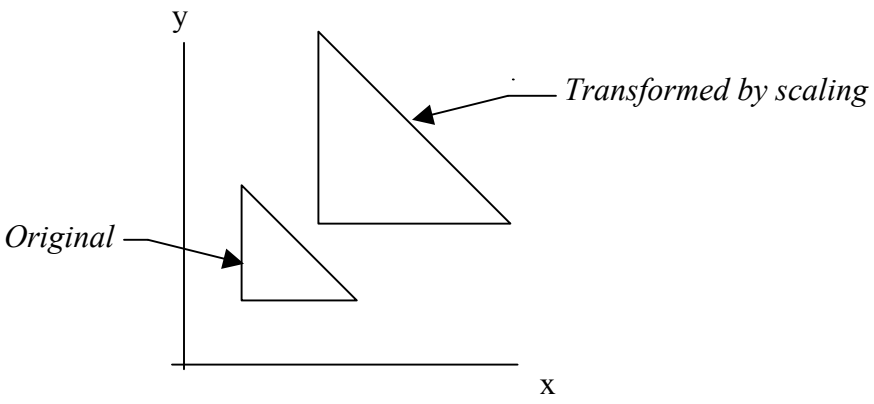
and the scaling transformation matrix is,

$$[T_s] = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The new points matrix can be evaluated by the equation

$[P^*] = [P]\,[T]$, and by substitution of the P and T values, we get

$$P^* = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 \\ 1 & 3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 0 & 1 \\ 4 & 2 & 0 & 1 \\ 2 & 6 & 0 & 1 \end{pmatrix}$$



Note that the new coordinates represent the original value times the scale factor. The old and the new positions of the triangle are shown in the figure.

2.6 Translation Transformation

In translation, every point on an object translates exactly the same distance. The effect of a translation transformation is that the original coordinate values increase or decrease by the amount of the translation along the x, y, and z-axes. For example, if line A(2,4), B(5,6) is translated 2 units along the positive x axis and 3 units along the positive y axis, then the new coordinates of the line would be

A'(2+2, 4+3), B'(5+2, 6+3) or

A'(4,7), B'(7,9).

The transformation matrix has the form:

$$[T_t] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x & y & 0 & 1 \end{pmatrix} \qquad (2.6)$$

where, x and y are the values of translation in the x and y direction, respectively. For translation transformation, the matrix equation is

$$[P^*] = [P]\,[T_t] \qquad (2.7)$$

where, $[T_t]$ is the translation transformation matrix.

***Example 2*:** Translate the rectangle (2,2), (2,8), (10,8), (10,2) 2 units along x-axis and 3 units along y-axis.

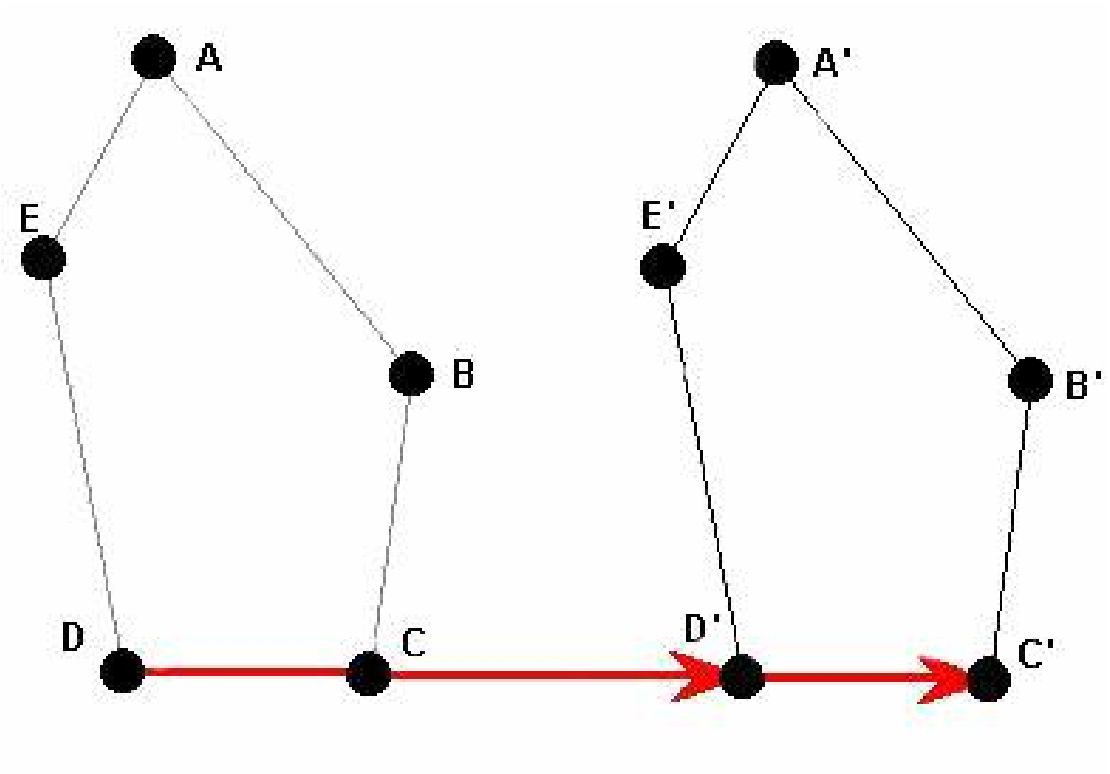
***Solution*:** Using the matrix equation for translation, we have

$[P^*] = [P]\,[T_t]$, substituting the numbers, we get

$$[P^*] = \begin{pmatrix} 2 & 2 & 0 & 1 \\ 2 & 8 & 0 & 1 \\ 10 & 8 & 0 & 1 \\ 10 & 2 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 2 & 3 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 4 & 5 & 0 & 1 \\ 4 & 11 & 0 & 1 \\ 12 & 11 & 0 & 1 \\ 12 & 5 & 0 & 1 \end{pmatrix}$$

Note that the resultant coordinates are equal to the original x and y values plus the 2 and 3 units added to these values, respectively.



2.7 Rotation

We will first consider rotation about the z-axis, which passes through the origin (0,0,0), since it is the simplest transformation for understanding the rotation transformation. Rotation about an arbitrary axis, other than an axis passing through the origin, requires a combination of three or more transformations, as we will see later.

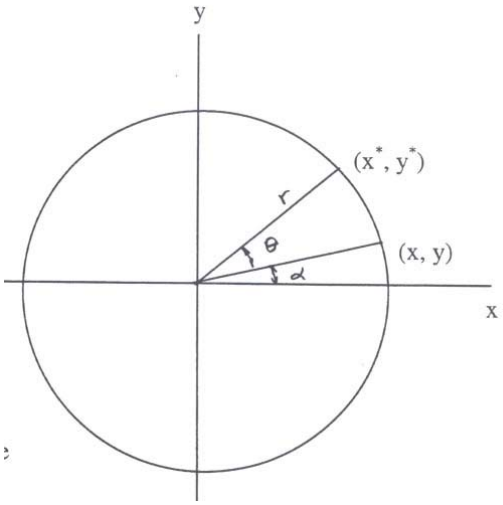
When an object is rotated about the z-axis, all the points on the object rotate in a circular arc, and the center of the arc lies at the origin. Similarly, rotation of an object about an arbitrary axis has the same relationship with the axis, i.e., all the points on the object rotate in a circular arc, and the center of rotation lies at the given point through which the axis is passing.

2.7.1 Derivation of the Rotation Transformation Matrix

Using trigonometric relations, as given below, we can derive the rotation transformation matrix. Let the point P(x, y) be on the circle, located at an angle α, as shown. If the point P is rotated an additional angle θ, the new point will have the coordinates (x*, y*). The angle and the original coordinate relationship is found as follows.

$$\left.\begin{aligned} x &= r\cos\alpha \\ y &= r\sin\alpha \end{aligned}\right\}$$ *Original coordinates of point P.*

$$\left.\begin{aligned} x^* &= r\cos(\alpha+\theta) \\ y^* &= r\sin(\alpha+\theta) \end{aligned}\right\}$$ *The new coordinates.*



where, α is the angle between the line joining the initial position of the point and the x-axis, and θ is the angle between the original and the new position of the point.

Using the trigonometric relations,
we get,

$$x^* = r(\cos\alpha \cos\theta - \sin\alpha \sin\theta) = x\cos\theta - y\sin\theta$$
$$y^* = r(\cos\alpha \sin\theta + \sin\alpha \cos\theta) = x\sin\theta + y\cos\theta$$

In matrix form we can write these equations as

$$[x^* \; y^*] = [x \quad y] \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \tag{2.8}$$

In general, the points matrix and the transformation matrix given in equation (2.8) are re-written as

$$[x^* \quad y^* \quad 0 \quad 1] = [x \quad y \quad 0 \quad 1] \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.9}$$

Thus, a point or any object can be rotated about the z-axis (in 2-D) and the new coordinates of the object found by the product of the points matrix and the rotation matrix, derived here.

2.7.2 Rotation of an Object about an Arbitrary Axis

Rotation of a geometric model about an arbitrary axis, other than any of the coordinate axes, involves several rotational and translation transformations. When we rotate an object about the origin (in 2-D), we in fact rotate it about the z-axis. Every point on the object rotates along a circular path, with the center of rotation at the origin. If we wish to rotate an object about an arbitrary axis, which is perpendicular to the xy-plane, we will have to first translate the axis to the origin and then rotate the model, and finally, translate so that the axis of rotation is restored to its initial position. If we erroneously use the equation (2.9) directly, to rotate the object about a fixed axis, and skip the translation of this point to the origin, we will in fact end up rotating the object about the z-axis, and not about the fixed axis.

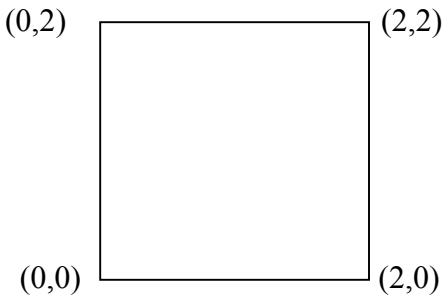
Thus, the rotation of an object about an arbitrary axis, involves three steps:

Step 1: Translate the fixed axis so that it coincides with the z-axis
Step 2: Rotate the object about the axis
Step 3: Translate the fixed axis back to the original position.

Note: When the fixed axis is translated, the object is also translated. The axis and the object go through all the transformations simultaneously.

We will now illustrate the above procedure by the following example.

Example 3*:* Rotate the rectangle (0,0), (2,0), (2, 2), (0, 2) shown below, 30^0 ccw about its centroid and find the new coordinates of the rectangle.



***Solution*:** Centroid of the rectangle is at point (1, 1). We will first translate the centroid to the origin, then rotate the rectangle, and finally, translate the rectangle so that the centroid is restored to its original position.

1. ***Translate the centroid to the origin***: The matrix equation for this step is

$$[P^*]_1 = [P]\,[T_t], \text{ where } [P] = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \end{pmatrix}$$

$$\text{and} \qquad [T_t] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & -1 & 0 & 1 \end{pmatrix}$$

2. ***Rotate the Rectangle 30^0 ccw About the z-axis*:** The matrix equation for this step is given as

$[P^*]_2 = [P^*]_1 [T_r]$, where, $[P^*]_1$ is the resultant points matrix obtained in step 1, and $[T_r]$ is the rotation transformation, where $\theta = 30^0$ ccw. The transformation matrix is,

$$[T_r]_\theta = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} .866 & .5 & 0 & 0 \\ -.5 & .866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3. ***Translate the Rectangle so that the Centroid Lies at its Original Position*:** The matrix equation for this step is

$[P^*]_3 = [P^*]_2 [T_{-t}]$, where $[T_{-t}]$ is the reverse translation matrix, given as

$$[T_{-t}] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Now we can write the entire matrix equation that combines all the three steps outlined above. The equation is,

$$[P^*] = [P] [T_t] [T_r] [T_{-t}]$$

Substituting the values given earlier, we get,

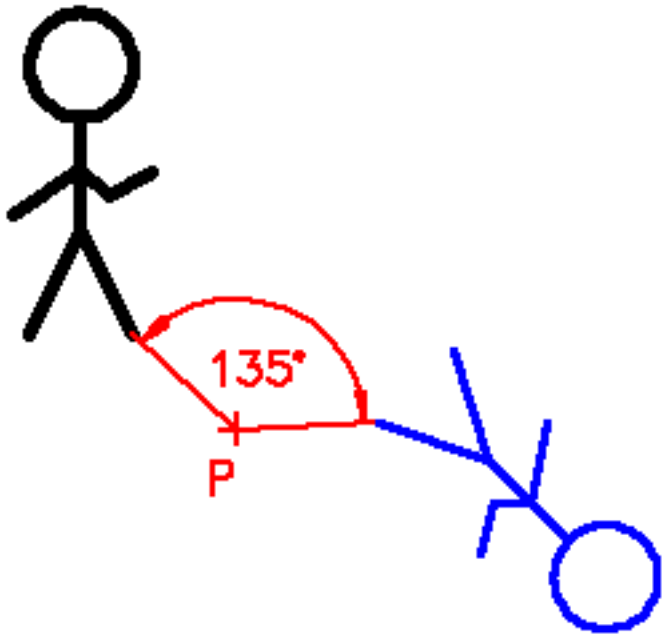
$$[P^*] = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos 30^0 & \sin 30^0 & 0 & 0 \\ -\sin 30^0 & \cos 30^0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \text{ x}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0.6340 & -0.3660 & 0 & 1 \\ -0.3660 & 1.3660 & 0 & 1 \\ 1.3660 & 2.3660 & 0 & 1 \\ -0.3660 & 1.3660 & 0 & 1 \end{pmatrix}$$

The first two columns represent the new coordinates of the rotated rectangle.

An example of Rotational Transformaion.



2.8 Combined Transformations

Most applications require the use of more than one basic transformation to achieve desired results. As stated earlier, scaling with an arbitrarily fixed point involves both scaling and translation. And rotation around a given point, other than the origin, involves rotation and translation. We will now consider these combined transformations.

2.8.1 Scaling With an Arbitrary Point

In uniform scaling, all points and their coordinates are scaled by a factor s. Therefore, unless the fixed point is located at (0, 0), it will be moved to a new location with coordinates s-times x and s-times y. To scale an object about a fixed point, the fixed point is first moved to the origin and then the object is scaled. Finally, the object is translated or moved so that the fixed point is restored to its original position. The transformation sequence is,

$$[P^*] = [P]\,[T_t]\,[T_s]\,[T_{-t}]$$

Where, $[T_t]$ is the translation transformation matrix, for translation of the fixed point to the origin,

$[T_s]$ is the scaling transformation matrix, and

$[T_{-t}]$ is the reverse translation matrix, to restore the fixed point to its original position.

Note: The order of matrix multiplication progresses from left to right and the order should not be changed.

The three transformation matrices $[T_t]\,[T_s]\,[T_{-t}]$ can be concatenated to produce a single transformation matrix, which uniformly scales an object while keeping the pivot point fixed. Thus, the resultant, concatenated transformation matrix for scaling is,

$$[T_s]_R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x & -y & 0 & 1 \end{pmatrix} \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x & y & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ x-sx & y-sy & 0 & 1 \end{pmatrix} \qquad (2.10)$$

The concatenated equation can be used directly instead of the step-by-step matrix solution. This form is preferable when writing a CAD program.

Example 4: Given the triangle, described by the homogeneous points matrix below, scale it by a factor 3/4, keeping the centroid in the same location. Use (a) separate matrix operation and (b) condensed matrix for transformation.

$$[P] = \begin{pmatrix} 2 & 2 & 0 & 1 \\ 2 & 5 & 0 & 1 \\ 5 & 5 & 0 & 1 \end{pmatrix}$$

Solution

(a) The centroid of the triangle is at,

$x = (2+2+5)/3 = 3$, and $y = (2+5+5)/3 = 4$ or the centroid is C(3,4).

We will first translate the centroid to the origin, then scale the triangle, and finally translate it back to the centroid. Translation of triangle to the origin will give,

$$[P^*]_1 = [P]\,[T_t] = \begin{pmatrix} 2 & 2 & 0 & 1 \\ 2 & 5 & 0 & 1 \\ 5 & 5 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & -4 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -1 & -2 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 \end{pmatrix}$$

Scaling the triangle, we get,

$$[P^*]_2 = [P^*]_1\,[T_s] = \begin{pmatrix} -1 & -2 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} .75 & 0 & 0 & 0 \\ 0 & .75 & 0 & 0 \\ 0 & 0 & .75 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -0.75 & -1.5 & 0 & 1 \\ -0.75 & 0.75 & 0 & 1 \\ 1.5 & 0.75 & 0 & 1 \end{pmatrix}$$

Translating the triangle so that the centroid is positioned at (3, 4), we get

$$[P^*] = [P^*]_2\,[T_{-t}] = \begin{pmatrix} -.75 & -1.5 & 0 & 1 \\ -.75 & .75 & 0 & 1 \\ 1.5 & .75 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 3 & 4 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2.25 & 2.5 & 0 & 1 \\ 2.25 & 4.75 & 0 & 1 \\ 4.5 & 4.75 & 0 & 1 \end{pmatrix}$$

(b) The foregoing set of three operations can be reduced to a single operation using the condensed matrix with x = 3, and y = 4. See equation (2.10) on page 16.

$$[P^*] = [P]\,[T_{cond}] = \begin{pmatrix} 2 & 2 & 0 & 1 \\ 2 & 5 & 0 & 1 \\ 5 & 5 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.75 & 0 & 0 & 0 \\ 0 & 0.75 & 0 & 0 \\ 0 & 0 & 0.75 & 0 \\ 3\text{-}0.75(3) & 4\text{-}0.75(4) & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 2.25 & 2.5 & 0 & 1 \\ 2.25 & 4.75 & 0 & 1 \\ 4.5 & 4.75 & 0 & 1 \end{pmatrix}$$

2.8.2 Rotation About an Arbitrary Point (in xy-plane)

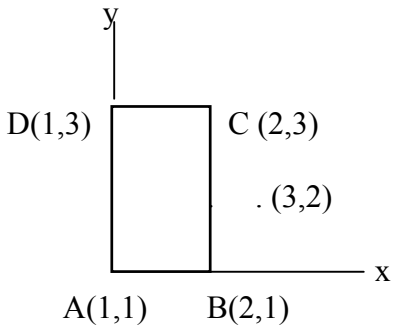
In order to rotate an object about a fixed point, the point is first moved (translated) to the origin. Then, the object is rotated around the origin. Finally, it is translated back so that the fixed point is restored to its original position. For rotation of an object about an arbitrary point, the sequence of the required transformation matrices and the condensed matrix is given as,

$[T_{cond}] = [T_t]\,[T_r]\,[T_{-t}]$ or

$$[T_{cond}] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x & -y & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x & y & 0 & 1 \end{pmatrix} \qquad (2.11)$$

where, θ is the angle of rotation and the point (x, y) lies in the xy plane.

Example 5: Rotate the rectangle formed by points A(1,1), B(2,1), C(2,3), and D(1,3) 30^0 ccw about the point (3,2).



***Solution*:** We will first translate the point (3,2) to the origin, then rotate the rectangle about the origin, and finally, translate the rectangle back so that the original point is restores to its original position (3,2). The new coordinates of the rectangle are found as follows.

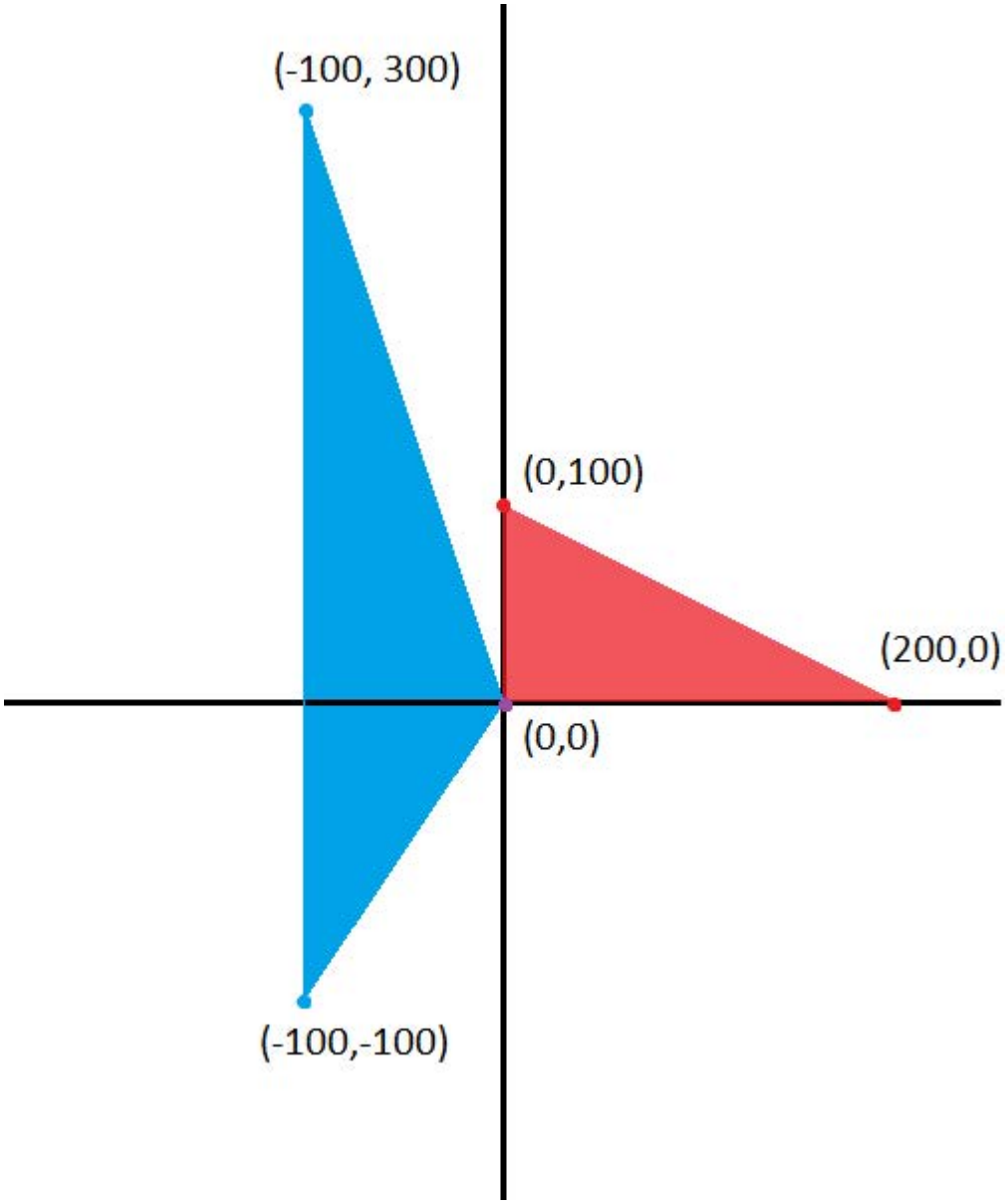
$$[P^*] = [P]\,[T_t]\,[T_r]\,[T_{-t}]$$

$$= \begin{pmatrix} 1 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 \\ 2 & 3 & 0 & 1 \\ 1 & 3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & -2 & 0 & 1 \end{pmatrix} \begin{pmatrix} .866 & .5 & 0 & 0 \\ -.5 & .866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1.77 & .13 & 0 & 1 \\ 0.77 & 1.87 & 0 & 1 \\ 1.63 & 2.37 & 0 & 1 \\ 2.63 & 0.63 & 0 & 1 \end{pmatrix}$$

These are the new coordinates of the rectangle after the rotation.

An example of Combined Transformation.



2.9 Mirroring

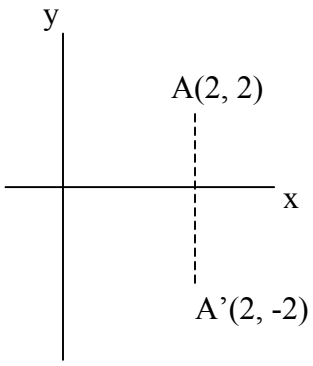
In modelling operations, one frequently used operation is mirroring an object. Mirroring is a convenient method used for copying an object while preserving its features. The mirror transformation is a special case of a negative scaling, as will be explained below.

Let us say, we want to mirror the point A(2,2) about the x-axis(i.e., xz-plane), as shown in the figure.

The new location of the point, when reflected about the x-axis, will be at (2, -2). The point matrix [P*] = [2 -2] can be obtained with the matrix transformation given below.

$$[P^*] = [2 \quad 2 \quad 0 \quad 1] \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= [2 \quad -2 \quad 0 \quad 1]$$



The transformation matrix above is a special case of a non-uniform scaling with $s_x = 1$ and $s_y = -1$. We can extend this concept to mirroring around the y, z, and any arbitrary axis, as will be explained in the following discussion.

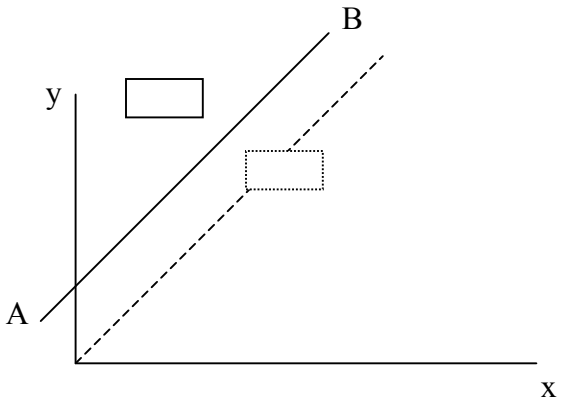
2.9.1 Mirroring About an Arbitrary Plane

If mirroring is required about an arbitrary plane, other than one defined by the coordinate axes, translation and/or rotation can be used to align the given plane with one of the coordinate planes. After mirroring, translation or rotation must be done in reverse order to restore the original geometry of the axis.

We will use the figure shown below, to illustrate the procedure for mirroring an object about an arbitrary plane. We will mirror the given rectangle about a plane passing through the line AB and perpendicular to the xy-plane. It should be noted that in each of the transformations, the plane and the rectangle have a fixed relationship, i.e., when we move the plane (or line AB, the rectangle also moves with it. A step-by-step procedure for mirroring the rectangle about the plane follows.

Note: We are using line AB to represent the plane, which passes through it. Mirroring can be done only about a plane, and not about a line.

Step 1: Translate the line AB (i.e., the plane) such that it passes through the origin, as shown by the dashed line.



Step 2: Next, rotate the line about the origin (or the z-axis) such that it coincides with x or y axes (we will use the x-axis).

Step 3: Mirror the rectangle about the x-axis.

Step 4: Rotate the line back to its original orientation.

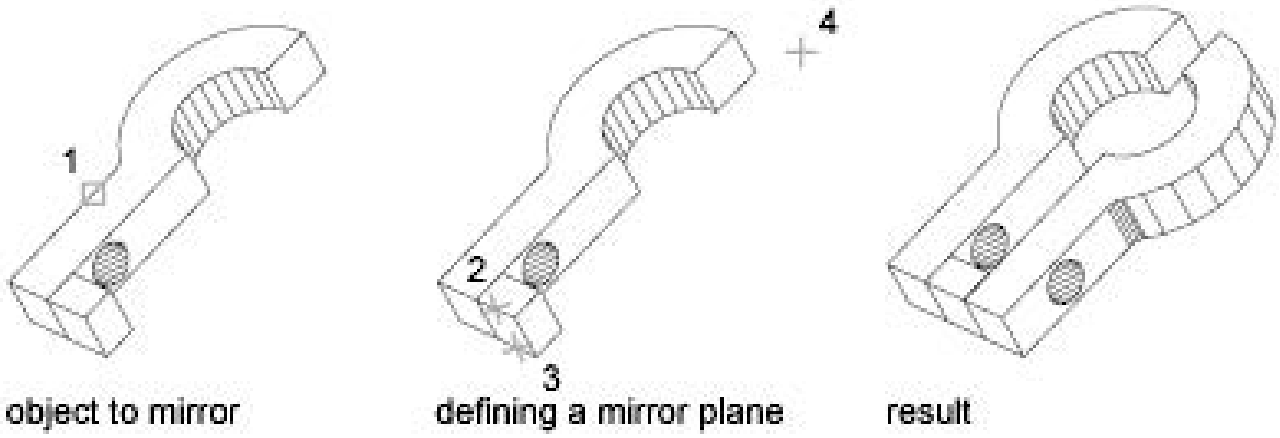
Step 5: Translate the line back to its original position.

The new points matrix, in terms of the original points matrix and the five transformation matrices is given as,

$$[P^*] = [P]\,[T_t]\,[T_r]\,[T_m]\,[T_{-r}]\,[T_{-t}]$$

(Note: A negative sign is used in the subscripts to indicate a reverse transformation).

Where, the subscripts t, r, and m represent the translation, rotation, and mirror operations, respectively.



CHAPTER 3

THREE-DIMENSIONAL TRANSFORMATION

3.1 Introduction

A three-dimensional object has a three-dimensional geometry, and therefore, it requires a three-dimensional coordinate transformation. A right-handed coordinate system is used to carry out a 3-D transformation.

The scaling and translation transformations are essentially the same as two-dimensional transformations. However, the points matrix will have a non-zero 3rd column. Additionally, the transformation matrices contain some non-zero values in the third row and third column, as shown below.

A general scaling transformation matrix is given as:

$$[T_s] = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.1)$$

Where, s_x, s_y, s_z are scale factors along x, y, and z-axes, respectively.

Translation Transformation matrix:

$$[T_t] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x & y & z & 1 \end{pmatrix} \qquad (3.2)$$

3.2 Rotation Transformation

The two-dimensional rotation transformation is in reality a special case of a three-dimensional rotation about the z-axis. We will denote it by $[T_{rz}]$, where, the second subscript z indicates rotation about the z-axis. Similarly, rotation about the x and y-axes are denoted as $[T_{rx}]$, and $[T_{ry}]$, respectively. The transformation matrices are given below.

$$[T_{rz}] = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.3}$$

$$[T_{rx}] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.4}$$

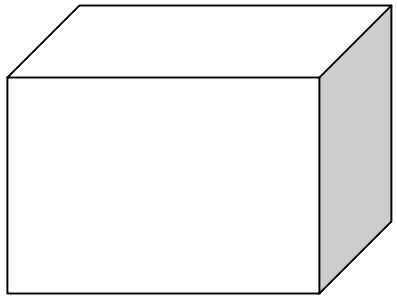
$$[T_{ry}] = \begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.5}$$

We will now present some examples of 3-D transformations.

Three-dimensional Scaling Example

Example 1: The coordinates of a cube are given below. Scale the cube uniformly by1/2.

$$[P] = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 2 & 0 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 0 & 2 & 2 & 1 \end{pmatrix}$$



***Solution*:** The new coordinates of the cube are found by the product of the points matrix and the scaling matrix,

$$[P^*] = [P]\,[T_s] = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 2 & 0 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 0 & 2 & 2 & 1 \end{pmatrix} \begin{pmatrix} ½ & 0 & 0 & 0 \\ 0 & ½ & 0 & 0 \\ 0 & 0 & ½ & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Three-Dimensional Rotation Example

Next, we will consider rotation about the x and y-axes. Note that the expression for rotation about the y-axis has a negative sign associated with the sinθ term in the first row (unlike the x and z-axes rotation, where the negative sign is associated with the sinθ term in the second and the third rows, respectively). Also, the right-hand-thumb rule must be followed when applying the value of angle θ. For example, if the angle of rotation is given as 30^0 clockwise, the value used in the Sine and Cosine terms should be (-30^0), and not $(+30^0)$.

Note: Clockwise and counter-clockwise directions are determined by the right-hand-thumb rule.

***Example 2*:** The points matrix for a wedge is given as follows. Rotate the wedge 30^0 ccw around the x-axis and then 45^0 cw around the y-axis. The points matrix is,

$$[P] = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 4 & 0 & 0 & 1 \\ 4 & 0 & 2 & 1 \\ 0 & 3 & 0 & 1 \\ 0 & 3 & 2 & 1 \end{pmatrix}$$

***Solution*:** First, we will rotate the wedge around the x-axis, and then about the y-axis.

Rotation about the x-axis,

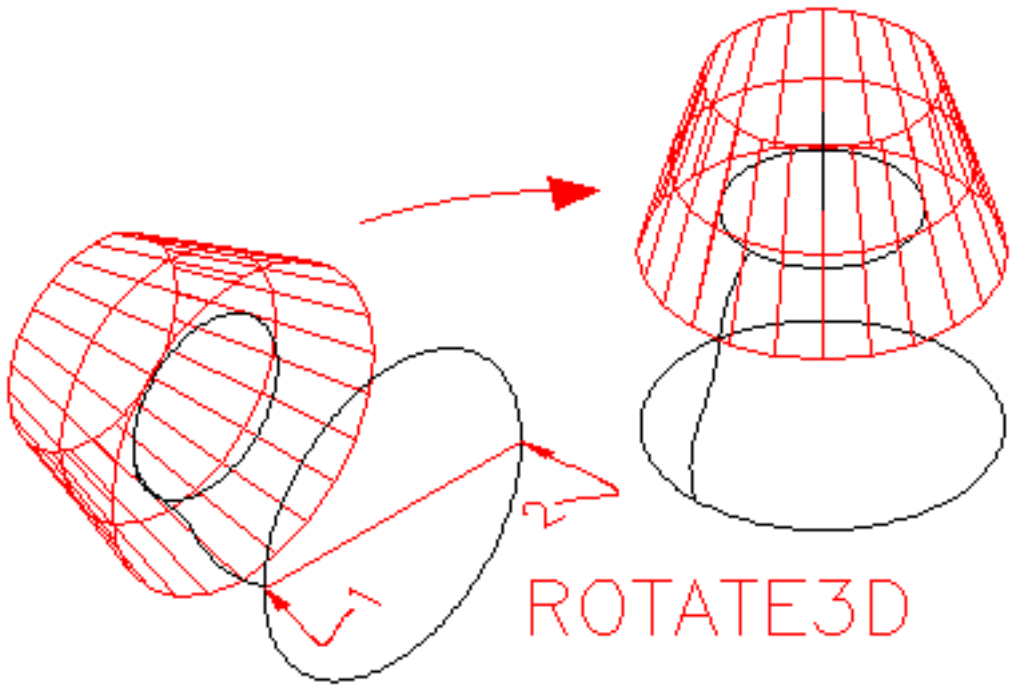
$$[P^*]_x = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 4 & 0 & 0 & 1 \\ 4 & 0 & 2 & 1 \\ 0 & 3 & 0 & 1 \\ 0 & 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(30^0) & \sin(30^0) & 0 \\ 0 & -\sin(30^0) & \cos(30^0) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Next, we rotate the wedge about the y-axis,

$$[P^*] = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 4 & 0 & 0 & 1 \\ 4 & 0 & 2 & 1 \\ 0 & 3 & 0 & 1 \\ 0 & 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(30^0) & \sin(30^0) & 0 \\ 0 & -\sin(30^0) & \cos(30^0) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times$$

$$\begin{pmatrix} \cos(-45^0) & 0 & -\sin(-45^0) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(-45^0) & 0 & \cos(-45^0) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ -1.22 & -1 & 1.22 & 1 \\ 2.82 & 0 & 2.82 & 1 \\ 1.6 & -1 & 4.05 & 1 \end{pmatrix}$$

Note that, rotation about the axis is positive and hence, $+30^0$. Where as rotation about the y-axis is negative -45^0.



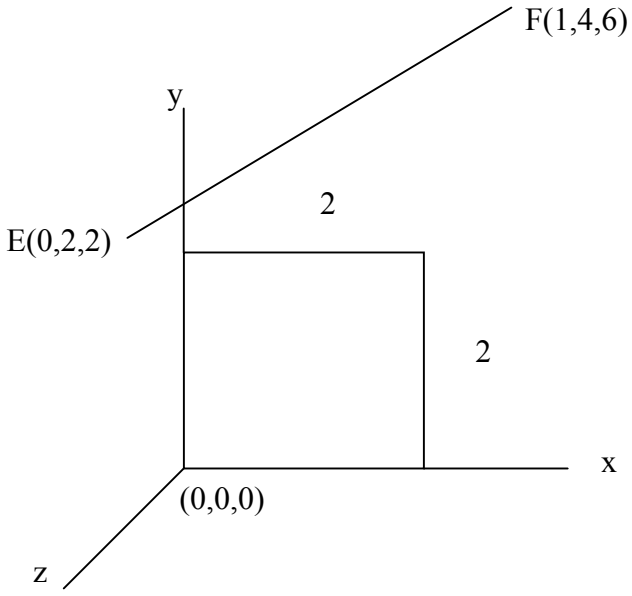
3.3 Rotation of an object about an Arbitrary Axis

A 3-D rotation of a geometric model about an arbitrary axis is complex, and involves several rotation and translation transformations. Following is a step by step procedure to accomplish the transformation.

1. Translate the given axis so that it will pass through the origin.
2. Rotate the axis about x-axis (or y-axis) so that it will lie in the xz-plane (angle α).
3. Rotate the axis about the y-axis so that it will coincide with the z-axis (angle φ).
4. Rotate the geometric object about the z-axis (angle θ).
5. Reverse of step 3.
6. Reverse of step 2.
7. Reverse of step 1.

We will illustrate this procedure by the following example.

Example 3: Rotate the rectangle shown, 30^0 ccw about the line EF and find the new coordinates of the rectangle.



Solution: We will make use of the seven-step procedure outlined above and write the applicable transformation matrix in each step. After we have generated all the transformation matrices, we will solve for the new coordinates of the rectangle at the end of the 7^{th} step.

1. Translate the given axis so that it will pass through the origin

Translation of the line EF to origin is given as,

$$[P^*]_1 = [P]\,[T_t], \text{ where } [P] = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \end{pmatrix} \text{ and } [T_t] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -2 & -2 & 1 \end{pmatrix}$$

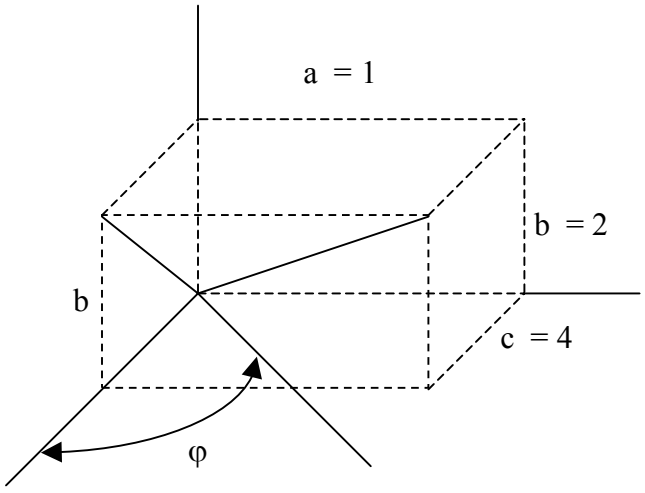
2. Rotate the axis so that it will lie in the yz-plane

The line EF is now rotated an angle α, about the x-axis so that it will lie in the xz-plane. The angle α is calculated with trigonometric relations, shown in the figure.

$$\cos\alpha = c/d = c/\sqrt{(b^2 + c^2)} = 4/(4.4721) = .8944$$

$$\sin\alpha = b/d = 2/(4.4721) = .4472$$

Now, $[P^*]_2 = [P][T_t][T_r]_\alpha$, where,



$$[T_r]_\alpha = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & .8944 & .4472 & 0 \\ 0 & -.4472 & .8944 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3. Rotate the line so that it will coincide with the z-axis

We will now rotate the line an angle φ about the y-axis so that it will coincide with the z-axis. The value of the angle φ is calculated from the trigonometry of the figure shown.

$\sin\varphi = a/L = 1/\sqrt{(a^2 + b^2 + c^2)} = 1/(4.5825) = 0.2182$

$\cos\varphi = d/L = (4.4721)/(4.5825) = 0.9759$

Now, the points matrix at this step is $[P^*]_3 = [P^*]_2[T_r]_\varphi$, and

$$[T_r]_\varphi = \begin{pmatrix} \cos\varphi & 0 & -\sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.9759 & 0 & -0.2182 & 0 \\ 0 & 1 & 0 & 0 \\ 0.2182 & 0 & 0.9759 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4. Rotate the Geometric Object about the z-axis

Up to this point we have translated and rotated the rectangle so that its original position is changed and the line is coincident with the z-axis. To understand the effect of these steps, imagine that the rectangle and the line are frozen in space in a box. Now, move (translate) the line to the origin so that the new coordinates of the point E are (0,0,0), rotate the box about the x-axis, so that the line EF lies in the xz-plane, finally, rotate the box about the y-axis so that it coincides with the z-axis. The noteworthy point in this analogy is that the transformation carried out in steps 1 through 3, affect both the coordinates of the line as well as that of the rectangle. Now we are ready to carry out the rotation of the rectangle about line EF. Since the axis of rotation is now coincident with the z-axis, we can apply the equation of rotation about the z-axis, defined earlier. Therefore,

$$[T_r]_\theta = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.866 & 0.5 & 0 & 0 \\ -0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5. Reverse of Step 3

In this step we will rotate the frozen box an angle -φ, about the y-axis. Since cos (-φ) = cosφ, and sin(-φ) = -sin(φ), the transformation matrix is,

$$[T_r]_{-\varphi} = \begin{pmatrix} \cos(-\varphi) & 0 & -\sin(-\varphi) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(-\varphi) & 0 & \cos(-\varphi) & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0.9759 & 0 & 0.2182 & 0 \\ 0 & 1 & 0 & 0 \\ -0.2182 & 0 & 0.9759 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

6. Reverse of Step 2

Rotate the box an angle -α about the x-axis. The transformation matrix is,

$$[T_r]_{-\alpha} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-\alpha) & \sin(-\alpha) & 0 \\ 0 & -\sin(-\alpha) & \cos(-\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.8944 & -0.4472 & 0 \\ 0 & 0.4472 & 0.8944 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

7. Reverse of Step 1

In this final step, we will translate the box so that the corner E will move back to its original coordinates (0,2,2). The transformation matrix is,

$$[T_{-t}] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 2 & 2 & 1 \end{pmatrix}$$

This completes all the seven steps that are necessary to rotate the rectangle about the line EF. The new coordinates of the rectangle are given by the equation,

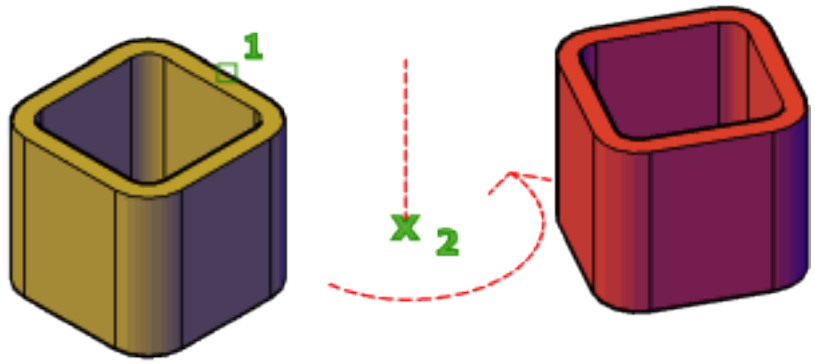
$$[P^*] = [P]\,[T_t]\,[T_r]_\alpha\,[T_r]_\varphi\,[T_r]_\theta\,[T_r]_{-\varphi}\,[T_r]_{-\alpha}\,[T_{-t}]$$

The concatenated transformation matrix is,

$$[T]_c = [T_t]\,[T_r]_\alpha\,[T_r]_\varphi\,[T_r]_\theta\,[T_r]_{-\varphi}\,[T_r]_{-\alpha}\,[T_{-t}] = \begin{pmatrix} 0.9312 & 0.1634 & -0.3256 & 0 \\ -0.1743 & 0.9846 & -0.0044 & 0 \\ 0.3199 & 0.0609 & 0.9454 & 0 \\ -0.2913 & -0.0909 & 0.1179 & 1 \end{pmatrix}$$

$$\text{and } [P^*] = [P][T]_c = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.9312 & 0.1634 & -0.3256 & 0 \\ -0.1743 & 0.9846 & -0.0044 & 0 \\ 0.3199 & 0.0609 & 0.9454 & 0 \\ -0.2913 & -0.0909 & 0.1179 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} -0.2913 & -0.0909 & 0.1179 & 1 \\ 1.5712 & 0.2359 & -0.5334 & 1 \\ 1.2226 & 2.2051 & -0.5421 & 1 \\ -0.6399 & 1.8783 & 0.1092 & 1 \end{pmatrix}$$



CHAPTER 4

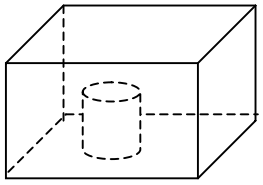
CURVES

4.1 Introduction

In order to understand the significance of curves, we should look into the types of model representations that are used in geometric modelling. Curves play a very significant role in CAD modelling, especially, for generating a wireframe model, which is the simplest form for representing a model.

We can display an object on a monitor screen in three different computer-model forms:
- Wireframe model
- Surface Model
- Solid model

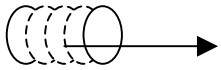
Wireframe model: A wireframe model consist of points and curves only, and looks as if its made up with a bunch of wires. This is the simplest CAD model of an object. Advantages of this type of model include ease of creation and low level hardware and software requirements. Additionally, the data storage requirement is low. The main disadvantage of a wireframe model is that it can be very confusing to visualize. For example, a blind hole in a box may look like a solid cylinder, as shown in the figure.



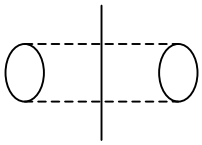
A wireframe model – Model of a Solid object with a blind hole

In spite of its ambiguity, a wireframe model is still the most preferred form, because it can be created quickly and easily to verify a concept of an object. The wireframe model creation is somewhat similar to drawing a sketch by hand to communicate or conceptualize an object. As stated earlier, a wireframe model is created using points and curves only.

Surface Model: sweeping a curve around or along an axis can create a surface model. The figures below show two instances of generating a surface model.



Generating a cylinder by sweeping a circle in the direction of an axis

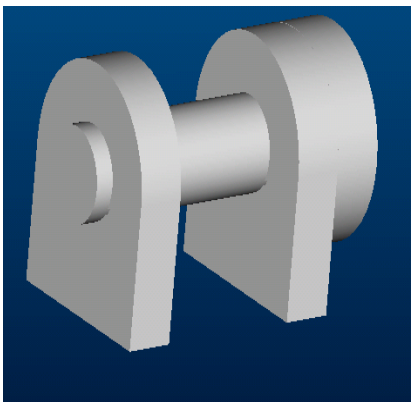


generating a donut by sweeping a circle around an axis

The appearance or resolution of a surface model depends on the number of sweeping instances we select. For a realistic looking model, we need to select a large number of instances, requiring a large computer memory, or, opt for a not-so realistic model by selecting a small number of instances, and save memory. In some commercial CAD packages we have the option of selecting the resolution of a model, other packages have a fixed value for resolution that cannot be changed by users.

Surface models are useful for representing surfaces such as a soft-drink bottle, automobile fender, aircraft wing, and in general, any complicated curved surface. One of the limitations of a surface model is that there is no geometric definition of points that lie inside or outside the surface.

Solid Model: Representation of an object by a solid model is relatively a new concept. There were only a couple of solids modelling CAD programs available in late 1980s, and they required mainframe computers to run on. However, in 1990s, due to the low cost and high speed, PCs have become the most popular solid modelling software platform, prompting almost all the CAD vendors to introduce their 3-D solid modelling software that will run on a PC.



Solid models represent objects in a very realistic and unambiguous form; however, they require a large amount of storage memory and high-end computer hardware. A solid model can be shaded and rendered in desired colors to give it a more realist appearance.

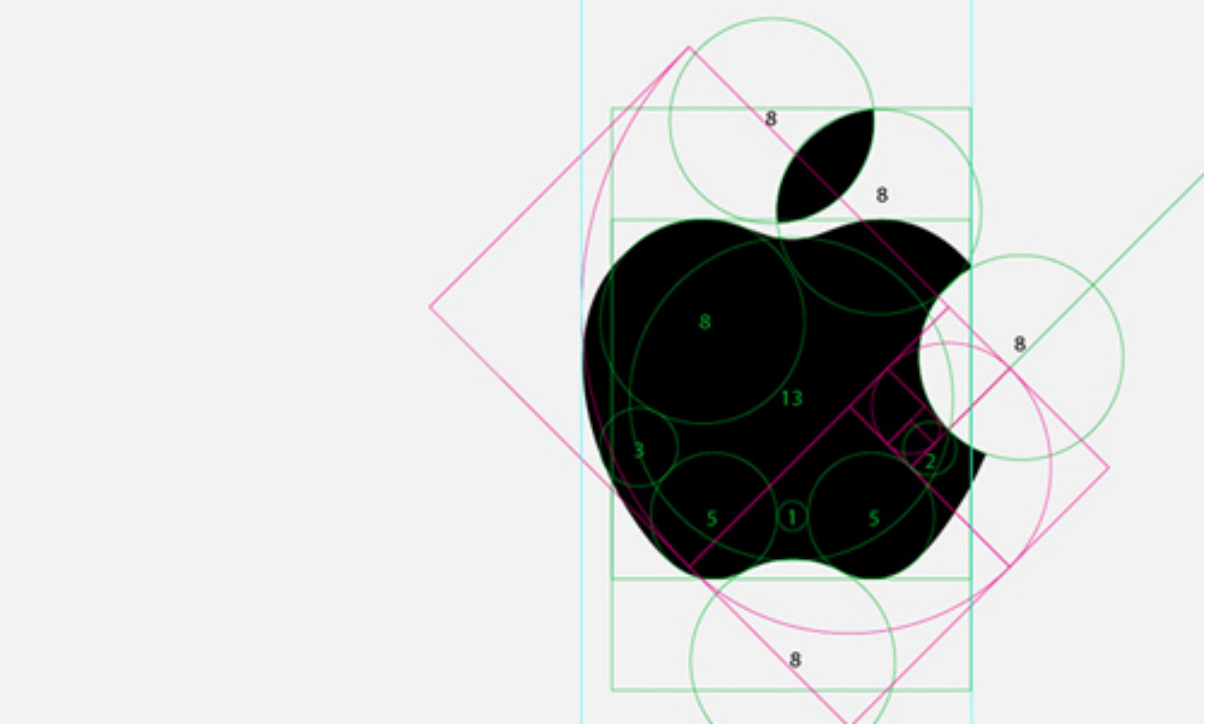
4.2 Role of Curves in Geometric modelling

Curves are used to draw a wireframe model, which consists of points and curves; the curves are utilized to generate surfaces by performing parametric transformations on them. A curve can be as simple as a line or as complex as a B-spline. In general, curves can be classified as follows:

- **Analytical Curves**: This type of curve can be represented by a simple mathematical equation, such as, a circle or an ellipse. They have a fixed form and cannot be modified to achieve a shape that violates the mathematical equations.
- **Interpolated curves**: An interpolated curve is drawn by interpolating the given data points and has a fixed form, dictated by the given data points. These curves have some limited flexibility in shape creation, dictated by the data points.
- **Approximated Curves**: These curves provide the most flexibility in drawing curves of very complex shapes. The model of a curved automobile fender can be easily created with the help of approximated curves and surfaces.

In general, sweeping a curve along or around an axis creates a surface, and the generated surface will be of the same type as the generating curve, e.g., a fixed form curve will generate a fixed form surface.

As stated earlier, curves are used to generate surfaces. To facilitate the computer-language algorithm, curves are represented by parametric equations. Non-parametric equations are used only to locate a point of intersection on the curve, and not for generating them. Let us briefly discus the parametric and non-parametric form of a curve.



4.3 Parametric and Non-parametric Equations of a Curve

The mathematical representation of a curve can be classified as either parametric or non-parametric (natural). A non-parametric equation has the form,

$y = c_1 + c_2 x + c_3 x^2 + c_4 x^3$ *Explicit non-parametric equation*

This is an example of an explicit non-parametric curve form. In this equation, there is a unique single value of the dependent variable for each value of the independent variable. The implicit non-parametric form of an equation is,

$(x - x_c)^2 + (y - y_c)^2 = r^2$ *Implicit non-parametric equation*

In this equation, no distinction is made between the dependent and the independent variables.

Parametric Equations: Parametric equations describe the dependent and independent variables in terms of a parameter. The equation can be converted to a non-parametric form, by eliminating the dependent and independent variables from the equation. Parametric equations allow great versatility in constructing space curves that are multi-valued and easily manipulated. Parametric curves can be defined in a constrained period ($0 \le t \le 1$); since curves are usually bounded in computer graphics, this characteristic is of considerable importance. Therefore, parametric form is the most common form of curve representation in geometric modelling. Examples of parametric and non-parametric equations follow.

	Non-Parametric	**Parametric**
***Circle*:**	$x^2 + y^2 = r^2$	$x = r\cos\theta,\ y = r\sin\theta$

Where, θ is the parameter.

CAD programs prefer a parametric equation for generating a curve. Parametric equations are converted into matrix equations – to facilitate a computer solution, and then varying a parameter from 0 to 1 creates the points or curves. In this course, we will use the following parameters, with the range indicated,

$0 \le t \le 1$ $0 \le s \le 1$ $0 \le \theta \le 2\pi s$ $0 \le \varphi \le 2\pi s$

4.4 Fixed-Form or Analytical Curves

4.4.1 Equation of a Straight Line: The simplest fixed-form curve is a straight line. Parametric equation of a straight line is given as,

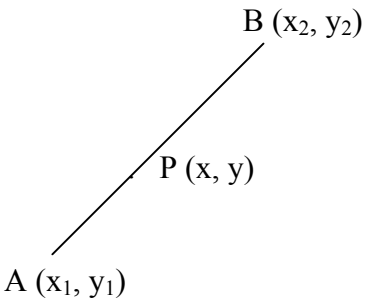
$$P(t) = A + (B-A)\, t \qquad (4.1)$$

The parametric equation of line AB can be derived as,

$$x = x_1 + (x_2 - x_1)\, t$$

$$y = y_1 + (y_2 - y_1)\, t$$

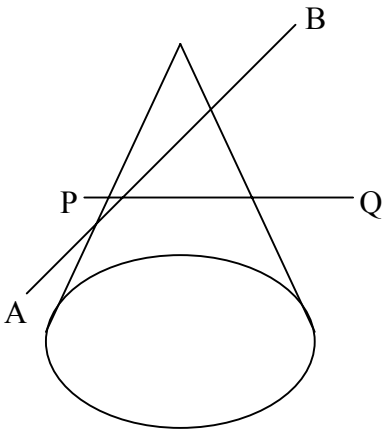
where, $0 \leq t \leq 1$



The point P on the line is sweeped from A to B,
as the value of t is varied from 0 to 1.

4.4.2 Conic Sections or Conic Curves

A conic curve is generated when a plane intersects a cone, as shown.



The intersection of the plane PQ and the cone is a circle, where as, the intersection created by the plane AB is an ellipse. Other curves that can be created are parabola and hyperbola.

Conic curves are used to create simple wireframe models of objects, which have edges that can be represented by these analytical curves. The fixed-form or analytical curves do not have inflection points, i.e., curves have slopes that are either positive or negative and do not change their sign (positive slope will remain positive and negative slope will remain negative). All conic curves can be represented by a quadratic equation, for example, circular and elliptical curves have quadratic polynomial equations.

4.4.3 Circular Curve

The non-parametric equation of a circle is,

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \qquad (4.2)$$

Where, x_c, and y_c are coordinates of the center, and r is radius of the circle.

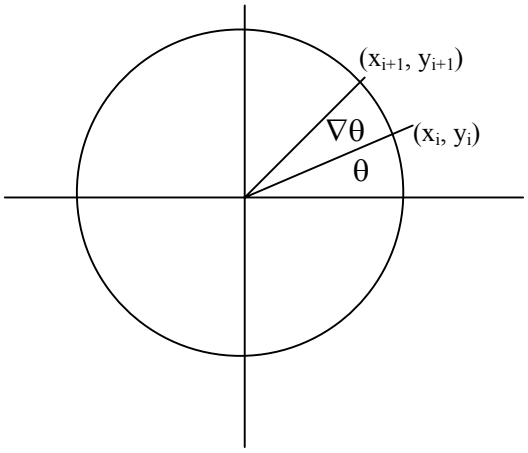
If we were to use this form of the equation for plotting a circle or a circular curve, we will first calculate several values of x and y along the circumference of the circle, and then plot them. The curve thus generated will be of a poor quality, unless we plot a very large number of data points, which will result in a significant demand for storage of these data points. Therefore, as stated earlier, in CAD programs, we use a parametric equation, which avoids the need for storage of the data points, and provides a smooth curve. The parametric equation of the above circle can be written as,

$$x_i = x_c + r\cos\theta$$
$$y_i = y_c + r\sin\theta \qquad (4.3)$$

This equation is converted into a matrix form so that a computer can solve it. We will now convert this equation into a matrix form.

Let us assume that the plot starts at the point (x_i, y_i), and the center lies at the origin. We increment θ to $(\theta + \Delta\theta)$, giving us the new point on the circle (x_{i+1}, y_{i+1}), or

$$x_{i+1} = r\cos(\theta + \Delta\theta)$$
$$y_{i+1} = r\sin(\theta + \Delta\theta)$$



Expanding it by the use of trigonometric identities, we get:

$$x_{i+1} = r\cos\theta\cos\Delta\theta - r\sin\theta\sin\Delta\theta$$

$$y_{i+1} = r\sin\theta\cos\Delta\theta + r\cos\theta\sin\Delta\theta$$

Substituting the values: $x_i = r\cos\theta$, and $y_i = r\sin\theta$, we get

$$x_{i+1} = x_i\cos\Delta\theta - y_i\sin\Delta\theta$$

$$y_{i+1} = y_i\cos\Delta\theta + x_i\sin\Delta\theta$$

In matrix form, these equations can be written as,

$$[x_{i+1}\ \ y_{i+1}\ \ 0\ \ 1] = [x_i\ \ y_i\ \ 0\ \ 1] \begin{pmatrix} \cos\Delta\theta & \sin\Delta\theta & 0 & 0 \\ -\sin\Delta\theta & \cos\Delta\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (4.4)$$

Equation (4.4) is valid for a circle that has center at the origin. To find the equation of a circle that has center located at an arbitrary point (x_c, y_c), we can use the translation transformation. Note that the equation (4.4) can be interpreted as rotational transformation of points x_i and y_i about the origin. Now, instead of rotation about the origin, we wish to rotate the point about the fixed point (x_c, y_c). This can be accomplished by the three-step approach, discussed in chapter 2, i.e., first translate the fixed point to the origin, rotate the object, and finally translate it so that the fixed point is restored to its original position. Using this procedure we will get:

$$[x_{i+1} \quad y_{i+1} \quad 0 \quad 1] =$$

$$[x_i \quad y_i \quad 0 \quad 1] \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_c & -y_c & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\Delta\theta & \sin\Delta\theta & 0 & 0 \\ -\sin\Delta\theta & \cos\Delta\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times$$

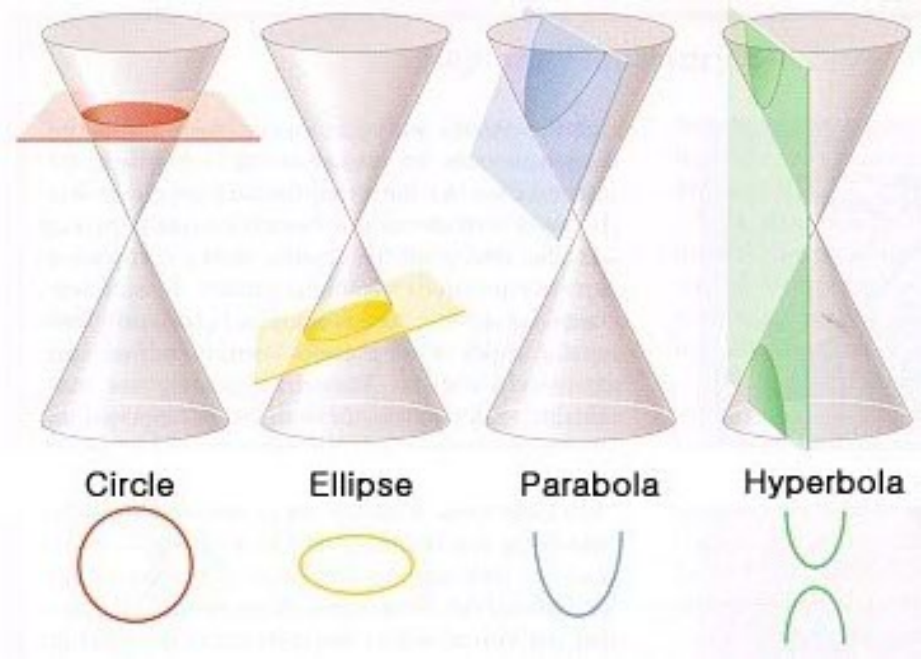
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_c & y_c & 0 & 1 \end{pmatrix} \qquad (4.5)$$

Simplifying the equation we get,

$$x_{i+1} = x_c + (x_i - x_c)\cos\Delta\theta - (y_i - y_c)\sin\Delta\theta$$

$$y_{i+1} = y_c + (x_i - x_c)\sin\Delta\theta + (y_i - y_c)\cos\Delta\theta \qquad (4.6)$$

Even though, equations (4.6) can be used as an iterative formula to plot a circle or a circular curve, using the EXCEL or MATLAB, or any other plot routines, the matrix equation (4.5) is the preferred form for a CAD program. The original CAD programs used iterative formulas to generate curves. The BASIC and FORTRAN languages were used to write the CAD codes.

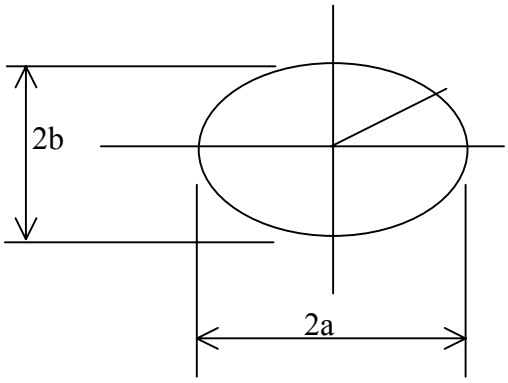


4.4.4 Ellipse

Following the procedure outlined in the previous section, we can derive the parametric equations of an ellipse. Parametric equation of an ellipse is given by
the equation

$x_i = a \cos\theta$

$y_i = b \sin\theta$

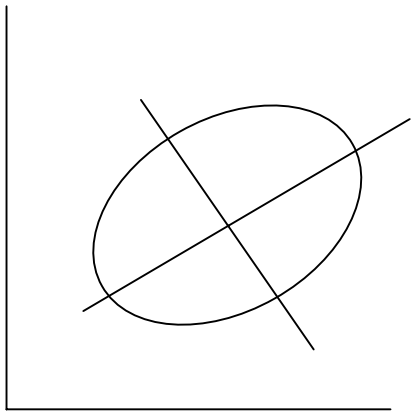


For a point on the ellipse, in general, the equation is

$$x_{i+1} = x_i \cos\Delta\theta - (a/b)\, y_i \sin\Delta\theta$$
$$y_{i+1} = y_i \cos\Delta\theta - (b/a)\, x_i \sin\Delta\theta \qquad (4.7)$$

For a more general case, when the axes of the ellipse are not parallel to the coordinate axes, and the center of the ellipse is at a distance x_c, y_c from the origin, the equation of the ellipse is given below. Let α be the angle that the major axis makes with the horizontal (x-axis), as shown. The equation of the ellipse can be derived as

$$x_i = x_c + x'_i \cos\alpha - y'_i \sin\alpha$$
$$y_i = y_c + x'_i \sin\alpha + y'_i \cos\alpha \qquad (4.8)$$



Where, x' and y' are the coordinate values of a Point on the ellipse, in term of the rotated axes x' and y'.

Equations (4.8) can be used to write either as an iterative formula or as a matrix equation for creating an elliptical curve.

4.5 Interpolated Curves

Interpolation method can be applied to draw curves that pass through a set of the given data points. The resulting curve can be a straight line, quadratic, cubic, or higher order curve. We are quite familiar, and have used, the linear interpolation of a straight line, given by the formula

$$f(x) = f(x_i) + [f(x_{i+1}) - f(x_i)]\ [(x-x_i) / (x_{i+1} - x_i)] \qquad (4.9)$$

Now, we will discuss the higher order curves, which are represented by higher order polynomials. Lagrange polynomial is a popular polynomial function used for interpolation of high order polynomials.

4.5.1 Lagrange Polynomial

When a sequence of planar points (x_0, y_0), (x_1, y_1), (x_2, y_2), ….(x_n, y_n) is given, the n^{th} degree of interpolated polynomial can be calculated by the Lagrange Polynomial equation,

$$f_n(x) = \Sigma\ y_i\ L_{i,n}(x) \qquad (4.10)$$

where,

$$L_{i,n}(x) = [(x - x_0) \ldots. (x - x_{i-1})\ (x - x_{i+1}) \ldots. (x - x_n)] / [(x_i - x_0) \ldots. (x_i - x_{i-1})\ (x_i - x_{i+1}) \ldots. (x_i - x_n)]$$

To understand the above expression better, note that

- The term $(x - x_i)$ is skipped in the numerator, and
- The denominator starts with the term $(x_i - x_0)$ and skips the term $(x_i - x_i)$, which will make the expression equal to infinity.

Example: Using the Lagrange polynomial, find the expression of the curve containing the points, $P_0(1, 1)$, $P_1(2, 2)$, $P_2(3, 1)$

Solution: Here, $n = 2$ and $x_0 = 1$, $y_0 = 1$, $x_1 = 2$, $y_1 = 2$, etc. The polynomial is of a second degree. Expanding the Lagrange equation, we get,

$$f_2(x) = y_0\ [(x - x_1)\ (x - x_2)] / [(x_0 - x_1)\ (x_0 - x_2)] + y_1\ [(x - x_0)\ (x - x_2)] / [(x_1 - x_0)\ (x_1 - x_2)] + y_2\ [(x - x_0)\ (x - x_1)] / [(x_2 - x_0)\ (x_2 - x_1)]$$

$= (1)\,[(x-2)(x-3)] / [(1-2)(1-3)] + (2)\,[(x-1)(x-3)] / [(2-1)(2-3)] + (1)\,[(x-1)(x-2)] / [(3-1)(3-2)]$

$= \tfrac{1}{2}(x^2 - 5x + 6) - 2(x^2 - 4x + 3) + \tfrac{1}{2}(x^2 - 3x + 2)$ or

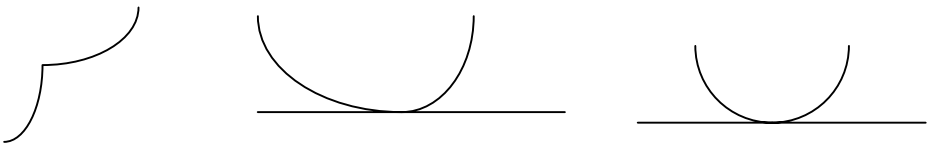
$f_2(x) = -x^2 + 4x - 2$

This is the explicit non-parametric equation of a circle; the given points lie on the circumference.

4.5.2 Parametric Cubic Curve or Cubic Spline – Synthetic Curves

The analytical and interpolated curves, discussed in the previous section (4.4) and (4.5) are insufficient to meet the requirements of mechanical parts that have complex curved shapes, such as, propeller blades, aircraft fuselage, automobile body, etc. These components contain non-analytical, synthetic curves. Design of curved boundaries and surfaces require curve representations that can be manipulated by changing data points, which will create bends and sharp turns in the shape of the curve. The curves are called synthetic curves, and the data points are called vertices or control points. If the curve passes through all the data points, it is called an interpolant (interpolated). Smoothness of the curve is the most important requirement of a synthetic curve.

Various continuity requirements at the data points can be specified to impose various degrees of smoothness of the curve. A complex curve may consist of several curve segments joined together. Smoothness of the resulting curve is assured by imposing one of the continuity requirements. A zero order continuity (C^0) assures a continuous curve, first order continuity (C^1) assures a continuous slope, and a second order continuity (C^2) assures a continuous curvature, as shown below.



C^0 Continuity – The curve is Continuous everywhere

C^1 Continuity- Slope Continuity at the common point

C^2 Continuity - Curvature continuity at the common point

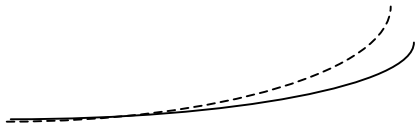
A cubic polynomial is the lowest degree polynomial that can guarantee a C^2 curve. Higher order polynomials are not used in CAD, because they tend to oscillate about the control points and require large data storage. Major CAD/CAM systems provide three types of synthetic curves: Hermite Cubic Spline, Bezier Curves, and B-Spline Curves.

Cubic Spline curves pass through all the data points and therefore they can be called as interpolated curves. Bezier and B-Spline curves do not pass through all the data points, instead, they pass through the vicinity of these data points. Both the cubic spline and Bezier curve have first-order continuity, where as, B-Spline curves have a second-order continuity.

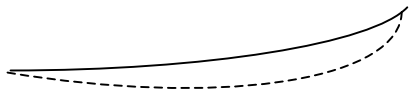
4.5.3 Hermite Cubic Spline

Hermite cubic curve is also known as parametric cubic curve, and cubic spline. This curve is used to interpolate given data points that result in a synthetic curve, but not a free form, unlike the Bezier and B-spline curves. The most commonly used cubic spline is a three-dimensional planar curve (not twisted). The curve is defined by two data points that lie at the beginning and at the end of the curve, along with the slopes at these points. It is represented by a cubic polynomial. When two end points and their slopes define a curve, the curve is called a Hermite cubic curve. Several cubic splines can be joined together by imposing the slope continuity at the common points. In design applications, cubic splines are not as popular as the Bezier and B-spline curves. There are two reasons for this:

- The curve cannot be modified locally, i.e., when a data point is moved, the entire curve is affected, resulting in a global control, as shown in the figure.
- The order of the curve is always constant (cubic), regardless of the number of data points. Increase in the number of data points increases shape flexibility, However, this requires more data points, creating more splines, that are joined together (only two data points and slopes are utilized for each spline).



Effect of Moving the Data Point



Effect of Change in slope

4.5.4 Equation of a Cubic Spline

A cubic spline is a third-degree polynomial, defined as

$$P(t) = \sum a_i t^i \qquad (4.11)$$

where, $0 \le t \le 1$, and P(t) is a point on the curve.
Expanding the above equation, we get

$$P(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0 \qquad (4.12)$$

If (x,y,z) are the coordinates of point P, the equation (4.12) can be written as,

$$\begin{aligned} x(t) &= a_{3x} t^3 + a_{2x} t^2 + a_{1x} t + a_{0x} \\ y(t) &= a_{3y} t^3 + a_{2y} t^2 + a_{1y} t + a_{0y} \\ z(t) &= a_{3z} t^3 + a_{2z} t^2 + a_{1z} t + a_{0z} \end{aligned} \qquad (4.13)$$

There are 12 unknown coefficients, $a_{ij,}$ known as the algebraic coefficients. These coefficients can be evaluated by applying the boundary conditions at the end points. From the coordinates of the end points of each segment, six of the twelve needed equations are obtained. The other six equations are found by using the tangent vectors at the two ends of each segment. Substituting the boundary conditions at t = 0, and t = 1, we get,

$P(0) = a_0$, and (a)

$P(1) = a_3 + a_2 + a_1 + a_0$ (b)

To find the tangent vectors, we differentiate equation (4.12), and get,

$$P'(t) = 3 a_3 t^2 + 2 a_2 t + a_1$$

Applying the boundary conditions at t = 0 and t = 1, we get,

$P'(0) = a_1$ (c)

$P'(1) = 3 a_3 + 2 a_2 + a_1$ (d)

Solving for the coefficients in terms of the P(t) and P'(t) values in equations (a) through (d), we get,

$$\begin{aligned} a_0 &= P(0) \\ a_1 &= P'(0) \\ a_2 &= -3\,P(0) + 3\,P(1) - 2\,P'(0) - P'(1) \\ a_3 &= 2\,P(0) - 2\,P(1) + P'(0) + P'(1) \end{aligned} \qquad (4.14)$$

The equation

$$P(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0$$

can be written, with coefficients a_{ij} replaced by the P(t) and P'(t) values in equations (4.14), resulting,

$$P(t) = [2 P(0) – 2 P(1) + P'(0) + P'(1)] t^3 + [-3 P(0) + 3 P(1) – 2 P'(0) – P'(1)] t^2 + P'(0) t + P(0)$$

Or, rearranging the terms, we get,

$$P(t) = [(2 t^3 – 3 t^2 + 1)] P(0) + [(-2 t^3 + 3 t^2)] P(1) + [(t^3 – 2 t^2 + t)] P'(0) + [(t^3 – t^2)] P'(1)$$

In matrix form the equation can be written as,

$$P(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P(0) \\ P(1) \\ P'(0) \\ P'(1) \end{pmatrix} \qquad (4.15)$$

The equation in short form can be written as: $P(t) = [t]\,[M]_H\,[G]$

Where, the terms [t], $[M]_H$, and [G] correspond to the terms on the right hand side of the equation (4.15). $[M]_H$ is called Hermite matrix of a cubic spline, and represents the constant matrix. The term [G] is called geometric coefficient matrix. Let us consider an example to understand how the equation (4.15) works.

Example 5: A parametric cubic curve passes through the points (0,0), (2,4), (4,3), (5, -2) which are parametrized at t = 0, ¼, ¾, and 1, respectively. Determine the geometric coefficient matrix and the slope of the curve when t = 0.5.

Solution: The points on the curve are

(0,0) at t = 0
(2,4) at t = ¼

(4,3) at t = ¾
(5,-2) at t = 1

Substituting in equation (4.15), we get,

$$\begin{pmatrix} 0 & 0 \\ 2 & 4 \\ 4 & 3 \\ 5 & -2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0.0156 & 0.0625 & 0.25 & 1 \\ 0.4218 & 0.5625 & 0.75 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P(0) \\ P(1) \\ P'(0) \\ P'(1) \end{pmatrix}$$

Solving, we get,

$$\begin{pmatrix} P(0) \\ P(1) \\ P'(0) \\ P'(1) \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 5 & -2 \\ 10.33 & 22 \\ 4.99 & -26 \end{pmatrix}$$

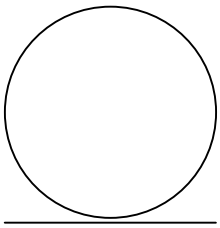
The slope at t = 0.5 is found by taking the first derivative of the equation (4.15), as follows,

$$P'(t) = [3t^3 \quad 2t \quad 1 \quad 0] \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 5 & -2 \\ 10.33 & 22 \\ 4.99 & -26 \end{pmatrix}$$

Therefore,

P'(0.5) = [3.67 -2.0], or

Slope = $\Delta x/\Delta y$ = -2.0/3.67 = - 0.545



Note: coinciding the end points, and imposing equal values of the slopes, as shown, can create closed shape of a cubic spline.

4.6 Approximated Synthetic Curves

In the previous sections, we have studied the analytical and interpolated curves, now we will focus on the approximated curves. Bezier and B-spline curves represent the approximated curves, these curves are synthetic, and can be joined together to form a very smooth curve. For data fitting, interpolated curves work best, whereas, for free form geometry, interpolation cannot be used, and approximation becomes necessary. In many engineering applications, smoothness of a curve is preferred over the quality of interpolation. These curves are flexible, local changes in the shape do not affect the entire shape of the curve. Let us study the Bezier curve first, followed by the B-spline.

4.6.1 Bezier Curves

Equation of the Bezier curve provides an approximate polynomial that passes near the given control points and through the first and last points. In 1960s, the French engineer P. Bezier, while working for the Renault automobile manufacturer, developed a system of curves that combine the features of both interpolating and approximating polynomials. In this curve, the control points influence the path of the curve and the first two and last two control points define lines which are tangent to the beginning and the end of the curve. Several curves can be combined and blended together. In engineering, only the quadratic, cubic and quartic curves are frequently used.

4.6.2 Bezier's Polynomial Equation

The curve is defined by the equation

$$P(t) = \sum V_i B_{i,n}(t) \qquad \text{where, } 0 \le t \le 1 \text{ and } i = 0, 1, 2, \ldots, n \qquad (4.16)$$

Here, V_i represents the n+1 control points, and $B_{i,n}(t)$ is the blending function for the Bezier representation and is given as

$$B_{i,n}(t) = \binom{n}{i} (t^i)(1-t)^{n-i} \qquad (4.17)$$

Where n is the degree of the polynomial and

$$\binom{n}{i} = \frac{n!}{i!\,(n-i)!} \qquad i = 0, 1, 2, \ldots\ldots, n$$

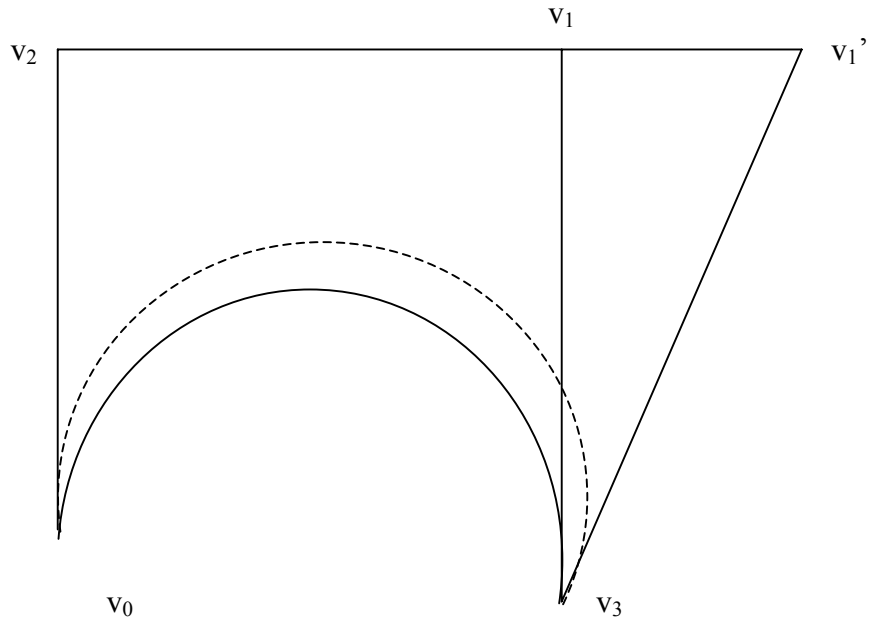
These blending functions satisfy the following equations

$$B_{i,n}(t) > 0 \quad \text{for all } i$$

$$\sum B_{i,n}(t) = 1 \qquad (4.18)$$

The equations (4.18) force the curve to lie entirely within the convex figure (or envelop) set by the extreme points of the polygon formed by the control points. The envelope represents the figure created by stretching a rubber band around all the control points.

The figure below shows that the first two points and the last two points form lines that are tangent to the curve. Also, as we move point v_1, the curve changes shape, such that the tangent lines always remain tangent to the curve.



Relationship between end-points and curve slope

Bezier's blending function produces an nth degree polynomial for n+1 control points and forces the Bezier curve to interpolate the first and last control points. The intermediate control points pull the curve toward them, and can be used to adjust the curve to the desired shape.

4.6.3 Third Order Bezier Polynomial

We will simplify the Bezier's equation for n = 3 (a cubic curve). The procedure developed here can be extended to the other values of n.

For n = 3, we will have four control points, namely, V_0, V_1, V_2, V_3. i will vary from 0 to 3. The Bezier's equation,

$P(t) = \sum V_i B_{i,3}(t)$ can be expanded to give, (4.19)

$P(t) = V_0 B_{0,3} + V_1 B_{1,3} + V_2 B_{2,3} + V_3 B_{3,3}$ and

$$B_{0,3} = \frac{3!}{0!\,3!} t^0 (1-t)^3 = (1-t)^3$$

$$B_{1,3} = \frac{3!}{1!\,2!} t^1 (1-t)^2 = 3t(1-t)^2$$

$$B_{2,3} = \frac{3!}{2!\,1!} t^2 (1-t)^1 = 3t^2(1-t)$$

$$B_{3,3} = \frac{3!}{3!\,0!} t^3 (1-t)^0 = t^3$$

By substituting the above values in the equation (4.19) we get

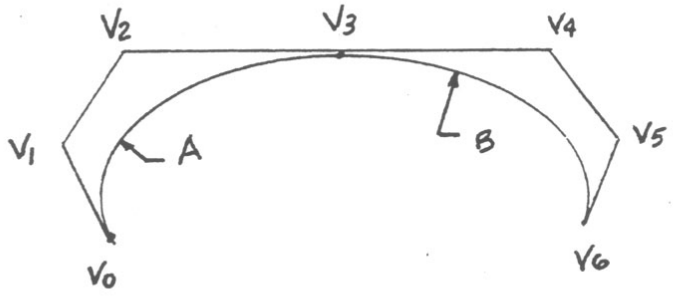
$$P(t) = (1-t)^3 V_0 + 3t(1-t)^2 V_1 + 3t^2(1-t) V_2 + t^3 V_3$$

In matrix form this equation is written as

$$P(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} \quad (4.20)$$

4.6.4 Blending Two or More Bezier Curves

Two or more Bezier curves can be blended to provide a desired curve of a complex nature. When joining curves, slope continuity is maintained by having three collinear points, the middle one being common to the adjoining curves, as shown.



Point V_3 is the middle point of the common points V_2,, V_3,, and V_4 of curves A and B.

NOTE: Using the Bezier curves, we can create closed curves by making the first and last points of the control points coincide.

Example: A cubic Bezier curve is described by the four control points: (0,0), (2,1), (5,2), (6,1). Find the tangent to the curve at t = 0.5.

Solution: We will use the Bezier cubic polynomial, given in equation (4.20), which is,

$$P(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix}$$

where, $V_0 = (0,0)$
$V_1 = (2,1)$
$V_2 = (5,2)$
$V_3 = (6,1)$

The tangent is given by the derivative of the general equation above,

$$P'(t) = [3t^2 \quad 2t \quad 1 \quad 0] \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} \qquad (4.21)$$

At t = 0.5, we get,

$$P'(t) = [3(.5)^2 \quad 2(.5) \quad 1 \quad 0] \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix}$$

$$= [6.75 \quad 1.5 \quad 0 \quad 1]$$

4.6.5 B-Spline Curve

B-spline curves use a blending function, which generates a smooth, single parametric polynomial curve through any number of points. To generate a Bezier curve of the same quality of smoothness, we will have to use several pieces of Bezier curves. Unlike the Bezier curve, the degree of the polynomial can be selected independently of the number of control points. The degree of the blending function controls the degree of the resulting B-spline curve. The curve has good local control, i.e., if one vertex is moved, only some curve segments are affected, and the rest of the curve remains unchanged.

The mathematical derivation of the B-spline curve is complex and beyond the scope of this course. The equation is of the form:

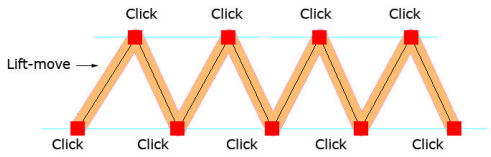
$$P(t) = \Sigma N_{i,k}(t) V_i \qquad (4.22)$$

Where, P(t) is a point on the curve.

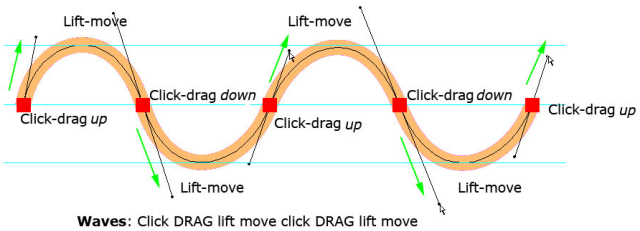
i indicates the position of control point i
k is order of curve
$N_{i,k}(t)$ are blending functions
V_i are control points

The matrix form of the uniform cubic B-spline curve is:

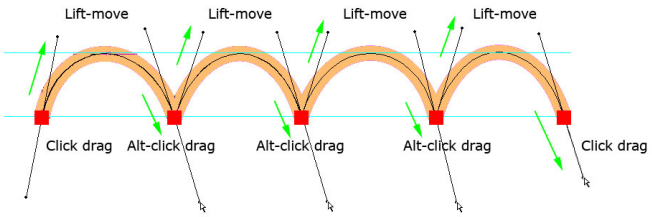
$$P_i(t) = 1/6[t^3 \quad t^2 \quad t \quad 1] \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix} \begin{pmatrix} V_{i-1} \\ V_i \\ V_{i+1} \\ V_{i+2} \end{pmatrix} \qquad (4.23)$$



Mountain Peaks: Left-click lift move, left-click lift move



Waves: Click DRAG lift move click DRAG lift move



Bumps: Click Drag lift move ALT click drag lift move ALT click drag

CHAPTER 5

SURFACES

5.1 Introduction

Wire frame models are unable to represent complex surfaces of objects like car, ship, airplane wing, castings etc. A surface model can be used to represent the surface profile of these objects. Also, surface model can be used for calculating mass properties, interference between parts, generating cross-sectioned views, generating finite element mesh, and generating NC tool paths for continuous path machining. Additionally, surface model can be used to fit experimental data, discretized solutions of differential equations, construction of pressure surface, construction of stress distribution etc.

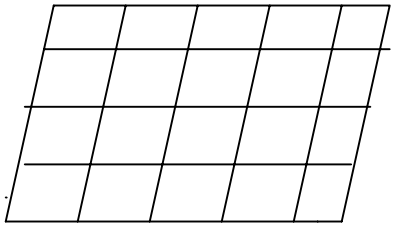
Surface creation on a CAD system usually requires wire frame entities: lines, curves, points, etc. All analytical and synthetic curves can be used to generate surfaces.
In order to visualize surfaces on a graphic display, a mesh, say m x n in size is usually displayed; the mesh size is controlled by the user. Most CAD systems provide options to set the mesh size.

A surface of an object is more complete and less ambiguous representation than its wire frame model; it is an extension of a wire frame model with additional information.
A wire frame model can be extracted from a surface model by deleting all surface entities (not the wireframe entities – point, lines, or curves!). Databases of surface models are centralized and associative, manipulation of surface entities in one view is automatically reflected in the other views. Surface models can be shaded and represented with hidden lines.

5.2 Types of Surfaces

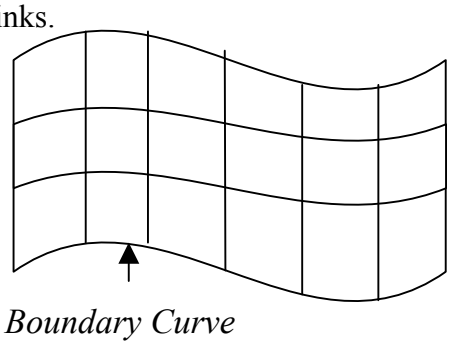
5.2.1 Plane Surface

This is the simplest surface, requires 3 non-coincidental points to define an infinite plane. The plane surface can be used to generate cross sectional views by intersecting a surface or solid model with it.



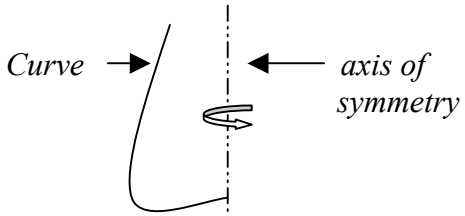
5.2.2 Ruled (lofted) Surface

This is a linear surface. It interpolates linearly between two boundary curves that define the surface. Boundary curves can be any wire frame entity. The surface is ideal to represent surfaces that do not have any twists or kinks.



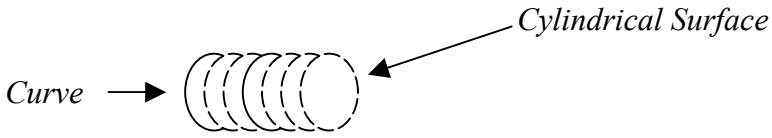
5.2.3 Surface of Revolution

This is an axisymmetric surface that can model axisymmetric objects. It is generated by rotating a planar wire frame entity in space about the axis of symmetry of a given angle.



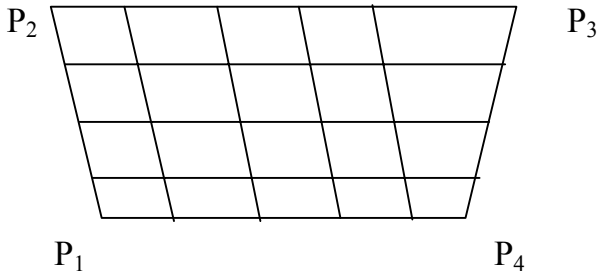
5.2.4 Tabulated Surface

This is a surface generated by translating a planar curve a given distance along a specified direction. The plane of the curve is perpendicular to the axis of the generated cylinder.



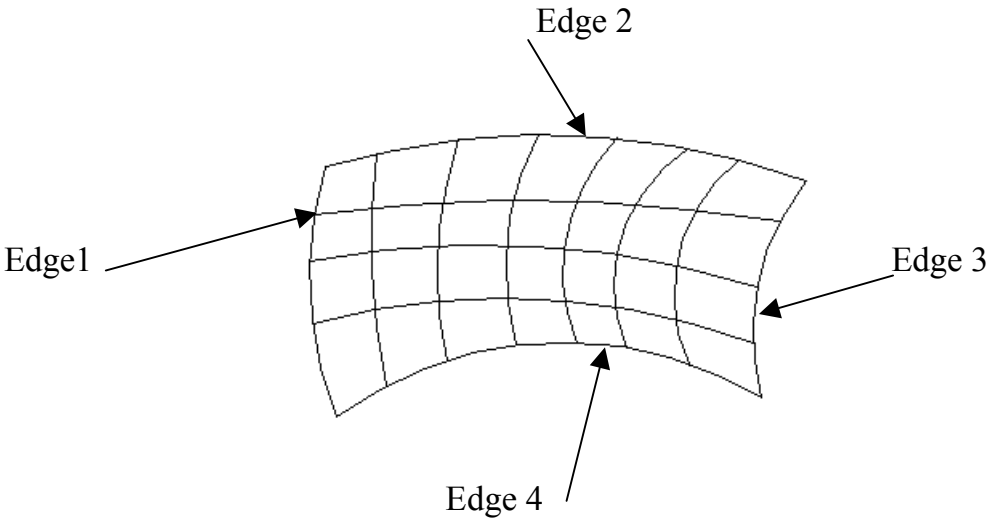
5.2.5 Bi-linear Surface

This 3-D surface is generated by interpolation of 4 endpoints. Bi-linear surfaces are very useful in finite element analysis. A mechanical structure is discretized into elements, which are generated by interpolating 4 node points to form a 2-D solid element.



5.2.6 Coons Patch

Coons patch or surface is generated by the interpolation of 4 edge curves as shown.



5.2.7 Bezier Surface

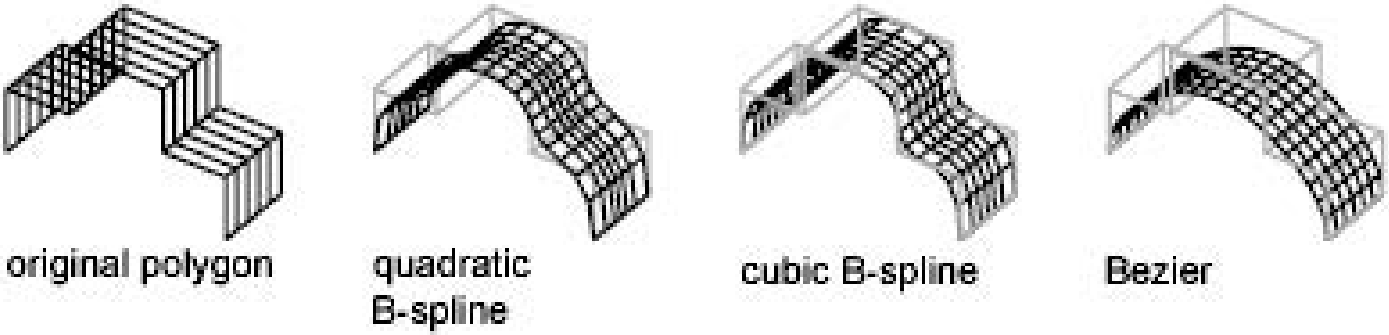
This is a synthetic surface similar to the Bezier curve and is obtained by transformation of a Bezier curve. It permits twists and kinks in the surface. The surface does not pass through all the data points.

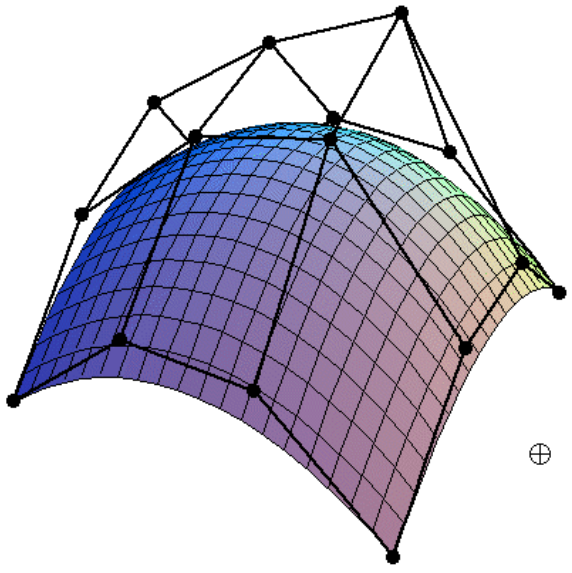
5.2.8 B-Spline Surface

This is a synthetic surface and does not pass through all data points. The surface is capable of giving very smooth contours, and can be reshaped with local controls.

Mathematical derivation of the B-spline surface is beyond the scope of this course. Only limited mathematical consideration will be given here.

Computer generated surfaces play a very important part in manufacturing of engineering products. A surface generated by a CAD program provides a very accurate and smooth surface, which can be generated by NC machines without any room for misinterpretation. Therefore, in manufacturing, computer generated surfaces are preferred. Since surfaces are mathematical models, we can quickly find the centroid, surface area, etc. Another advantage of CAD surfaces is that they can be easily modified.





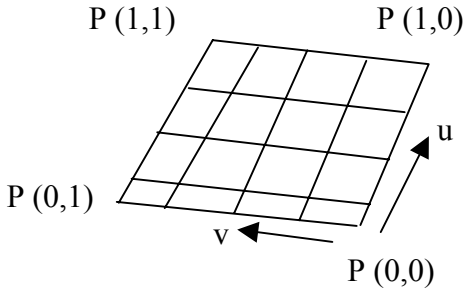
Bezier Surface

5.3 Interpolated Surfaces – Bilinear Surface

A bilinear surface is obtained by linear interpolation between four points, which may or may not lie in the same plane. The four points appear as vertices or corner points and the parameter values u and v create lines at various intervals to provide the surface visibility, shown in the figure. The parameters u and v are defined as

$0 \le u \le 1$, and $0 \le v \le 1$

Bilinear Patch



The interpolated parametric equation of a bilinear surface is given as:

$$P(u,v) = (1-u)(1-v)\,P(0,0) + u(1-v)\,P(1,0) + (1-u)\,P(0,1) + u\,v\,P(1,1)$$

In matrix form, it can be written as

$$P(u,v) = [(1-u)(1-v) \;\; u(1-v) \;\; (1-u)v \;\; uv] \begin{pmatrix} P(0,0) \\ P(1,0) \\ P(0,1) \\ P(1,1) \end{pmatrix}$$

Node points in FEA

Application of Bilinear Surfaces
Bilinear patches are extensively used in 2-D finite element analysis (FEA). In FEA, an engineering structure is defined by several bilinear surfaces (elements), which are created by joining points on the structure's geometry, called nodes. The nodes are connected to other nodes to create quadrilateral surfaces. Points not lying on the nodes are calculated by interpolation. Thus, the entire structure is completely defined by the nodes and the bilinear surfaces.

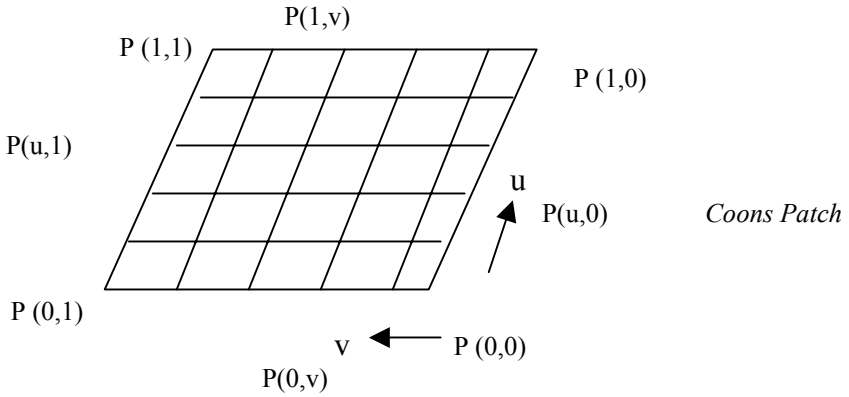
Drawbacks of Bilinear Surfaces
Bilinear surfaces have a very limited use, mainly, for FEA. Since only 4 points can be used in the interpolation, the smoothness of the generated surface is limited. Additionally, there is no flexibility to control shapes of the surface, unlike the sweeped surfaces.

5.4 Interpolated Surfaces – Coons Patch

A linear interpolation between four bounded curves is used to generate a Coons surface, also called as Coons patch. The method is credited to S. Coons who developed this concept for generating a surface.

Linear interpolation between the boundary curves P(0,v), P(u,0), P(1,v) , and P(u,1) gives the equation

$$Q(u,v) = (1-v)\,P(u,0) + u\,P(1,v) + v\,P(u,1) + (1-u)\,P(0,v)$$



The above equation gives wrong values at the corners (u,v = 0 and 1). For example, substituting the values of u and v we get,

$$Q(0,0) = P(0,0) + P(0,0) = 2P(0,0)$$
$$Q(1,0) = 2P(1,0), \text{ etc.}$$

Which are obviously wrong values. Therefore, The coons patch is created by modification of the interpolation equation, where the corners are subtracted. The modified interpolation equation is given as,

$$P(u,v) = (1-v)\,P(u,0) + u\,P(1,v) - v\,P(u,1) + (1-u)\,P(0,v) - (1-u)\,(1-v)\,P(0,0) - u\,(1-v)\,x\,P(1,0) - (1-u)\,v\,P(0,1) - u\,v\,P(1,1).$$

For computational purposes, it is more convenient to write this equation as,

$$P(u,v) = [(1-v) \quad u \quad v \quad (1-u)] \begin{pmatrix} P(u,0) \\ P(1,v) \\ P(u,1) \\ P(0,v) \end{pmatrix}$$ Eqns. of the boundary curves

$$- [(1-u)(1-v) \quad u(1-v) \quad (1-u)v \quad uv] \begin{pmatrix} P(0,0) \\ P(1,0) \\ P(0,1) \\ P(1,1) \end{pmatrix}$$ End-points (coordinates)

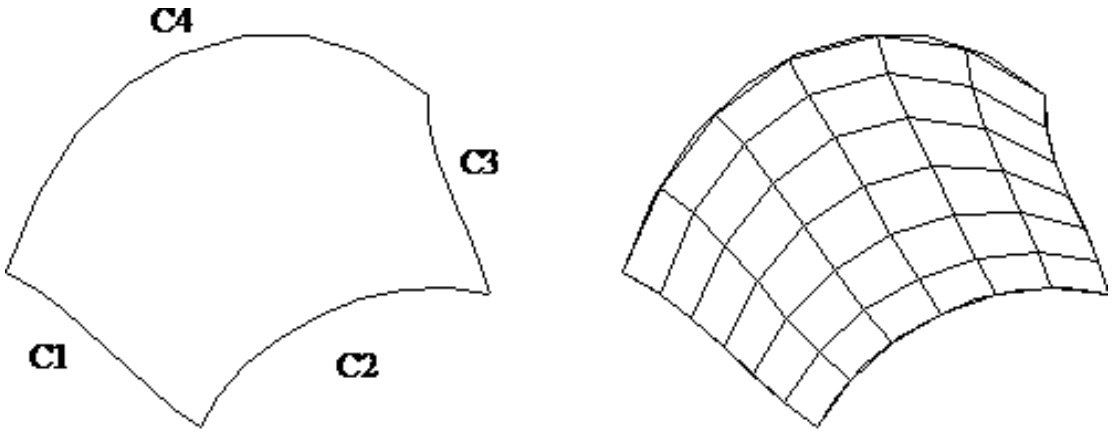
Which gives,

$$Q(u,v) = [(1-u) \quad u \quad 1] \begin{pmatrix} -P(0,0) & -P(0,1) & P(0,v) \\ -P(1,0) & -P(1,1) & P(1,v) \\ P(u,0) & P(u,1) & 0 \end{pmatrix} \begin{pmatrix} (1-v) \\ v \\ 1 \end{pmatrix}$$

Other interpolated surfaces include the Parametric Cubic patches.

Applications

Coons surface is easy to create, and therefore, many 2-D CAD packages utilize it for generating models. However, it has only a limited application since the surface is inflexible and cannot create very smooth surfaces. It would be very difficult to produce a smooth automobile fender using the Coons surface. Several CAD software, including AutoCAD, use this surface for generating surfaces between 4-bounded edges.

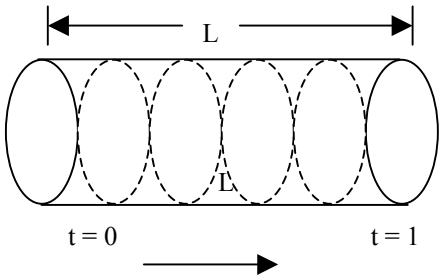


(a) Closed boundary (b) Coons patch

5.5 Linearly Swept Surfaces

A swept surface is generated when a curve is parametrically translated or rotated. In CAD, a surface is represented by a series of curves, which are parametrically generated at various instances. For example, a cylindrical surface is generated when a circular arc is translated up to the given dimension using a parameter t, where t varies as, $0 \leq t \leq 1$.

In the figure shown, the cylindrical surface is generated when a circular arc is translated a distance L, with the interim instances at t = 0.1, 0.2, 0.3, … 1. Here, the parameter t is given 10 values, and therefore, the surface of the cylinder is represented by 10 circular curves. The appearance of the surface improves as the parameter t varies at smaller intervals. Thus, if t is varied with $\Delta t = 0.01$, there will be 100 circular curves representing the surface.



A surface is an extension of a curve. The parametric representation of a curve is given by a single-vector equation of the form:

$$P(t) = [x(t) \quad y(t) \quad z(t)]$$

Here, only one parametric variable or one degree of freedom is needed. Whereas, a surface representation requires two parametric variables, and the equation is given as:

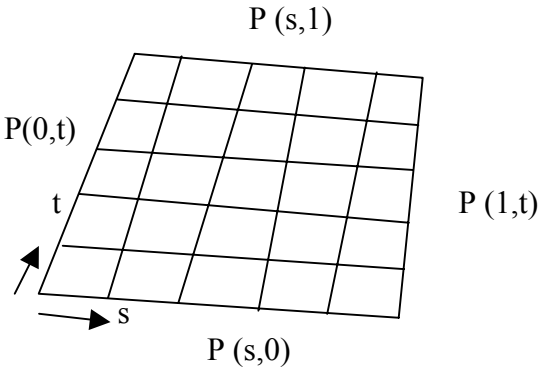
$$Q(s,t) = [x(s,t) \quad y(s,t) \quad z(s,t)]$$

Tracing a point in the s and t directions, as shown in the figure on the next page, generates a surface. One parameter variable is kept constant while varying the other one. A series of curves is created along the s and t directions. For example, constraining the parameters s and t between zero and 1, the set of curves generated along the s direction is,

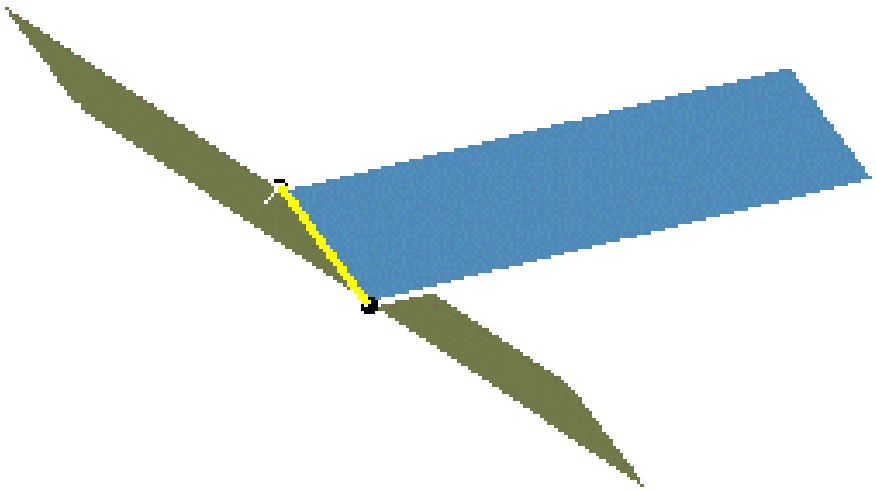
$$P(0,t), P(0.1,t), P(0.2, t) \ldots\ldots\ldots\ldots P(1, t)$$

and the other set of curves along the t direction is,

P(s,0), P(s,0.1),…P(S, 0.9), P(s,1).

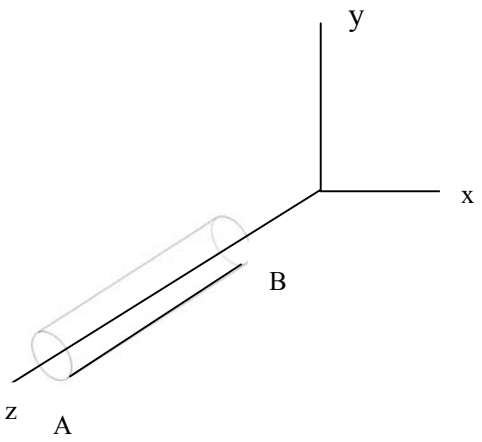


Thus, creation of a surface requires creation of the multiple curves that constitute it. This concept can be applied to both, the surface that has an analytical formulation (conic sections) and to a free-form surface (Bezier, B-spline).



5.6 Revolved Surfaces (Circular Sweep)

Surface of revolution is obtained by rotating a plane-curve around an axis. In the figure shown, line AB is rotated about the z-axis through an

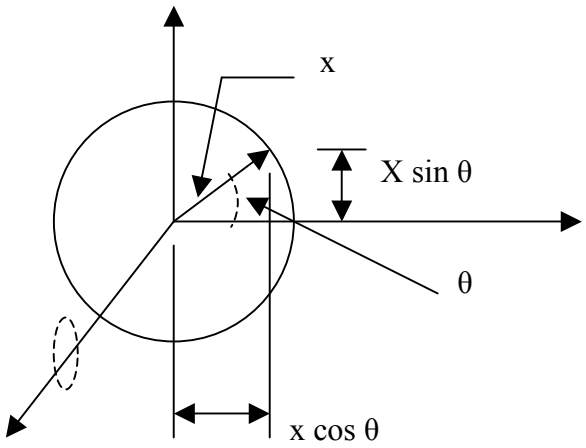


angle of 2π radians, generating a cylinder. A line or curve when revolved can generate all kinds of surfaces, based on the condition of rotation. Any point on the surface is a function of two parameters t and θ. Here, t describes the entity to be rotated and θ represents the angle of rotation. In general, a point on line AB (lying in the xz-plane) is represented by [x(t), 0, z(t)] and, when rotated by θ radians, it becomes [x(t)cosθ, x(t)sinθ, z(t)].

In general, the point matrix gives a point on the surface of revolution obtained by rotation around the z-axis,

$P(t, \theta) = [x(t) \cos\theta \quad x(t) \sin\theta \quad z(t)]$

In matrix form the equation can be written as,



$$P(t,\theta) = [x(t) \quad 0 \quad z(t) \quad 1] \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Note: The above rotation matrix is equivalent to the rotational transformation matrix studied earlier, which is,

$$\begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Thus, the generated surface is a rotational transformation of a line (or curve), except θ is not constant, but has values, $0 \le \theta \le 2\pi$.

Example

Generate the conical surface obtained by rotation of the line segment AB around the z-axis with,

A = (1,0,1) and B = (7,0,7).

Solution

Line AB can be represented in parametric form as:

P (t) = [x(t) y(t) z(t)], and the parametric equation of a line is,

P (t) = A + (B-A) t

based on this equation, the coordinates of a of point on the line are given as,

x(t) = 1 + (7-1) t = 1+ 6t,
y(t) = 0

$z(t) = 1 + (7-1)\, t = 1 + 6t$

The equation of the surface as given above is,

$P(t, \theta) = [x(t) \cos\theta \quad x(t)\sin\theta \quad z(t)]$ or

$= [(1+6t) \cos\theta \quad (1+6t) \sin\theta \quad (1+6t)]$ - *equation of the surface*

Any point on the surface can be located by substituting t and θ values in the above equation, e.g.: at $t = 0.4$ and $\theta = \pi/2$ radians

$P(0.4, \pi/2) = [1+6(.4)\cos(\pi/2) \quad 1+ 6(.4) \sin(\pi/2) \quad 1 + 6(.4)]$

$= [0 \quad 3.4 \quad 3.4]$, which is the point on the surface at $(.4, \pi/2)$

Example

Generate a Torus by rotating a circle of radius r and the center at (a,0,0) about the z-axis.

Solution

Rotating a circle contained in the x z plane around the z-axis can generate a torus. The center of the circle has coordinates (a,0,0) and equation of the circle in parametric form is given as;

$P(\phi) = [(a + r\cos\phi, 0, r\sin\phi]$

The torus is represented by,

$Q(\phi, \theta) = \{[(a + r\cos\phi)\cos\theta], [(a + r\cos\phi)\sin\theta], r\sin\phi\}$ – *equation of the torus*

In this case, the parameters are φ and θ.

5.7 Circular Sweep of a Synthetic Curve

Equation of a synthetic curve (free-form curve), as derived earlier, is given as,

$P(t) = [t][M][V]$

The surface of revolution is then given by,

$Q(t, \theta) = [t][M][V][Tr]^{\theta} = [Q(t)][Tr]^{\theta}$

Where, $Q(t, \theta)$ is the equation of the curve, and $[Tr]^{\theta}$ is the rotation matrix about the z-axis.

Note: To rotate the curve about the axis, we will have to use the translation and rotation matrices.

Example:

A cubic Bezier curve is defined by the control points: P_1 (1,0,2), P_2 (3,0,4), P_3 (2,0,6), P_4 (5,0,7). Find the surface of revolution obtained by revolving the curve about the z-axis and calculate the point on the surface at $t = 0.5$, $\theta = \pi/4$ rad.

Solution

The cubic Bezier curve is given by the equation,

$$P(t) = [t][m][v]$$

$$= [t^3 \;\; t^2 \;\; t \;\; 1] \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

Substituting the coordinates of the points, we get

$$P(t) = [t^3 \;\; t^2 \;\; t \;\; 1] \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 2 & 1 \\ 3 & 0 & 4 & 1 \\ 2 & 0 & 6 & 1 \\ 5 & 0 & 7 & 1 \end{pmatrix}$$

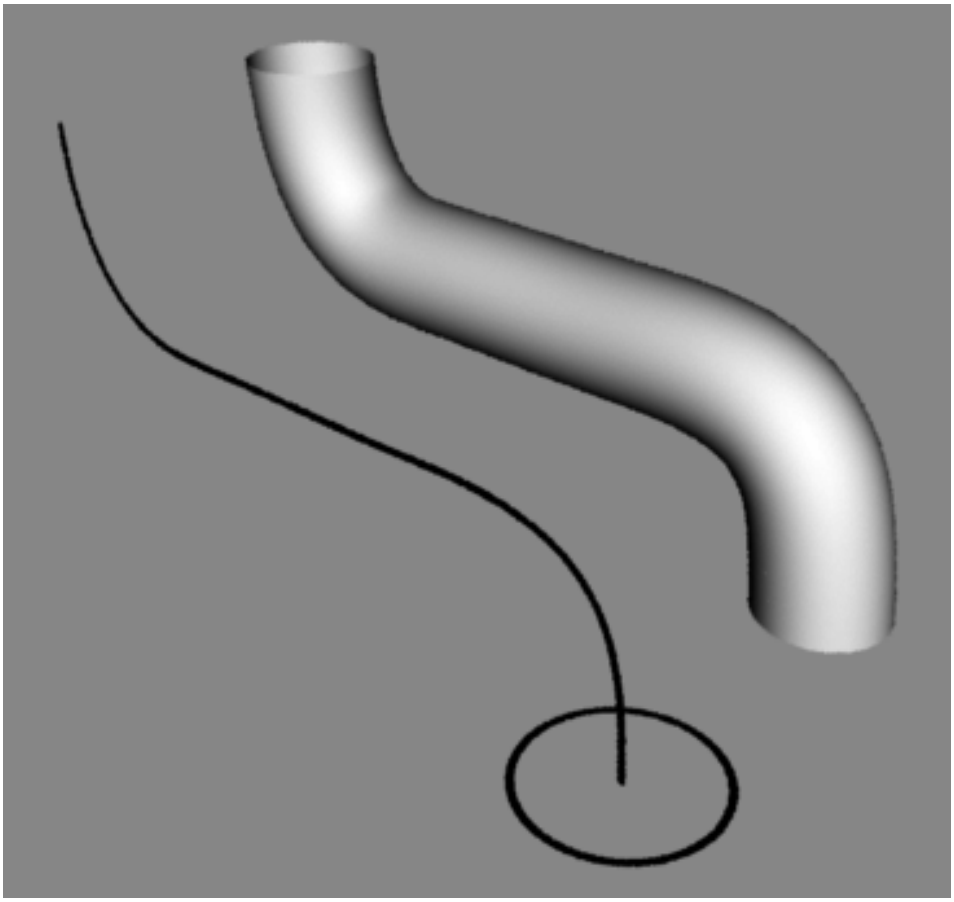
The surface of revolution is:

$$Q(t,\theta) = [t^3 \quad t^2 \quad t \quad 1] \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 2 & 1 \\ 3 & 0 & 4 & 1 \\ 2 & 0 & 6 & 1 \\ 5 & 0 & 7 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad 0 \le \theta \le 2\pi n \qquad 0 \le n \le 1$$

For $t = 0.5$ and $\theta = \pi/4$, the surface equation is,

$$Q(t,\theta) = [(.5)^3 \quad (.5)^2 \quad (.5) \quad 1] \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 2 & 1 \\ 3 & 0 & 4 & 1 \\ 2 & 0 & 6 & 1 \\ 5 & 0 & 7 & 1 \end{pmatrix} \begin{pmatrix} \cos(\pi/4) & \sin(\pi/4) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= [1.86 \qquad 1.86 \quad 4.86 \quad 1]$$



5.8 Creating a Surface by Parametric Sweeping

In the examples given above, sweeping a curve parametrically generated the surfaces. In parametric sweeping procedure, a surface is generated through the movement of a line or a curve along or around a defined path. The curve is sweeped as the sweep parameter is varied from the values of 0 to 1, creating several instances of the curve along the sweep path. In general, the equation of the surface can be given as,

$$Q(t, s) = P(t)\, T(s)$$

Where, P (t) is the parametric equation of a curve and T(s) is the sweep transformation based on the shape of the path. The sweep transformation can consist of translation, scaling, rotation or a combined transformation. If the path is a straight line, the points along the path on the line can be represented by,

$$x(s) = as$$
$$y(s) = bs$$
$$z(s) = cs$$

and T (s) is given as,

$$T(s) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ as & bs & cs & 1 \end{pmatrix}$$

Where, a, b, c are coordinate values, and $0 \leq s \leq 1$

This is equivalent to a three-dimensional translation of a curve with several traces generated along the path, controlled by how the parameter s is varied.

Example
Consider the Bezier curve defined by the control points P1 = (0,5,0), P2 = (3,4,0), P3 = (2,0,0), and P4 = (5,0,0). Translate the curve five units along the z-axis to generate a swept surface.

Solution
$Q(t,s) = [P(t)]\,[T_t]$, substituting the numbers, we get,

$$Q(t, s) = [t^3 \quad t^2 \quad t \quad 1] \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 5 & 0 & 1 \\ 3 & 4 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 5 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 5s & 1 \end{pmatrix}$$

Substituting the value of s and solving the matrices can calculate any point on the surface.

5.9 Creating a Surface by Sweeping a polygon

Any polygon can be swept around a given path to generate a surface. The equation of the surface is given as,

$$Q(s, t) = [P]\{T(s)]$$

Where, [P] is the point matrix, and T(s) is the transformation matrix.

Example:

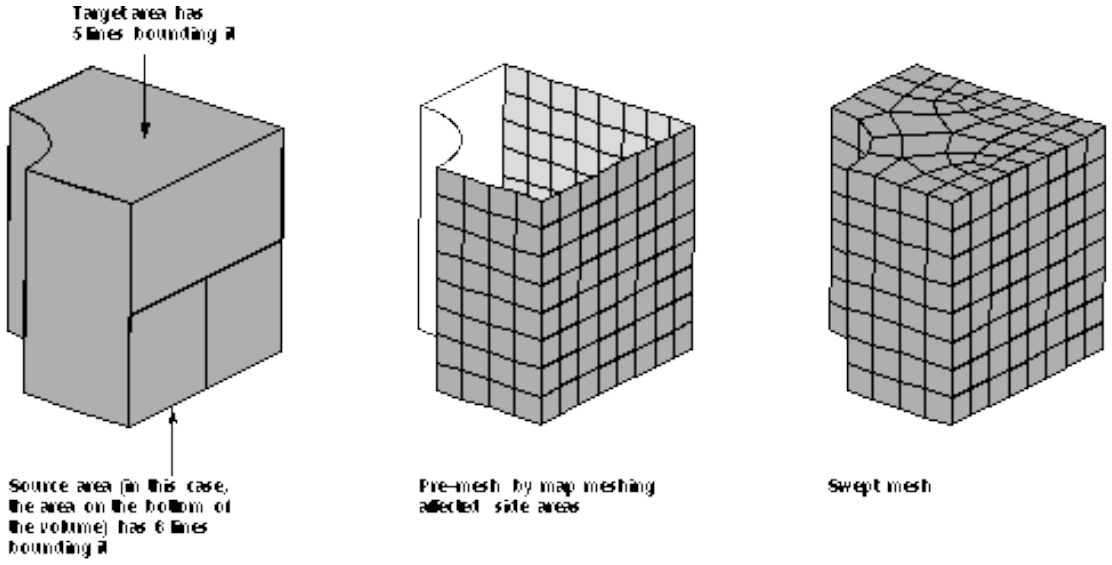
Sweep (rotate) the triangle A(2,2), B(5,7), C(-2,-5) around x-axis and generate the surface

solution:

$$Q(s,t) = [P]\,[T(s)]$$

$$= \begin{pmatrix} 2 & 2 & 0 & 1 \\ 5 & 7 & 0 & 1 \\ -2 & -5 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 2\pi n & \sin 2\pi n & 0 \\ 0 & -\sin 2\pi n & \cos 2\pi n & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Note: The value of n locates various positions on the swept surface.



5.10 Creating a Parametric Cubic Patch

Parametric cubic patch or surface is generated by four boundary curves; the curves are parametric cubic polynomials. The equation of a parametric cubic curve was defined earlier as:

$$= [t^3 \quad t^2 \quad t \quad 1] \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P(0) \\ P(1) \\ P'(0) \\ P'(1) \end{pmatrix}$$

Constant matrix for n = 3 *geometry matrix*

Where P(0) = Coordinates of the first point at t = 0
P(1) = coordinates of the last point at t = 1
P'(0) = values of the slopes in x, y, z directions at t = 0
P'(1) = values of the slopes in x, y, z directions at t = 1

Analogous to a cubic curve, a parametric cubic surface can be defined by 16 points:

- 4 points for coordinates of the corner points
- 8 points for slopes in the s & t directions
- 4 points for twist vectors (second derivatives)

Using a procedure similar to the one carried out in the derivation of the cubic curve, we can derive the geometric coefficient matrix for the surface, which is given as,

$$[G]_H = \left(\begin{array}{cc:cc} P(0,0) & P(0,1) & P_t(0,0) & P_t(0,1) \\ P(1,0) & P(1,1) & P_t(1,0) & P_t(1,1) \\ \hdashline P_s(0,0) & P_s(0,1) & P_{st}(0,0) & P_{st}(0,1) \\ P_s(1,0) & P_s(1,1) & P_{st}(1,0) & P_{st}(1,1) \end{array}\right)$$

Which can be broken into 4 groups as,

$$\left(\begin{array}{c:c} \text{Position of corner points,} & \text{Derivatives w.r.t. t of corner points} \\ \hdashline \text{Derivatives w.r.t s at corner points,} & \text{Cross derivatives at corner points} \end{array}\right)$$

Twist vectors, not shown here, are the partial derivatives: dP_s/dt & dP_t/ds. These vectors control the internal shape of the surface.

With the geometric coefficient matrix defined, the equation of the surface can be written as,

$$P(s.t) = [s]\,[M]_H\,[G]_H\,[M_H]^T\,[t]^T$$

Where: $[s] = [s^3 \quad s^2 \quad s \quad 1]$
$[M]_H$ = [Constant matrix for n = 3]
$[M_H]^T$ = Transpose of $[M]_H$
$[G]_H$ = Geometry matrix as defined by the 16 points, and

$$[t]^T = \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

Example:

Given: A parametric cubic surface is defined by its Cartesian components as follows:

$$x(s,t) = [s^3 \quad s^2 \quad s \quad 1] \begin{pmatrix} 3 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 2 & 1 & 1 & 1 \\ 0 & 2 & -1 & 0 \end{pmatrix} \begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

$$y(s,t) = [s^3 \quad s^2 \quad s \quad 1] \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 \\ 1 & 2 & 0 & 2 \end{pmatrix} \begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

$$z(s,t) = [s^3 \quad s^2 \quad s \quad 1] \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 2 & 0 \\ 3 & 1 & 2 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

Obtain the normal vector at the point where s = ½, t = ½

Solution:

$$P(s,t) = [S]\,[M]_H\,[G]_H\,[M_H]^T\,[t]^T = [x(s,t),\ y(s,t),\ z(s,t)]$$

$$X(s,t) = [s]\,[A]_x\,[t]^T$$

Where $[A]_x = [M]_H\,[G]_H\,[M_H]^T$

The normal vector n is given by

$$n = P_{sx}\ P_t$$

where $Ps = \partial P/ds$ & $Pt = \partial P/dt$

$$x(s,t) = [s^3 \quad s^2 \quad s \quad 1] \begin{pmatrix} 3 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 2 & 1 & 1 & 1 \\ 0 & 2 & -1 & 0 \end{pmatrix} \begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

$$x_s(s,t) = [3s^2 \quad 2s \quad 1 \quad 0] \begin{pmatrix} 3 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 2 & 1 & 1 & 1 \\ 0 & 2 & -1 & 0 \end{pmatrix} \begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

$$x_t(s,t) = [s^3 \quad s^2 \quad s \quad 1] \begin{pmatrix} 3 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 2 & 1 & 1 & 1 \\ 0 & 2 & -1 & 0 \end{pmatrix} \begin{pmatrix} 3t^2 \\ 2t \\ 1 \\ 0 \end{pmatrix}$$

at s = 0.5 & t = 0.5

$x_s(s,t) = 4.5313$

$x_t(s,t) = 3.3438$

similarly, we can evaluate $y_s(s,t)$, $y_t(s,t)$, $z_s(s,t)$ and $z_t(s,t)$

$y_s(s,t) = 2.5313$

$y_t(s,t) = 5.5313$

$z_s(s,t) = 6.9375$

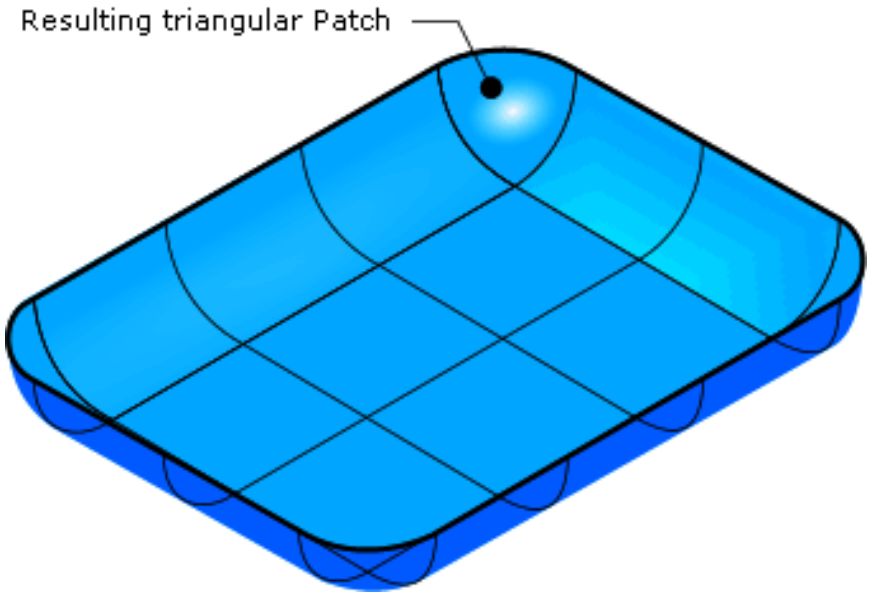
$z_t(s,t) = 5.4375$

And, $P_s(s,t) = [4.5313, 2.5313, 6.9375]$
$P_t(s,t) = [3.3438, 5.5313, 5.4375]$

$$n = P_s(0.5,0.5) \times P_t(0.5,0.5) = \begin{pmatrix} i & j & k \\ 4.5313 & 2.5313 & 6.9375 \\ 3.3438 & 5.5313 & 5.4375 \end{pmatrix}$$

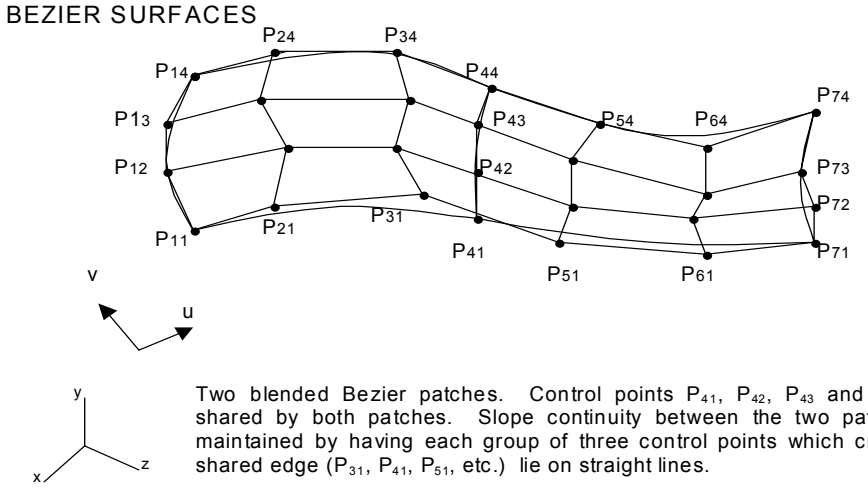
$$= -24.61\, i - 1.4413 j + 16.59\, k$$

Example for cubic patch.



5.11 Bezier Surface

Just as parametric cubic curves are extended to parametric cubic patches, Bezier curves may be extended to Bezier surface patch. While the surface passes through the four corner points, the control points control all other points on the surface.
Using the placement of these points to specify edge slope is more intuitive than determining the parametric slopes and twist vectors for the parametric cubic curve surface.
Bezier surface, as a result, is easier to use because the control points themselves approximate the location of the desired surface. Bezier surfaces can be generated with any order of the Bezier curve. Two surface patches can be joined and the two surfaces do not have to be of the same order, one can be cubic and the other a quadratic.
Blending Bezier patches with slope continuity requires that (1) control points on the common edges be shared and (2) three control points – one on the edge and ones on the either sides of the edge – form a straight line, as shown in the figure below.



Two blended Bezier patches. Control points P_{41}, P_{42}, P_{43} and P_{44} are shared by both patches. Slope continuity between the two patches is maintained by having each group of three control points which cross the shared edge (P_{31}, P_{41}, P_{51}, etc.) lie on straight lines.

In Bezier surface:

- The surface takes the general shape of the control points.
- The surface is contained within the convex hull of the control points.
- The corner of the surface and the corner control points are coincident.

General Equation of the Bezier surface is given as,

$$Q(s,t) = \Sigma \Sigma V_{i,j} B_{i,n}(s) B_{j,m}(t)$$

$$0 \leq s,t \leq 1$$

$V_{i,j}$ defines the control points

$B_{i,n}(s)$ & $B_{j,m}(t)$ are the Bernstein blending functions in the s and t directions.

In matrix form, the Bezier surface can be represented by,

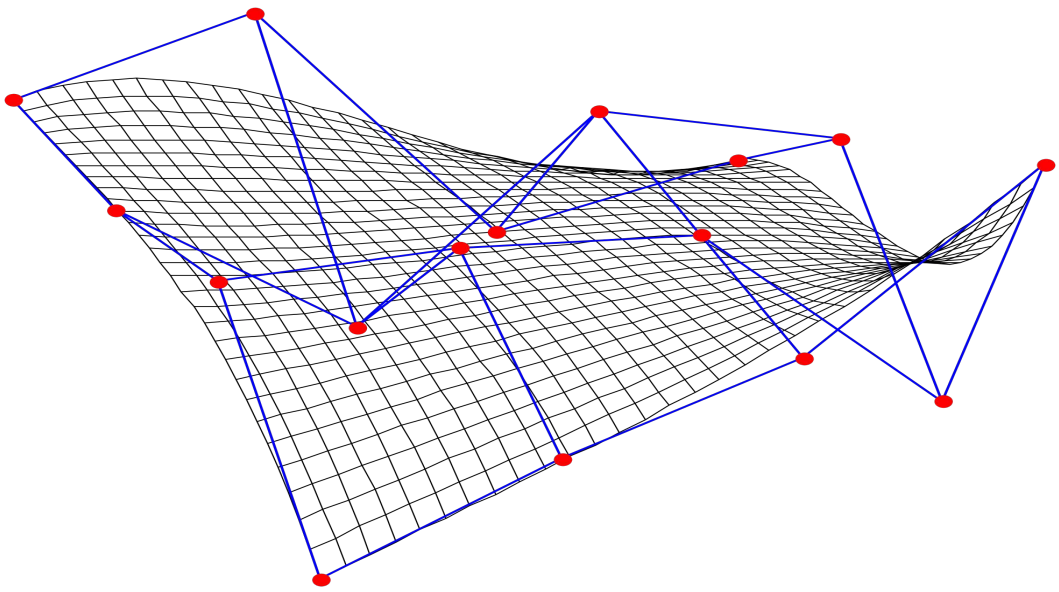
$$Q(s,t) = [S]\,[M]_B\,[V]_B\,[([M)^T]_B\,[t]^T$$

For a cubic surface this equation reduces to:

$$Q(s,t) = [s^3 \quad s^2 \quad s \quad 1] \begin{pmatrix} -1 & 3 & -3 & 1 \\ -3 & -6 & 3 & 0 \\ 3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} v_{0,0} & v_{0,1} & v_{0,2} & v_{0,3} \\ v_{1,0} & v_{1,1} & v_{1,2} & v_{1,3} \\ v_{2,0} & v_{2,1} & v_{2,2} & v_{2,3} \\ v_{3,0} & v_{3,1} & v_{3,2} & v_{3,3} \end{pmatrix} x$$

$$\begin{pmatrix} 1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

Note that, to represent a cubic Bezier surface, 16 control points must be specified, and several Bezier surfaces can be combined to create a complex surface.

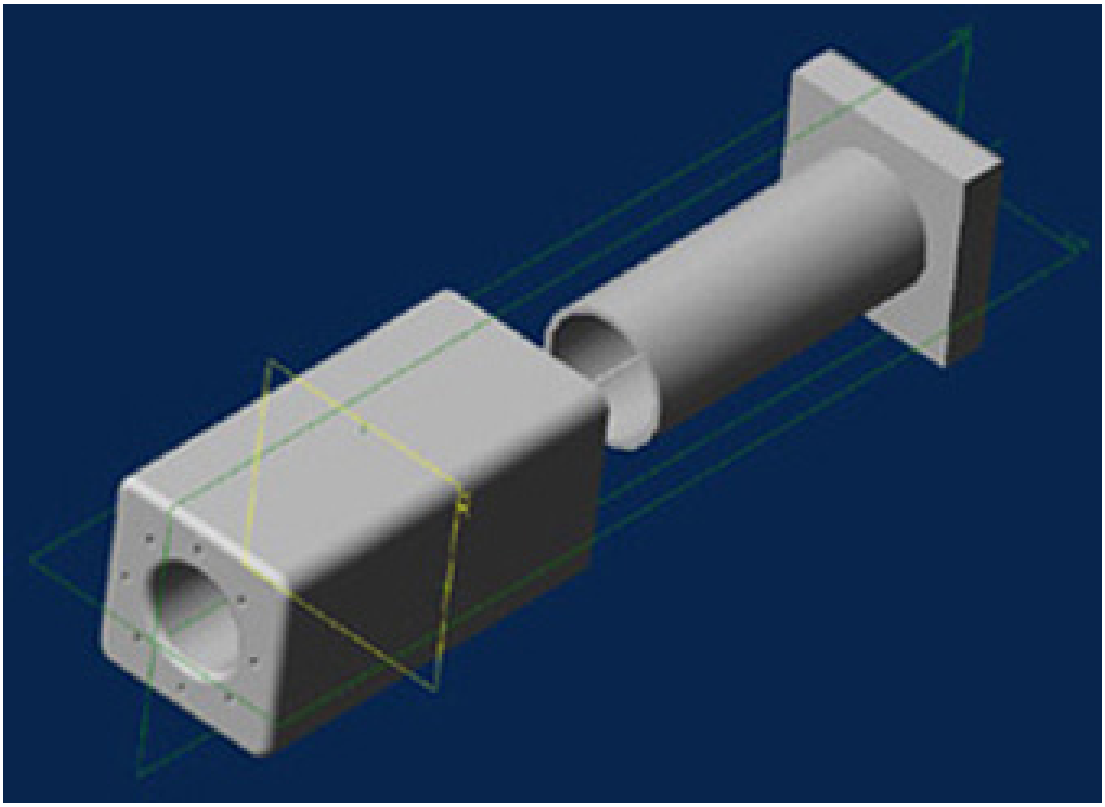


SOLID modelling

6.1 Application of Solid Models

In mechanical engineering, a solid model is used for the following applications:

1. **Graphics**: generating drawings, surface and solid models
2. **Design**: Mass property calculation, interference analysis, finite element modelling, kinematics and mechanism analysis, animation, etc.
3. **Manufacturing**: Tool path generation and verification, process planning, dimension inspection, tolerance and surface finish.
4. **Component Assembly**: Application to robotics and flexible manufacturing: Assembly planning, vision algorithm, kinematics and dynamics driven by solid models.



6.2 Solid Model Representation

There are three different forms in which a solid model can be represented in CAD:

- Wireframe Model
- Surface Model
- Solid Model

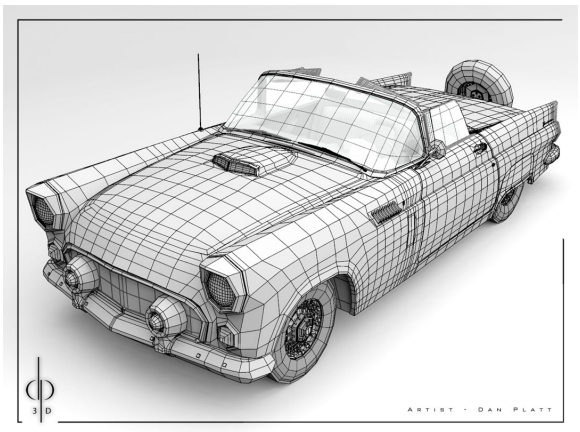
Wireframe Models: Joining points and curves creates wireframe models. These models can be ambiguous and unable to provide mass property calculations, hidden surface removal, or generation of shaded images. Wireframe models are mainly used for a quick verification of design ideas.

Surface Models: Surface models are created using points, lines, and planes. A surface model is unable to identify points that do not lie on the surface, and therefore, the moment of inertia, volume, or sections of the model cannot be obtained. A surface model can be shaded for better visibility. Surface models are used for modelling surfaces of engineering components.

Solid models: Solid models are the most preferred form of CAD models. and represent unambiguous image of a component. A solid model can be used to analyze the moment of inertia, mass, volume, sections of the model, etc.

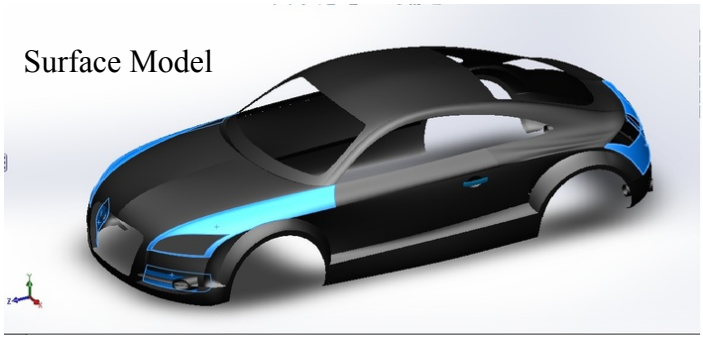
Solid models are mathematical models of objects in the real world that satisfy specific properties, listed below.

1. **Bounded**: The boundary must limit and contain the interior of the solid.
2. **Homogeneously Three-Dimensional**: No dangling edges or faces be present so that the boundary is always in contact with the interior of the solid.
3. **Finite**: The solid must be finite in size.



Wireframe Model

Surface Model



Solid Model

6.3 Solid Model Creation Scheme

A solid model can be generated by the following schemes.

1. Constructive Solid Geometry (CSG)
2. Boundary Representation (B-Rep)
3. Sweeping

A brief description of these schemes follows.

6.3.1 Constructive Solid Geometry Scheme

This scheme is based on the principle that two primitives can be combined to produce a new solid model. This method is also known as 'Building Block' method. The scheme uses the Union, Intersection, and Subtraction techniques to create three-dimensional models, which are based on the Boolean operation. The steps involved in generating a solid model are:

1. Select the primitives from a library
2. Go through the scaling, dimension modification, and any other transformations.
3. Combine the primitives to create the desired solid model.

Since CSG method uses solid primitives, internal details of the object are automatically contained in the model. The model can be sectioned to reveal internal details and can be used for calculating mass, volume, moment of inertia, etc.

New solid models can be created from the primitives or other solid models by the following operations:

- **Union (U)**: Two solids are joined and the common volume of one of the primitives is neglected in the resulting solid.
- **Subtraction or Difference (-)**: One solid is subtracted from the other and the resultant solid retains only the uncut portion of the solid.
- **Intersection (∏):** When two solids are combined, the resultant solid represents the common volume of the two solids.

The most common primitive solids found in a CAD program are:
Block, Cylinder, Cone, Sphere, Wedge, and Torus.

6.3.2 Boundary Representation (B-Rep) Scheme

This scheme is based on the concept that a physical object is bounded by a set of faces. A solid model is created by combining faces and contains vertices, edges, loops, and bodies. Only the boundary surfaces of the model are stored and the volumetric properties are calculated by the Gauss Divergence theorem, which relates volume integral to surface integrals. This scheme can model a variety of solids depending on the primitive surfaces (planar, curved, or sculptured). There are two types of solid models in this scheme:

1. Polyhedral solids
2. Curved solids

1. Polyhedral Solids: Polyhedral models consist of straight edges, e.g., a non-cylindrical surface: box, wedge, combination of two or more non-cylindrical bodies, etc. Polyhedral solids can have blind or through holes, and two or three-dimensional faces, with no dangling edges. A valid polyhedral abides by the Euler's equation:

$$F - E + V - L = 2 (B-G)$$

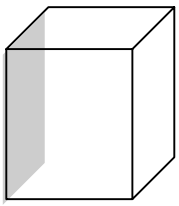
Where,

F = Face
E = Edge
V = Vertices
L = Inner Loop
B = Bodies
G = Through holes

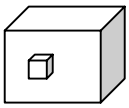
A simple polyhedral has no holes; each face is bounded by a single set of connected edges (bounded by one loop of edges).

Euler's equation for a simple polyhedral can be reduced to: F - E + V = 2

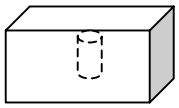
Example: For the box shown, F = 6, E = 12, and V = 8



Examples of other types of polyhedral are shown below.

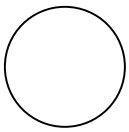


Polyhedral with two loops

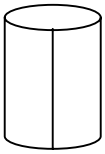


Polyhedral with a blind hole

2. Curved Solids: A curved solid is similar to a polyhedral object but it has curved faces and edges. Spheres and cylinders are examples of curved solids.



Sphere with F = 1, V = 1, E = 0



Cylinder: F = 3, E = 3, V= 2

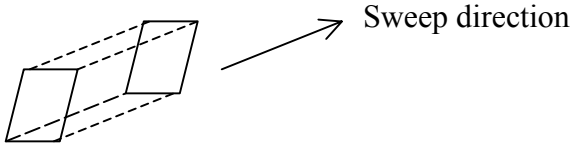
Primitives: In B-rep, a model is made up of the following primitives:

- **Vertex**: A point in space
- **Edge**: A finite, no-intersecting space curve bounded by two vertices that are not necessarily distinct.
- **Face**: A finite connected, non-self-intersecting, region of a closed oriented surface, bounded by one or more loops.
- **Loop**: An ordered alternating sequence of vertices and edges. A loop defines a non-self-intersecting closed space curve, which may be a boundary of a face.
- **Body**: Entity that has faces, edges and vertices. A minimum body is a point.

B-rep scheme is closely related to the traditional drafting method.

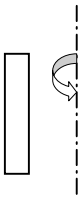
6.3.3 Sweeping Scheme

Sweeping can create a solid model. The method is useful for creating 2 ½ – dimension models. The generated models are axisymmetric and have uniform thickness (i.e., extruded models). There are two types of sweeps: linear and rotational. In linear sweep, a closed 2-D sketch is extruded through the desired length, creating a homogeneous and axisymmetric model, as shown in the figure.



Linear sweep – Creating a box by sweeping a rectangle

In rotational sweep, a closed sketch is rotated around an axis. The generated model is always axisymmetric.



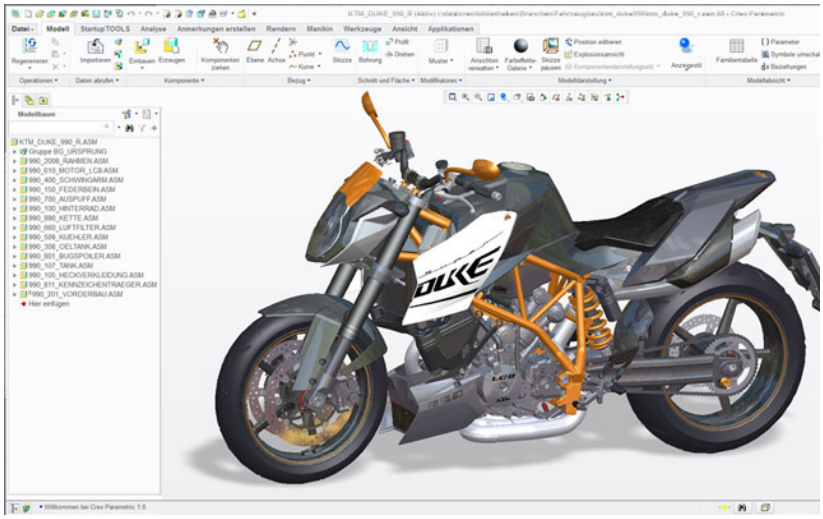
In addition to the two sweeps described above, a model can also be created by a non-linear sweep. In this type of sweep, a closed sketch is swept along a non-linear path.

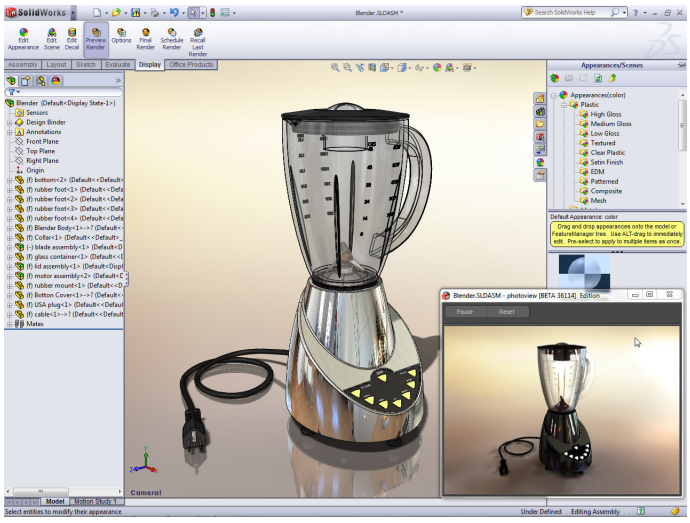
6.4 Commercial Modelers

Most commercial software use the sweeping scheme. ProE and Solidworks are good examples of software that utilize sweep technique to generate a 3-D model. In both of these programs, a 2-D sketch is created and extruded to generate a 3-D base-model. The base model is then used to add or modify features. Most engineering components can be created by this technique.

Relatively new software, Ironcad utilizes the CSG technique to create 3-D models. There are pre-build models in the library (catalog) of the software that can be retrieved and modified as needed. The availability of the primitives (basic solid models) accelerates the process of model generation.

All the three software mentioned above are parametric modelers. Another popular software, AutoCAD is capable of generating 3-D models; however, this software is basically a 2-D modeler and lacks the parametric feature. AutoCAD is capable of creating a solid model with CSG, B-rep, and the sweep methods, but limited to only very simple models.





Finite Elemental Analysis *(FEA)*

CHAPTER 1

An Overview of the Finite Element Analysis

1.1 Introduction

Finite element analysis (FEA) involves solution of engineering problems using computers. Engineering structures that have complex geometry and loads, are either very difficult to analyze or have no theoretical solution. However, in FEA, a structure of this type can be easily analyzed. Commercial FEA programs, written so that a user can solve a complex engineering problems without knowing the governing equations or the mathematics; the user is required only to know the geometry of the structure and its boundary conditions. FEA software provides a complete solution including deflections, stresses, reactions, etc.

In order to become a skillful FEA user, a thorough understanding of techniques for modelling a structure, the boundary conditions and, the limitations of the procedure, are very crucial. Engineering structures, e.g., bridge, aircraft wing, high-rise buildings, etc., are examples of complex structures that are extremely difficult to analyze by classical theory. But FEA technique facilitates an easier and a more accurate analysis. In this technique the structure is divided into very small but finite size elements (hence the name finite element analysis). Individual behavior of these elements is known and, based on this knowledge; behavior of the entire structure is determined.

FEA solution of engineering problems, such as finding deflections and stresses in a structure, requires three steps:

1. Pre-process or modelling the structure
2. Analysis
3. Post processing

A brief description of each of these steps follows.

Step1: Pre-process or modelling the structure
Using a CAD program that either comes with the FEA software or provided by another software vendor, the structure is modeled. The final FEA model consists of several elements that collectively represent the entire structure. The elements not only represent segments of the structure, they also simulate it's mechanical behavior and properties.

Regions where geometry is complex (curves, notches, holes, etc.) require increased number of elements to accurately represent the shape; where as, the regions with simple geometry can be represented by coarser mesh (or fewer elements). The selection of proper elements requires prior experience with FEA, knowledge of structure's behavior, available elements in the software and their characteristics, etc. The elements are joined at the nodes, or common points.

In the pre-processor phase, along with the geometry of the structure, the constraints, loads and mechanical properties of the structure are defined. Thus, in pre-processing, the entire structure is completely defined by the geometric model. The structure represented by nodes and elements is called "mesh".

Step 2: Analysis
In this step, the geometry, constraints, mechanical properties and loads are applied to generate matrix equations for each element, which are then assembled to generate a global matrix equation of the structure. The form of the individual equations, as well as the structural equation is always,

$$\{F\} = [K]\{u\}$$

Where
$\{F\}$ = External force matrix.
$[K]$ = Global stiffness matrix
$\{u\}$ = Displacement matrix

The equation is then solved for deflections. Using the deflection values, strain, stress, and reactions are calculated. All the results are stored and can be used to create graphic plots and charts in the post analysis.

Step 3: Post processing
This is the last step in a finite element analysis. Results obtained in step 2 are usually in the form of raw data and difficult to interpret. In post analysis, a CAD program is utilized to manipulate the data for generating deflected shape of the structure, creating stress plots, animation, etc. A graphical representation of the results is very useful in understanding behavior of the structure
.

1.2 History of FEA

Engineering applications of finite element analysis is approximately 40 years old. Evolution of FEA is tied with the development in computer technology. With the enhancement in computer speed and storage capacity, FEA has become a very valuable engineering tool. NASA is credited for developing comprehensive FEA software in 1960's, known as NASTRAN. Rights of the software were purchased by McNeal Schwendler Corporation, who refined it and commercially marketed it under the name, MSC-NASTRAN. The first college course in FEA was offered in 1970. In the

early 1970's, application of FEA was limited to large corporations, who can afford expensive mainframe computers. However, in 1980's, with the introduction of desktop computers, application of FEA became popular and indispensable engineering tool. In late 80's, almost all the major FEA vendors introduced their software that can run on a PC.

In the past ten years, there were several significant development in FEA, including:

.

- Introduction of P- elements.
- Integration of sensitivity analysis and optimization capabilities.
- Availability of faster and cheaper desktop computers to run FEA software that previously required mainframe computers.
- Development of powerful CAD programs for modelling complex structures.
- Making software user-friendly.

1.3 How FEA works – Within software

The following steps can summarize FEA procedure that works inside software:

- Using the user's input, the given structure is graphically divided into small elements (sections or regions) so that each and every element's mechanical behavior can be defined by a set of differential equations.
- The differential equations are converted into algebraic equation, and then into matrix equations, suitable for a computer-aided solution.
- The element equations are combined and a global structural equation is obtained.
- Appropriate load and boundary conditions, supplied by the user, are incorporated in to the structural matrix.
- The structural matrix is solved and deflections of all the nodes are calculated.
- A node can be shared by several elements and the deflection at the shared node represents deflection of the sharing elements at the location of the node.
- Deflection at any other point in the element is calculated by interpolation of all the node points in the element.
- An element can have a linear or higher order interpolation function.
- The individual element matrix equations are assembled into a combined structure equation of the form $\{F\}=[k]\{u\}$.

As defined earlier,

$\{F\}$ = Column matrix of the externally applied loads.

$[k]$ = Stiffness matrix of the structure, which is always a symmetric matrix. This matrix is analogues to an equivalent spring constant of several connected springs.

{u} = Column matrix representing the deflection of all the node points, that results when the load {F} is applied.

1.4 How FEA works – User's interaction

The above described software procedure is mostly transparent to the user. A user has the following interaction with the software, through user's computer.

- Create the geometry, representing the structure: A CAD modelling software is used to create the structure's geometry.
- Provide the material properties, loads, constraints, etc.
- Analyze the result data.

1.5 Convergence – Assuring Optimum Mesh Size

How do we determine the exact number of elements for a structure and make sure that the FEA mesh is optimum? There is no exact answer to this question; however, if we keep refining a mesh until the variation in the result is less than a specified value, we will reach the desirable mesh density. Convergence refers to this process, where we optimize the mesh to arrive at the desired results. In general, there are three types of convergences:

1. **Von-Mises Stress (VMS) convergence**: Mesh is refined until the percentage variation in VMS is less than 1, 5, 10 or any given value selected by the user. VMS convergence should be avoided if there are stress concentration points, convergence will be difficult to achieve.
2. **Strain Energy Convergence:** Mesh is refined until the percentage variation in the average strain of elements is less than a chosen value. Strain convergence is a better criterion for optimizing an FEA mesh. Stress concentrations points do not significantly influence the average strain energy of elements and variation in strain energy is influenced by mesh size or polynomial order of the elements only.
3. **Deflection Convergence:** It is similar to the above convergences, except, node deflection values are used for the convergence criterion.

1.6 H- versus P- elements

In FEA, there are two types of elements:

1. H-elements and,
2. P-elements

H-element is the original and "classic" element. The name is derived from the field of numerical analysis, where the letter 'h' is used for the step size, to achieve convergence in the analysis. The h-element is always of low order, usually, linear or quadratic. When a finite element mesh is refined to achieve convergence, the procedure is called h-convergence. For h-elements, convergence is accomplished at the expense of excessively

large number of elements. The high stress concentration regions require a very fine mesh, thereby increasing the number of elements. Finite elements used by commercial programs in the 1970s and 80s, were all h-elements. However, with improvement in computer power and efficiency, a much more useful, p-elements were developed.

P-elements are relatively new, developed in late 1980s and offer not only the traditional static analysis, they provides option of optimizing a structure. P-elements can have edge-polynomial as high as 9^{th} order, unlike the low order polynomials of h-elements. The high polynomial edge order of p-elements makes it possible to model a curved edge of a structure with accuracy. Therefore, fewer elements can be used to achieve convergence. In FEA, the number of elements in the mesh usually remains fixed; convergence is achieved by increasing the polynomial order of the p-elements, rather than refinement of the mesh. For optimization, as the dimensions of the structure being analyzed are changed, the number of elements remains constant. Only, the polynomial order of the elements is changed as needed.

1.7 Bottom-up and Top-down approach

When modelling a structure (creating an FEA model), bottom – up approach refers to creation of model by defining the geometry of the structure with nodes and elements. These nodes and elements represent the physical structure. When an FEA model is created by this procedure, it is known as a bottom-up approach. This is the original procedure for creating FEA mesh, and requires a substantial investment in time and skill. When this method is employed, most of analyst's time is devoted to creation of the mesh, and only a fraction of time is spent for analysis and results interpretation.

In FEA, a top-down procedure refers to creation of FEA mesh by first building a solid model, using a 3-D CAD program, and then dividing the model into nodes and elements. Thus, the top-down method requires building of a geometric model of the structure and then using it to create an FEA mesh. The advantages of the top-down approach are obvious; we don't have to define the geometry of individual elements in the structure, which can be very time consuming. Obviously, a 3-D model requires high-end computer hardware, along with familiarity with the modelling software.

1.8 Discretization or Division of a structure into small elements

In FEA, an engineering structure is divided into small elements. These elements coincide with the geometry of the structure and represent the geometry and the mechanical properties in the regions.

Selection of elements to represent the structure is a matter of engineering judgement and prior experience with FEA procedure. A sound advice for beginners is: keep the elements size small enough to yield good results and yet large enough to reduce computational time. Smaller elements are desirable where the results are changing rapidly (change in

geometry, sharp corners, etc.). Large elements can be used where the results (deflection or stresses) are relatively constant.

In FEA, discretization of a structural model is another name for mesh generation. Most of the commercial FEA programs have the capability of automatically generating FEA mesh. User has to provide the element type, mechanical properties, constraints and loads.

1.9 Element types

Let us assume that we wish to find stress concentration in a steel plate with holes. For the FEA analysis of this plate, we would need elements that have shapes of triangular plates, quadrilateral plates, and plates with curved edge. Then these elements can replace and represent each and every part of the plate, including the circular edges near the hole.

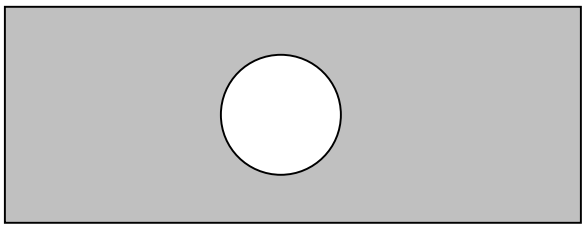


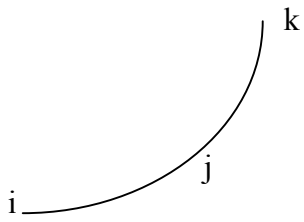
Plate with a hole

Thus, we need elements that have geometric shape similar to the real structure or region of the structure that is being modeled. One geometric shape cannot represent all possible engineering structural shapes. Therefore, we need elements that look like a plate, beam, cylinder, sphere, etc. However, in FEA, almost all structures can be approximated by the following basic elements:

1. **Line elements**: Element consisting of two nodes.

Example: Truss and beam elements.

In computers, a line, connecting two nodes at its ends as shown, represents a line element. The cross-sectional area is assumed constant throughout the element.

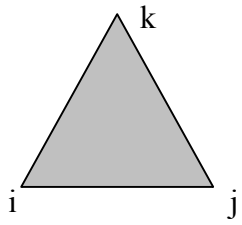


The element can have more than two nodes, and can be a curved rather than a straight line.

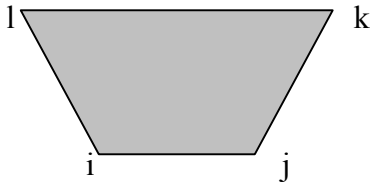
2. **2-D solid elements**: Elements that have geometry similar to a flat plate.

Example: Plane stress, plain strain, plates, shells, and axisymmetric elements.

2-D solid elements are plane elements, with constant thickness, and have either a triangular or quadrilateral shape, with 3 nodes or 4 nodes as shown.



2-D Solid: Triangular



2-D Solid: Quadrilateral

For higher order 2-D elements, the number of nodes can vary. For example, the element edges can be quadratic with 3 nodes on each edge. However, in most FEA analysis, only the straightedge elements are used.

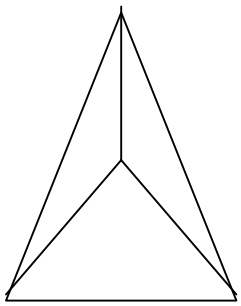
Loads on 2-D solid elements can be applied only in its plane, and deflections also occur only in the plane of the elements.

Axisymmetric element is a special case of 2-D plane stress element. We will discuss this element in detail later on.

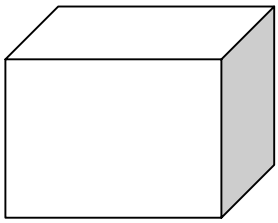
3. **3-D solid elements**: Element that have a 3-D geometry.

Example: Tetrahedron and hexahedron elements.

The basic 3-D solid elements have either a tetrahedral (4 faces) or hexahedral (6 faces) shape, as shown.



Tetrahedral - 4-nodes



Hexahedral - 8-nodes

The basic elements have corner nodes and straight edges, but the number of nodes and edge geometry can vary.

NOTES

1 For an accurate analysis in FEA, selection of the proper elements is very important. The selected elements must represent the engineering structure as close to the original structure as possible.

2 In addition to these basic elements, there are some special application elements, e.g., mass element and contact element. Almost all other special purpose elements can be derived from the three basic groups of the elements described above.

CHAPTER 2

The Basic FEA Procedure

2.1 Introduction

This chapter discusses the spring element, especially for the purpose of introducing various concepts involved in use of the FEA technique. A spring element is not very useful in the analysis of real engineering structures; however, it represents a structure in an ideal form for an FEA analysis. Spring element doesn't require discretization (division into smaller elements) and follows the basic equation F = ku. We will use it solely for the purpose of developing an understanding of FEA concepts and procedure.

2.2 Overview

Finite Element Analysis (FEA), also known as finite element method (FEM) is based on the concept that a structure can be simulated by the mechanical behavior of a spring in which the applied force is proportional to the displacement of the spring and the relationship F = ku is satisfied. In FEA, structures are modeled by a CAD program and represented by nodes and elements. The mechanical behavior of each of these elements is similar to a mechanical spring, obeying the equation, F = ku. Generally, a structure is divided into several hundred elements, generating a very large number of equations that can only be solved with the help of a computer.

The term 'finite element' stems from the procedure in which a structure is divided into small but finite size elements (as opposed to an infinite size, generally used in mathematical integration). The endpoints or corner points of the element are called nodes. Each element possesses its own geometric and elastic properties. Spring, Truss, and Beams elements, called line elements, are usually divided into small sections with nodes at each end. The cross-section shape doesn't affect the behavior of a line element; only the cross-sectional constants are relevant and used in calculations. Thus, a square or a circular cross-section of a truss member will yield exactly the same results as long as the cross-sectional area is the same. Plane and solid elements require more than two nodes and can have over 8 nodes for a 3 dimensional element.

A line element has an exact theoretical solution, e.g., truss and beam elements are governed by their respective theories of deflection and the equations of deflection can be found in an engineering text or handbook. However, engineering structures that have stress concentration points e.g., structures with holes and other discontinuities do not have a theoretical solution, and the exact stress distribution can only be found by an experimental method. However, the finite element method can provide an acceptable solution more efficiently. Problems of this type call for use of elements other than the line elements mentioned earlier, and the real power of the finite element is manifested. In order to develop an understanding of the FEA procedure, we will first deal with the spring element. In this chapter, spring structures will be used as building blocks for developing an understanding of the finite element analysis procedure. Both spring and truss elements give an easier modelling overview of the finite element analysis procedure, due to the fact that each spring and truss element, regardless of length, is an ideally sized element and do not need any further division. Therefore, in the following sections spring structures will be used to illustrate the finite element analysis procedure.

2.3 Understanding Computer and FEA software interaction - Using the Spring Element as an example

In the following example, a three-element structure is analyzed by finite element method. The analysis procedure presented here will be exactly the same as that used for a complex structural problem, except, in the following example, all calculations will be carried out by hand so that each step of the analysis can be clearly understood. All derivations and equations are written in a form, which can be handled by a computer, since all finite element analyses are done on a computer. The finite element equations are derived using Direct Equilibrium method. The example illustrates the interaction between computer and the FEA software used for solution.

Example 2.1

Two springs are connected in series with spring constant k_1, and k_2 (lb./in) and a force F (lb.) is applied. Find the deflection at nodes 2, and 3.

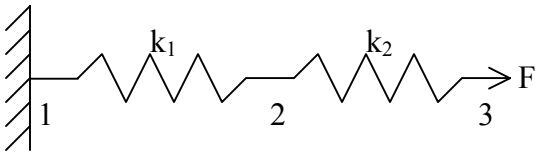


Figure 2.1

Solution:

For finite element analysis of this structure, the following steps are necessary:

Step 1: Derive the element equation for each spring element.

Step 2: Assemble the element equations into a common equation, knows as the global or Master equation.

Step 3: Solve the global equation for deflection at nodes 1 through 3.

Step 1: Derive the element equation for each spring element.

First, a general equation is derived for an element e that can be used for any spring element and expressed in terms of its own forces, spring constant and node deflections, as illustrated in figure 2.2.

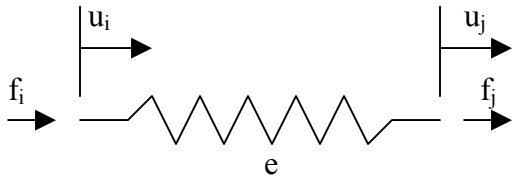


Figure 2.2

Element 'e' can be thought of as any element in the structure with nodes i and j, forces f_i and f_j, deflections u_i and u_j, and the spring constant k_e. Node forces f_i and f_j are internal forces and are generated by the deflections u_i and u_j at nodes i and j, respectively.

For a linear spring f = ku, and

$$f_i = k_e(u_j - u_i) = -k_e(u_i - u_j) = -k_e u_i + k_e u_j$$

For equilibrium, $f_j = -f_i = k_e(u_i - u_j) = k_e u_i - k_e u_j$

Or

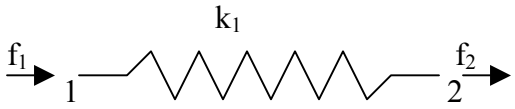
$$-f_i = k_e u_i - k_e u_j$$
$$-f_j = -k_e u_i + k_e u_j$$

Writing these equations in a matrix form, we get

$$\begin{Bmatrix} -f_i \\ -f_j \end{Bmatrix} = \begin{bmatrix} k_e & -k_e \\ -k_e & k_e \end{bmatrix} \begin{Bmatrix} u_i \\ u_j \end{Bmatrix}$$

The above matrix equation is a general form of an equation of a spring elements, and can be used to derive element equations for any spring element in this example, and in general, it is valid for any linear spring element. Thus, equations for each elements can be written as follows:

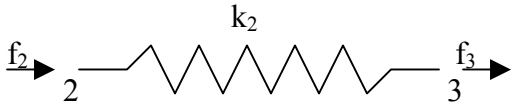
Element 1:



$$\begin{Bmatrix} -f_1 \\ -f_2 \end{Bmatrix}^{(1)} = \begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix}$$

Where, the superscript on the force matrix indicates the corresponding element.

Element 2:



$$\begin{Bmatrix} -f_2 \\ -f_3 \end{Bmatrix}^{(2)} = \begin{bmatrix} k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} u_2 \\ u_3 \end{Bmatrix}$$

Thus,

$f_1^{(1)} = -k_1(u_1 - u_2)$ $\quad$ $f_2^{(1)} = k_1(u_1 - u_2)$

$f_2^{(2)} = -k_2(u_2 - u_3)$ $\quad$ $f_3^{(2)} = k_2(u_2 - u_3)$

This completes the procedure for step 1.

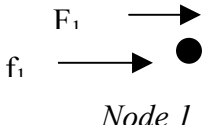
Note that $f_3 = F$ (lb.). This will be substituted in step 2. The above equations represent individual elements only and not the entire structure.

Step 2 : Assemble the element equations into a global equation.

The basis for combining or assembling the element equation into a global equation is the equilibrium condition at each node. When the equilibrium condition is satisfied by summing all forces at each node, a set of linear equations is created which links each element force, spring constant, and deflections. In general, let the external forces at each node be F_1, F_2, and F_3, as shown in figure 2.3. Using the equilibrium equation, we can find the element equations, as follows.

$$Node1: \sum F = 0 = f_1 + F_1$$

$$or \quad F_1 = -f_1 = k_1(u_1 - u_2) = k_1u_1 - k_1u_2$$



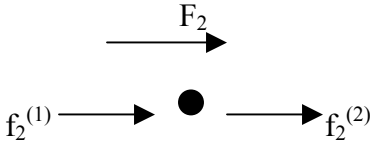
Node 1

Node 2: $\sum F = 0 = f_2^{(1)} + f_2^{(2)} + F_2$

Or $F_2 = -f_2^{(1)} - f_2^{(2)} =$

$= -k_1(u_1 - u_2) + k_2(u_2 - u_3)$

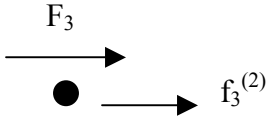
$= -k_1u_1 - k_1u_2 + k_2u_2 - k_2u_3$



Node 2

Node 3: $\sum F = 0$, $f_3^{(2)} + F_3 = 0$

Or $F_3 = -f_3^{(2)} = -k_2 (u_2 - u_3)$



Node 3

Figure 2.3

The superscript "e" in force $f_n^{(e)}$ indicates the contribution made by the element number e, and the subscript "n" indicates the node "n" at which forces are summed.
Rewriting the equations, we get,

$$\begin{aligned} k_1 u_1 - k_1 u_2 &= F1 \\ - k_1 u_1 + k_1 u_2 + k_2 u_2 - k_2 u_3 &= F2 \\ - k_2 u_2 + k_2 u_3 &= F3 \end{aligned} \qquad (2.1)$$

These equations can now be written in a matrix form, giving

$$\begin{pmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1+k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ F_3 \end{pmatrix}$$

This completes step 2 for assembling the element equations into a global equation. At this stage, some important conceptual points should be emphasized and will be discussed below.

2.3.1 Procedure for Assembling Element stiffness matrices

The first term on the left hand side in the above equation represents the stiffness constant for the entire structure and can be thought of as an equivalent stiffness constant, given as

$$[K_{eq}] = \begin{pmatrix} k_1 & -k_1 & 0 & 0 \\ -k_1 & k_1+k_2 & -k_2 & 0 \\ 0 & -k_2 & k_2+k_3 & -k_3 \\ 0 & 0 & -k_3 & k_3 \end{pmatrix}$$

A single spring element with a value K_{eq} will have an identical mechanical property as the structural stiffness in the above example.

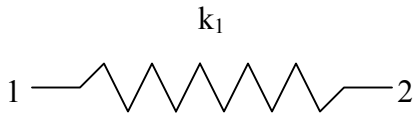
The assembled matrix equation represents the deflection equation of a structure without any constraints, and cannot be solved for deflections without modifying it to incorporate the boundary conditions. At this stage, the stiffness matrix is always symmetric with corresponding rows and columns interchangeable.

The global equation was derived by applying equilibrium conditions at each node. In actual finite element analysis, this procedure is skipped and a much simpler procedure is used. The simpler procedure is based on the fact that the equilibrium condition at each node must always be satisfied, and in doing so, it leads to an orderly placement of individual element stiffness constant according to the node numbers of that element. The procedure involves numbering the rows and columns of each element, according to the node numbers of the elements, and then, placing the stiffness constant in its

corresponding position in the global stiffness matrix. Following is an illustration of this procedure, applied to the example problem.

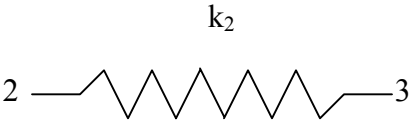
Element 1:

$$K^{(1)} = \begin{matrix} \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 \end{matrix} \\ \begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix} \end{matrix}$$



Element 2:

$$K^{(2)} = \begin{matrix} \\ 2 \\ 3 \end{matrix} \begin{matrix} \begin{matrix} 2 & 3 \end{matrix} \\ \begin{bmatrix} k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \end{matrix}$$



Assembling it according with the above-described procedure, we get,

$$[K_g] = \begin{matrix} \\ 1 \\ 2 \\ 3 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \end{matrix}$$

Note that the first constant k_1 in row 1 and column 1 for element 1 occupies the row 1 and column 1 in the global matrix. Similarly, for element 2, the constant k_2 in row 2 and column 2 occupies exactly the same position (row 2 and column 2) in the global matrix, etc.

In a large model, the node numbers can occur randomly, but the assembly procedure remains the same. It's important to place the row and column elements from an element into the global matrix at exactly the same position corresponding to the respective row and column.

2.3.2 Force matrix

At this stage, the force matrix is represented in a general form, with unknown forces F_1, F_2, and F_3

$$\begin{Bmatrix} F_1 \\ F_2 \\ F_3 \end{Bmatrix}$$

Representing the external forces at nodes 1, 2, and 3, in general terms, and not in terms of the actual known value of the forces. In the example problem, $F_1 = F_2 = 0$ and $F_3 = F$. the actual force matrix is then

$$\begin{Bmatrix} 0 \\ 0 \\ F \end{Bmatrix}$$

Generally, the assembled structural matrix equation is written in short as {F}=[k]{u}, or simply, F = k u, with the understanding that each term is an m x n matrix where m is the number of rows and n is the number of columns.

Step 3: Solve the global equation for deflections at nodes.

There are two steps for obtaining the deflection values. In the first step, all the boundary conditions are applied, which will result in reducing the size of the global structural matrix. In the second step, a numerical matrix solution scheme is used to find deflection values by using a computer. Among the most popular numerical schemes are the Gauss elimination and the Gauss-Sedel iteration method. For further reading, refer to any numerical analysis book on this topic. In the following examples and chapters, all the matrix solutions will be limited to a hand calculation even though the actual matrix in a finite element solution will always use one of the two numerical solution schemes mentioned above.

2.3.3 Boundary conditions

In the example problem, node 1 is fixed and therefore $u_1 = 0$. Without going into a mathematical proof, it can be stated that this condition is effected by deleting row 1 and column 1 of the structural matrix, thereby reducing the size of the matrix from 3 x 3 to 2 x 2. In general, any boundary condition is satisfied by deleting the rows and columns corresponding to the node that has zero deflection. In general, a node has six degrees of freedom (DOF), which include three translations and three rotations in x, y and z directions. In the example problem, there is only one degree of freedom at each node. The node deflects only along the axis of the spring.

In this section, the finite element analysis procedure for a spring structure has been established. The following numerical example will utilize the derivation and concepts developed above.

Example 2.2

In the given spring structure, $k_1 = 20$ lb./in., $k_2 = 25$ lb./in., $k_3 = 30$ lb./in., F = 5 lb. Determine deflection at all the nodes.

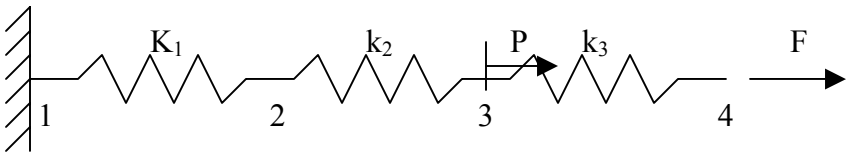


Figure 2.4

Solution

We would apply the three steps discussed earlier.

Step 1: Derive the Element Equations

As derived earlier, the stiffness matrix equations for an element e is,

$$K^{(e)} = \begin{pmatrix} k_e & -k_e \\ -k_e & k_e \end{pmatrix}$$

Therefore, stiffness matrix of elements 1, 2, and 3 are,

Element 1: $K^{(1)} = \begin{matrix} \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 \end{matrix} \\ \begin{pmatrix} 20 & -20 \\ -20 & 20 \end{pmatrix} \end{matrix}$

Element 2: $K^{(2)} = \begin{matrix} \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 \end{matrix} \\ \begin{pmatrix} 25 & -25 \\ -25 & 25 \end{pmatrix} \end{matrix}$

Element 3:

$$K^{(3)} = \begin{matrix} \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 \end{matrix} \\ \begin{pmatrix} 30 & -30 \\ -30 & 30 \end{pmatrix} \end{matrix}$$

Step 2: Assemble element equations into a global equation

Assembling the terms according to their row and column position, we get

$$[K_g] = \begin{matrix} \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{pmatrix} 20 & -20 & 0 & 0 \\ -20 & 20+25 & -25 & 0 \\ 0 & -25 & 25+30 & -30 \\ 0 & 0 & 30 & 30 \end{pmatrix} \end{matrix}$$

Or, by simplifying

$$[K_g] = \begin{pmatrix} 20 & -20 & 0 & 0 \\ -20 & 45 & -25 & 0 \\ 0 & -25 & 55 & -30 \\ 0 & 0 & 30 & 30 \end{pmatrix}$$

The global structural equation is,

$$\begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix} = \begin{pmatrix} 20 & -20 & 0 & 0 \\ -20 & 45 & -25 & 0 \\ 0 & -25 & 55 & -30 \\ 0 & 0 & 30 & 30 \end{pmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}$$

Step 3: Solve for deflections

First, applying the boundary conditions u_1=0, the first row and first column will drop out. Next, $F_1 = F_2 = F_3 = 0$, and $F_4 = 5$ lb. The final form of the equation becomes,

$$\begin{Bmatrix} 0 \\ 0 \\ 5 \end{Bmatrix} = \begin{pmatrix} 45 & -25 & 0 \\ -25 & 55 & -30 \\ 0 & -30 & 30 \end{pmatrix} \begin{Bmatrix} u_2 \\ u_3 \\ u_4 \end{Bmatrix}$$

This is the final structural matrix with all the boundary conditions being applied. Since the size of the final matrices is small, deflections can be calculated by hand. It should be

noted that in a real structure the size of a stiffness matrix is rather large and can only be solved with the help of a computer. Solving the above matrix equation by hand we get,

$$0 = 45\,u_2 - 25\,u_3$$

$$0 = -25\,u_2 + 55\,u_3 - 30\,u_4 \qquad \text{Or} \qquad \begin{Bmatrix} u_2 \\ u_3 \\ u_4 \end{Bmatrix} = \begin{pmatrix} 0.2500 \\ 0.4500 \\ 0.6167 \end{pmatrix}$$

$$5 = -30\,u_3 + 30\,u_4$$

Example 2.3

In the spring structure shown $k_1 = 10$ lb./in., $k_2 = 15$ lb./in., $k_3 = 20$ lb./in., P= 5 lb. Determine the deflection at nodes 2 and 3.

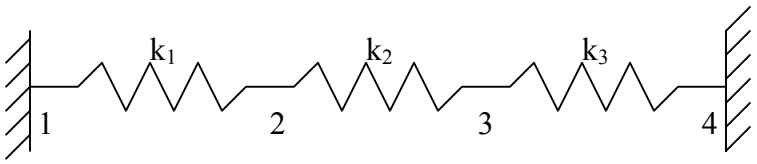


Figure 2.5

Solution:

Again apply the three steps outlined previously.

Step 1: Find the Element Stiffness Equations

Element 1:

$$[K^{(1)}] = \begin{matrix} \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 \end{matrix} \\ \begin{pmatrix} 10 & -10 \\ -10 & 10 \end{pmatrix} \end{matrix}$$

Element 2:

$$[K^{(2)}] = \begin{matrix} \\ 2 \\ 3 \end{matrix} \begin{matrix} \begin{matrix} 2 & 3 \end{matrix} \\ \begin{pmatrix} 15 & -15 \\ -15 & 15 \end{pmatrix} \end{matrix}$$

Element 3:

$$[K^{(3)}] = \begin{matrix} \\ 3 \\ 4 \end{matrix} \begin{matrix} \begin{matrix} 3 & 4 \end{matrix} \\ \begin{pmatrix} 20 & -20 \\ -20 & 20 \end{pmatrix} \end{matrix}$$

Step 2: Find the Global stiffness matrix

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 10 & -10 & 0 & 0 \\ -10 & 10+15 & -15 & 0 \\ 0 & -15 & 15+20 & -20 \\ 0 & 0 & -20 & 20 \end{pmatrix} \end{array} = \begin{pmatrix} 10 & -10 & 0 & 0 \\ -10 & 25 & -15 & 0 \\ 0 & -15 & 35 & -20 \\ 0 & 0 & -20 & 20 \end{pmatrix}$$

Now the global structural equation can be written as,

$$\begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix} = \begin{pmatrix} 10 & -10 & 0 & 0 \\ -10 & 25 & -15 & 0 \\ 0 & -15 & 35 & -20 \\ 0 & 0 & -20 & 20 \end{pmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}$$

Step 3: Solve for Deflections

The known boundary conditions are: $u_1 = u_4 = 0$, $F_3 = P = 3$lb. Thus, rows and columns 1 and 4 will drop out, resulting in the following matrix equation,

$$\begin{Bmatrix} 0 \\ 3 \end{Bmatrix} = \begin{pmatrix} 25 & -15 \\ -15 & 35 \end{pmatrix} \begin{Bmatrix} u_3 \\ u_3 \end{Bmatrix}$$

Solving, we get $u_2 = 0.0692$ & $u_3 = 0.1154$

Example 2.4

In the spring structure shown, $k_1 = 10$ N/mm, $k_2 = 15$ N/mm, $k_3 = 20$ N/mm, $k_4 = 25$ N/mm, $k_5 = 30$ N/mm, $k_6 = 35$ N/mm. F2 = 100 N. Find the deflections in all springs.

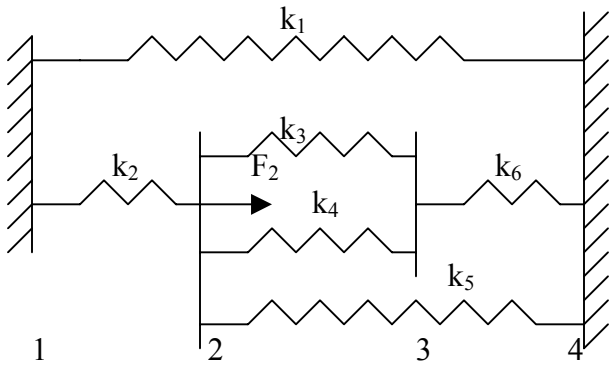


Fig. 2.6

Solution:

Here again, we follow the three-step approach described earlier, without specifically mentioning at each step.

Element 1:

$$[K^{(1)}] = \begin{matrix} \\ 1 \\ 4 \end{matrix} \begin{matrix} \begin{matrix} 1 & 4 \end{matrix} \\ \begin{pmatrix} 10 & -10 \\ -10 & 10 \end{pmatrix} \end{matrix}$$

Element 2:

$$[K^{(2)}] = \begin{matrix} \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 \end{matrix} \\ \begin{pmatrix} 15 & -15 \\ -15 & 15 \end{pmatrix} \end{matrix}$$

Element 3:

$$[K^{(3)}] = \begin{matrix} \\ 2 \\ 3 \end{matrix} \begin{matrix} \begin{matrix} 2 & 3 \end{matrix} \\ \begin{pmatrix} 20 & -20 \\ -20 & 20 \end{pmatrix} \end{matrix}$$

Element 4:

$$[K^{(4)}] = \begin{matrix} \\ 2 \\ 3 \end{matrix} \begin{matrix} \begin{matrix} 2 & 3 \end{matrix} \\ \begin{pmatrix} 25 & -25 \\ -25 & 25 \end{pmatrix} \end{matrix}$$

Element 5:

$$[K^{(5)}] = \begin{matrix} \\ 2 \\ 4 \end{matrix} \begin{matrix} \begin{matrix} 2 & 4 \end{matrix} \\ \begin{pmatrix} 30 & -30 \\ -30 & 30 \end{pmatrix} \end{matrix}$$

Element 6:

$$[K^{(6)}] = \begin{matrix} \\ 3 \\ 4 \end{matrix} \begin{matrix} \begin{matrix} 3 & 4 \end{matrix} \\ \begin{pmatrix} 35 & -35 \\ -35 & 35 \end{pmatrix} \end{matrix}$$

The global stiffness matrix is,

$$[K_g] = \begin{matrix} \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{pmatrix} 10+15 & -15 & 0 & -10 \\ -15 & 15+20+25+30 & -20-25 & -30 \\ 0 & -20-25 & 20+25+35 & -35 \\ -10 & -30 & -35 & 10+30+35 \end{pmatrix} \end{matrix}$$

And simplifying, we get

$$[K_g] = \begin{pmatrix} 25 & -15 & 0 & -10 \\ -15 & 90 & -45 & -30 \\ 0 & -45 & 80 & -35 \\ -10 & -30 & -35 & 75 \end{pmatrix}$$

And the structural equation is,

$$\begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix} = \begin{pmatrix} 25 & -15 & 0 & -10 \\ -15 & 90 & -45 & -30 \\ 0 & -45 & 80 & -35 \\ -10 & -30 & -35 & 75 \end{pmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}$$

Now, apply the boundary conditions, $u_1 = u_4 = 0$, $F_2 = 100$ N. This is carried out by deleting the rows 1 and 4, columns 1 and 4, and replacing F_2 by 100N. The final matrix equation is,

$$\begin{Bmatrix} 100 \\ 0 \end{Bmatrix} = \begin{pmatrix} 90 & -45 \\ -45 & 80 \end{pmatrix} \begin{Bmatrix} u_2 \\ u_3 \end{Bmatrix}$$

Which gives

$$\begin{Bmatrix} u_2 \\ u_3 \end{Bmatrix} = \begin{pmatrix} 1.5459 \\ 0.8696 \end{pmatrix}$$

Deflections:

Spring 1: $u_4 - u_1 = 0$

Spring 2: $u_2 - u_1 = 1.54590$

Spring 3: $u_3 - u_2 = -0.6763$

Spring 4: $u_3 - u_2 = -0.6763$

Spring 5: $u_4 - u_2 = -1.5459$

Spring 6: $u_4 - u_3 = -0.8696$

2.3.4 Boundary Conditions with Known Values

Up to now we have considered problems that have known applied forces, and no known values of deflection. Now we will consider the procedure for applying the boundary conditions where, deflections on some nodes are known. Solutions of these problems are found by going through some additional steps. As discussed earlier, after obtaining the

structural global matrix equation, deflections are found by solving the equation by applying a numerical scheme in a computer solution. However, when there are known nodal values and unknown nodal forces, the method is not directly applicable. In this situation, the structural equation is first modified by incorporating all boundary conditions and then the final matrix equation is solved by a computer using a numerical method, as mentioned earlier. The following procedure traces the necessary steps for solving problems that involve known nodal values.

2.3.5 Procedure for incorporating the known Nodal Values in the Final Structural Equation

There are two methods that are frequently used for applying boundary conditions to a structural matrix equation. In one method, the matrices are partitioned into two parts with known and unknown terms. In the second method, the known nodal values are applied directly in the structural matrix. Both methods can be used with equal effectiveness. The first method will not be discussed here. Details of the second method follow.

Consider the following linear equations,

$$k_{11}u_1 + k_{12}u_2 + k_{13}u_3 + k_{14}u_4 = F_1 \quad (2.2)$$
$$k_{21}u_1 + k_{22}u_2 + k_{23}u_3 + k_{24}u_4 = F_2 \quad (2.3)$$
$$k_{31}u_1 + k_{32}u_2 + k_{33}u_3 + k_{34}u_4 = F_3 \quad (2.4)$$
$$k_{41}u_1 + k_{42}u_2 + k_{43}u_3 + k_{44}u_4 = F_4 \quad (2.5)$$

These linear algebraic equations can be written in matrix form as follows.

$$\begin{pmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{pmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix}$$

Let the known nodal value at node 2 be $u_2 = U_2$ (a constant), then by the linear spring equation

$$F_2 = k_{22} U_2$$

Therefore, equation (2.2 – 2.5)) above can be reduced to $k_{22}u_2 = k_{22}U_2 = F_2$ and the matrix with this boundary condition can be written as

$$\begin{pmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ 0 & k_{22} & 0 & 0 \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{pmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix}$$

Now, equations 2.2, 2.4, 2.5 also contain the u_2 term and therefore these equations must also be modified. We can modify equation 1 by transferring the term $k_{12}u_2$ to the right hand side and replacing u_2 by U_2. The modified equation can be written as

$$K_{11}u_1 + 0 + k_{13}u_3 + k_{14}u_4 = F_1 - k_{12}U_2$$

Similarly, equations 3 and 4 can be written as

$$K_{31}u_1 + 0 + k_{33}u_3 + k_{34}u_4 = F_3 - k_{32}U_2$$
$$K_{41}u_1 + 0 + k_{43}u_3 + k_{44}u_4 = F_4 - k_{42}U_2$$

The final matrix equation is

$$\left(\begin{array}{cc:cc} k_{11} & 0 & k_{13} & k_{14} \\ \hdashline 0 & k_{22} & 0 & 0 \\ \hdashline k_{31} & 0 & k_{33} & k_{34} \\ k_{41} & 0 & k_{43} & k_{44} \end{array}\right)\left\{\begin{array}{c} u_1 \\ u_2 \\ u_3 \\ u_4 \end{array}\right\} = \left\{\begin{array}{l} F_1 - k_1U_2 \\ k_{22}U_2 \\ F_3 - k_{32}U_2 \\ F_4 - k_{42}U_2 \end{array}\right\}$$

The dotted line indicates changes made in the enclosed terms. The final matrix remains symmetric and has the same size. The boundary conditions for forces can now be incorporated and a numerical solution scheme can be used to solve this equation. This procedure is summarized in the following simple, step-by-step approach.

Given the known boundary conditions at node 2: $u_i = u_2 = U_2$, follow these steps to incorporate the known nodal values. Note that, here, i = 2 and j = 1,2,3,4.

Step 1: Set all terms in row 2 to zero, except the term in column 2 (kij = 0, $k_{ii} = k_{22} \neq 0$)

Step 2: Replace F_2 with the term $k_{22}U_2$ ($F_i = k_{ii}u_i$)

Step 3: Subtract the value k_{i2} U2 from all the forces, except F_2 (subtract k_{ji} from the existing values of f_j), where i = 1, 3, and 4

Step 4: Set all the elements in column 2 to zero, except, row2 (all $k_{ji} = 0$, k_{ii} # 0)

The above procedure now will be applied in the following example problem.

Example 2.5

In example problem 2.4 replace the force F by a nodal deflection of 1.5 mm on node 2 and rework the problem.

Solution

Rewriting the final structural matrix equation in example 2.4, we have

$$\begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix} = \begin{pmatrix} 25 & -15 & 0 & -10 \\ -15 & 90 & -45 & -30 \\ 0 & -45 & 80 & -35 \\ -10 & -30 & -35 & 75 \end{pmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}$$

Boundary condition are: $u_1 = u_4 = 0$, and $u_2 = U_2 = 1.5$mm. Applying the 4 steps described above in sequence,

Step 1: Set all terms in row 2 to zero, except the term in column 2 ($kij = 0$, $k_{ii} = k_{22} \neq 0$)

$$\begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix} = \begin{pmatrix} 25 & -15 & 0 & -10 \\ 0 & 90 & 0 & 0 \\ 0 & -45 & 80 & -35 \\ -10 & -30 & -35 & 75 \end{pmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}$$

Step 2: Replace F_2 with the term $k_{22} U_2 = (90)(1.5) = 135$, ($F_i = k_{ii} u_i$)

$$\begin{Bmatrix} F_1 \\ 135 \\ F_3 \\ F_4 \end{Bmatrix} = \begin{pmatrix} 25 & -15 & 0 & -10 \\ 0 & 90 & 0 & 0 \\ 0 & -45 & 80 & -35 \\ -10 & -30 & -35 & 75 \end{pmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}$$

Step 3: Subtract the value $k_{22} U_2$ from all the forces, except F_2 (subtract k_{ji} from the existing values of f_j)

$F_1 \rightarrow F_1 - (15)(1.5) = 22.5$ Row 1: $k_{j2} = k_{12} =$ -15
$F_3 \rightarrow F_3 - (-45)(1.5) = 67.5$ Row 2: $k_{j2} = k_{32} =$ -45
$F_4 \rightarrow F_4 - (-30)(1.5) = 45$ Row 2: $k_{j2} = k_{42} =$ - 30

Note: $F_1 = F_3 = F_4 = 0$.

The new force equation now is,

$$\begin{Bmatrix} 22.5 \\ 135 \\ 67.5 \\ 45 \end{Bmatrix}$$

Step 4: Set all the elements in column 2 to zero, except, row2 (all $k_{ji} = 0$, $k_{ii} \neq 0$)

Or, $k_{12} = k_{32} = k_{42} = 0$, and the new equation is,

$$\begin{Bmatrix} 22.5 \\ 135 \\ 67.5 \\ 45 \end{Bmatrix} = \begin{pmatrix} 25 & 0 & 0 & -10 \\ 0 & 90 & 0 & 0 \\ 0 & 0 & 80 & -35 \\ -10 & 0 & -35 & 75 \end{pmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}$$

This is the final equation after the nodal value $u_2 = 1.5$ mm is incorporated into the structural equation.

The same procedure can be followed for the boundary conditions $u_1 = u_4 = 0$. It can be stated that for zero nodal values, the procedure will always lead to elimination of rows and columns corresponding to these nodes, that is, the first and fourth rows as well as columns will drop out. The reader is encouraged to verify this statement.

Thus, the final equation is,

$$\begin{pmatrix} 90 & 0 \\ 0 & 80 \end{pmatrix} \begin{Bmatrix} u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} 135 \\ 67.5 \end{Bmatrix}$$

Solving for u_2 and u_3, we get

$$\begin{Bmatrix} u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} 1.5 \\ 0.8437 \end{Bmatrix}$$

Spring deflection is:

Spring 1:	$u_2 - u_1 = 1.500$
Spring 2:	$u_3 - u_1 = 0.8437$
Spring 3:	$u_3 - u_2 = -0.6563$
Spring 4:	$u_3 - u_2 = -0.6563$
Spring 5:	$u_4 - u_2 = -1.500$
Spring 6:	$u_4 - u_3 = -1.6875$

2.3.6 Structures That can be Modeled Using a Spring Elements

As mentioned earlier, almost all engineering materials are similar to a linear spring, satisfying the relation F = ku. Therefore, any structure that deflects only along its axial direction (with one degree of freedom) can be modeled as a spring element. The following example illustrates this concept.

Example 2.6

A circular concrete beam structure is loaded as shown. Find the deflection of points at 8", 16", and the end of the beam. E = 4 x 106 psi

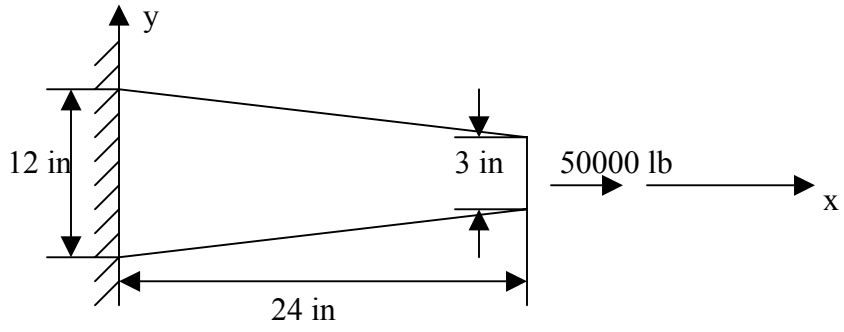


Figure 2.7

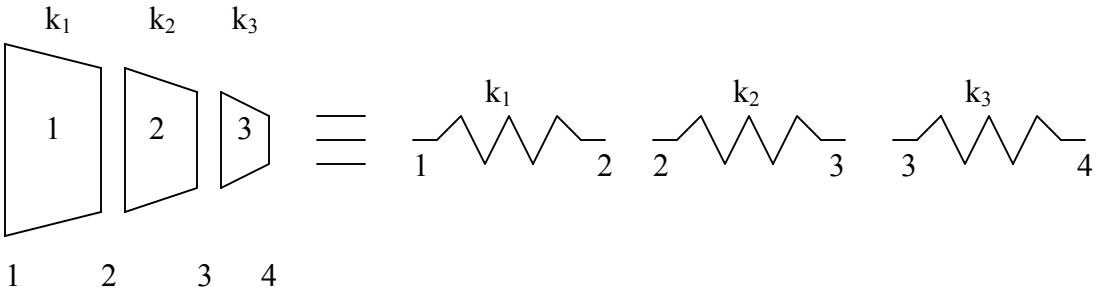
Solution

The beam structure looks very different from a spring. However, its behavior is very similar. Deflection occurs along the x-axis only. The only significant difference between the beam and a spring is that the beam has a variable cross-sectional area. An exact solution can be found if the beam is divided into an infinite number of elements, then, each element can be considered as a constant cross-section spring element, obeying the relation F = ku, where k is the stiffness constant of a beam element and is given by k = AE/L.

In order to keep size of the matrices small (for hand- calculations), let us divide the beam into only three elements. For engineering accuracy, the answer obtained will be in an acceptable range. If needed, accuracy can be improved by increasing the number of elements.

As mentioned earlier in this chapter, spring, truss, and beam elements are line-elements and the shape of the cross section of an element is irrelevant. Only the cross-sectional area is needed (also, moment of inertia for a beam element undergoing a bending load need to be defined). The beam elements and their computer models are shown in figure 2.8.

Here, the question of which cross-sectional area to be used for each beam section arises. A good approximation would be to take the diameter of the mid-section and use that to approximate the area of the element.



Beam sections *Equivalent spring elements*

Figure 2.8

Cross-sectional area

The average diameters are: $d_1 = 10.5$ in., $d_2 = 7.5$ in., $d_3 = 4.5$. (diameters are taken at the mid sections and the values are found from the height and length ratio of the triangles shown in figure 2.10), which is given as

$12/L = 3/(L-24)$, $L = 32$

Average areas are:

$A_1 = 86.59$ in2 $A_2 = 56.25$ in2 $A_3 = 15.9$ in2

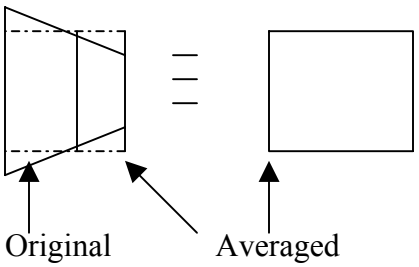


Figure 2.9

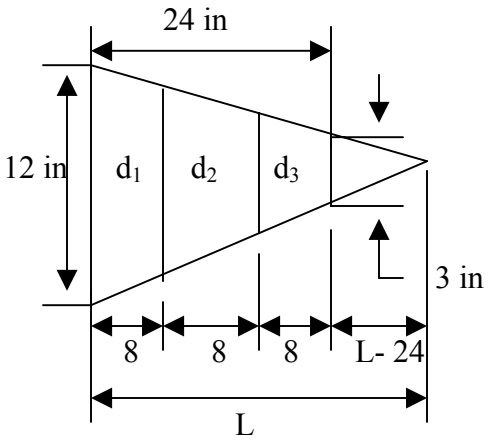


Figure 2.10

Stiffness

$k_1 = A_1 E/L_1 = (86.59)(4 \times 10^6/8) = 4.3295 \times 10^7$ lb./in., similarly,

$k_2 = A_2 E/L_2 = 2.8125 \times 10^7$ lb./in.
$k_3 = A_3 E/L_3 = 7.95 \times 10^6$ lb./in.

Element Stiffness Equations

$$[K^{(1)}] = 43.295 \times 10^7 \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

Similarly,

$$[K^{(2)}] = 28.125 \times 10^6 \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

$$[K^{(3)}] = 7.9500 \times 10^6 \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

Global stiffness matrix is

$$[K_g] = \begin{pmatrix} 43.295 & -43.295 & 0 & 0 \\ -43.295 & 43.295+28.125 & -28.125 & 0 \\ 0 & -28.125 & 28.125+7.95 & -7.95 \\ 0 & 0 & -7.95 & 7.95 \end{pmatrix} \times 10^6$$

Now the global structural equations can be written as,

$$10^6 \times \begin{pmatrix} 43.295 & -43.295 & 0 & 0 \\ -43.295 & 71.42 & -28.125 & 0 \\ 0 & -28.125 & 36.075 & -7.95 \\ 0 & 0 & -7.95 & 7.95 \end{pmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix} = \begin{Bmatrix} F1 \\ F2 \\ F3 \\ F4 \end{Bmatrix}$$

Applying the boundary conditions: $u_1 = 0$, and $F_1 = F_2 = F_3 = 0$, $F_4 = 5000$ lb., results in the reduced matrix,

$$10^6 \times \begin{pmatrix} 71.42 & -28.125 & 0 \\ -28.125 & 36.075 & -7.95 \\ 0 & -7.95 & 7.95 \end{pmatrix} \begin{Bmatrix} u_2 \\ u_3 \\ u_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 5000 \end{Bmatrix}$$

Solving we get,

$$\begin{Bmatrix} u_2 \\ u_3 \\ u_4 \end{Bmatrix} = \begin{Bmatrix} 0.0012 \\ 0.0029 \\ 0.0092 \end{Bmatrix} \text{ in.}$$

CHAPTER 3

Truss Element

3.1 Introduction

The single most important concept in understanding FEA, is the basic understanding of various finite elements that we employ in an analysis. Elements are used for representing a real engineering structure, and therefore, their selection must be a true representation of geometry and mechanical properties of the structure. Any deviation from either the geometry or the mechanical properties would yield erroneous results.

The elements used in commercial codes can be classified in two basic categories:

1. **Discrete elements**: These elements have a well defined deflection equation that can be found in an engineering handbook, such as, Truss and Beam/Frame elements. The geometry of these elements is simple, and in general, mesh refinement does not give better results. Discrete elements have a very limited application; bulk of the FEA application relies on the Continuous-structure elements.

2. **Continuous-structure Elements:** Continuous-structure elements do not have a well define deflection or interpolation function, it is developed and approximated by using the theory of elasticity. In general, a continuous-structure element can have any geometric shape, unlike a truss or beam element. The geometry is represented by either a 2-D or a 3-D solid element – the continuous- structure elements. Since elements in this category can have any shape, it is very effective in calculation of stresses at a sharp curve or geometry, i.e., evaluation of stress concentrations. Since discrete elements cannot be used for this purpose, continuous structural elements are extremely useful for finding stress concentration points in structures.

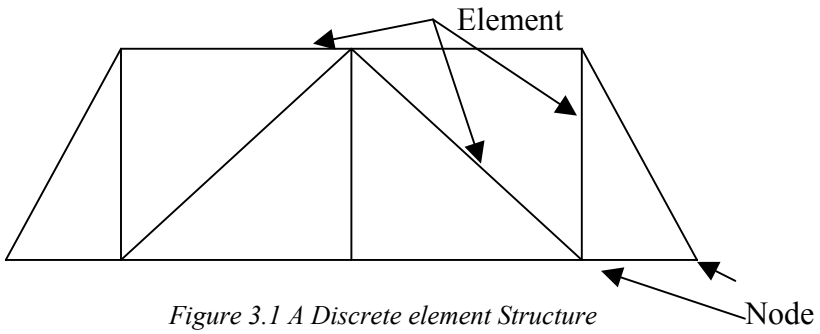


Figure 3.1 A Discrete element Structure

As explained earlier, for analyzing an engineering structure, we divide the structure into small sections and represent them by appropriate elements. Nodes always define geometry of the structure and elements are generated when the applicable nodes are connected. Results are always obtained for node points – and not for elements - which are then interpolated to provide values for the corresponding elements.

For a static structure, all nodes must satisfy the equilibrium conditions and the continuity of displacement, translation and rotation.

In the following sections, we will get familiar with characteristics of the basic finite elements.

3.2 Structures & Elements

Most 3-D structures can be analyzed using 2-D elements, which require relatively less computing time than the 3-D solid elements. Therefore, in FEA, 2-D elements are the most widely used elements. However, there are cases where we must use 3-D solid elements. In general, elements used in FEA can be classified as:

- Trusses
- Beams
- Plates
- Shells
- Plane solids
- Axisymmetric solids
- 3-D solids

Since Truss element is a very simple and discrete element, let us look at its properties and application first.

3.3 Truss Elements

The characteristics of a truss element can be summarized as follows:

- Truss is a slender member (length is much larger than the cross-section).
- It is a two-force member i.e. it can only support an axial load and cannot support a bending load. Members are joined by pins (no translation at the constrained node, but free to rotate in any direction).
- The cross-sectional dimensions and elastic properties of each member are constant along its length.
- The element may interconnect in a 2-D or 3-D configuration in space.
- The element is mechanically equivalent to a spring, since it has no stiffness against applied loads except those acting along the axis of the member.

- However, unlike a spring element, a truss element can be oriented in any direction in a plane, and the same element is capable of tension as well as compression.

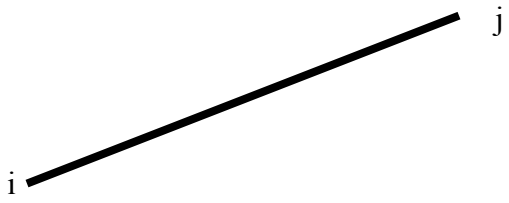


Figure 3.2 A Truss Element

3.3.1 Stress – Strain relation:

As stated earlier, all deflections in FEA are evaluated at the nodes. The stress and strain in an element is calculated by interpolation of deflection values shared by nodes of the element. Since the deflection equation of the element is clearly defined, calculation of stress and strain is rather simple matter. When a load F is applied on a truss member, the strain at a point is found by the following relationship.

$$\varepsilon = \frac{du}{dx}$$

or, $\varepsilon = \delta L/L$

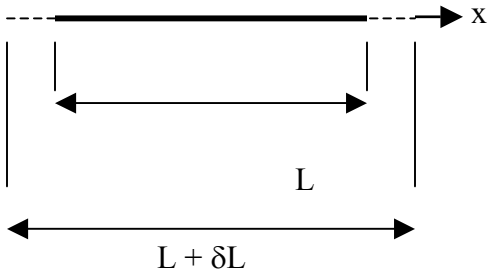


Figure 3.3 Truss member in Tension

where, ε = strain at a point
u = axial displacement of any point along the length L

By hook's law,

$$\sigma = E\varepsilon$$

Where, E = young's modulus or modulus of elasticity.

From the above relationship, and the relation,

$$F = A\sigma$$

the deflection, δL, can be found as

$$\delta L = FL/AE \qquad (3.1)$$

Where, F = Applied load
A = Cross-section area
L = Length of the element

3.3.2 Treatment of Loads in FEA

For a truss element, loads can be applied on a node only. If loads are distributed on a structure, they must be converted to the equivalent loads that can be applied at nodes. Loads can be applied in any direction at the node, however, the element can resist only the axial component, and the component perpendicular to the axis, merely causes free rotation at the joint.

3.3.3 Finite Element Equation of a Truss Structure

In this section, we will derive the finite element equation of a truss structure. The procedure presented here is the basis for all FEA analyses formulations, wherever h-element are used.

Analogues to the previous chapter, we will use the direct or equilibrium method for generating the finite element equations. Assembly procedure for obtaining the global matrix will remain the same.

In FEA, when we find deflections at nodes, the deflections are measured with respect to a global coordinate system, which is a fixed frame of reference. Displacements of individual nodes with respect to a fixed coordinate system are desirable in order to see the overall deformed structural shape. However, these deflection values are not convenient in the calculation of stress and strain in an element. Global coordinate system is good for predicting the overall deflections in the structure, but not for finding deflection, strain, and stress in an element. For this, it's much easier to use a local coordinate system. We will derive a general equation, which relates local and global coordinates.

In Figure 3.4, the global coordinates x-y can give us the overall deflections measured with respect to the fixed coordinate system. These deflections are useful for finding the final shape or clearance with the surroundings of the structure. However, if we wish to find the strain in some element, say, member 2-7 in figure 3.4, it will be easier if we know the deflections of node 2 and 3, in the y' direction. Thus, calculation of strain value is much easier when the local deflection values are known, and will be time-

consuming if we have to work with the x and y values of deflection at these nodes. Therefore, we need to establish a trigonometric relationship between the local and global coordinate systems. In Figure 3.4, xy coordinates are global, where as, x'y' are local coordinates for element 4-7

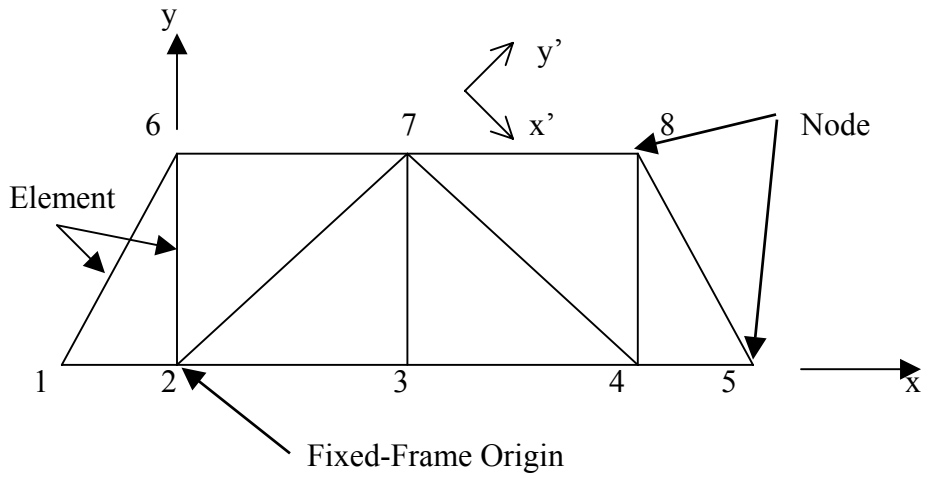


Figure 3.4. Local and Global Coordinates

3.3.4 Relationship Between Local and Global Deflections

Let us consider the truss member, shown in Figure 3.5. The element is inclined at an angle θ, in a counter clockwise direction. The local deflections are δ_1 and δ_2. The global deflections are: u_1, u_2, u_3, and u_4. We wish to establish a relationship between these deflections in terms of the given trigonometric relations.

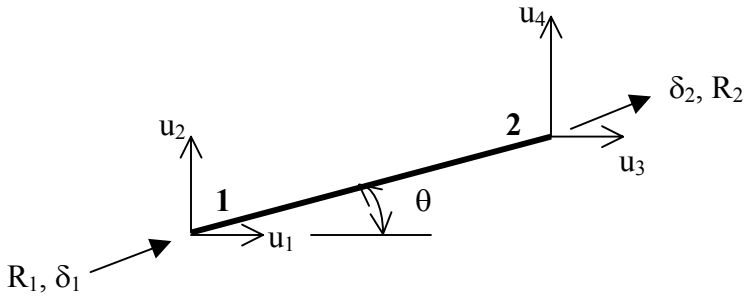


Figure 3.5 Local and Global Deflections

By trigonometric relations, we have,

$$\delta_1 = u_1 \cos\theta + u_2 \sin\theta = c\, u_1 + s\, u_2$$

$$\delta_2 = u_3 \cos\theta + u_4 \sin\theta = c\, u_3 + s\, u_4$$

where, $\cos\theta = c$, and $\sin\theta = s$

Writing the above equations in a matrix form, we get,

$$\begin{pmatrix} \delta_1 \\ \delta_1 \end{pmatrix} = \begin{pmatrix} c & s & 0 & 0 \\ 0 & 0 & c & s \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} \qquad (3.2)$$

Or, in short form, $\{\delta\} = [T]\{u\}$

Where T is called Transformation matrix.

Along with equation (3.2), we also need an equation that relates the local and global forces.

3.3.5 Relationship Between Local and Global Forces

By using trigonometric relations similar to the previous section, we can derive the desired relationship between local and global forces. However, it will be easier to use the work-energy concept for this purpose. The forces in local coordinates are: R_1 and R_2, and in global coordinates: f_1, f_2, f_3, and f_4, see Figure 3.6 for their directions.

Since work done is independent of a coordinate system, it will be the same whether we use a local coordinate system or a global one. Thus, work done in the two systems is equal and given as,

$W = \delta^T R = u^T f$, or in an expanded form,

$$W = \begin{pmatrix} \delta_1 & \delta_2 \end{pmatrix} \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = \begin{pmatrix} u_1 & u_1 & u_1 & u_1 \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix}$$

$$= \{\delta\}^T \{R\} \qquad = \{u\}^T \{f\}$$

Substituting $\{\delta\} = [T]\{u\}$ in the above equation, we get,

$$[[T]\{u\}]^T \{R\} = \{u\}^T \{f\}, \text{ or}$$

$$\{u\}^T [T]^T \{R\} = \{u\}^T \{f\}, \text{ dividing by } \{u\}^T \text{ on both sides, we get,}$$

$$[T]^T \{R\} = \{f\} \qquad (3.3)$$

Equation (3.3) can be used to convert local forces into global forces and vice versa.

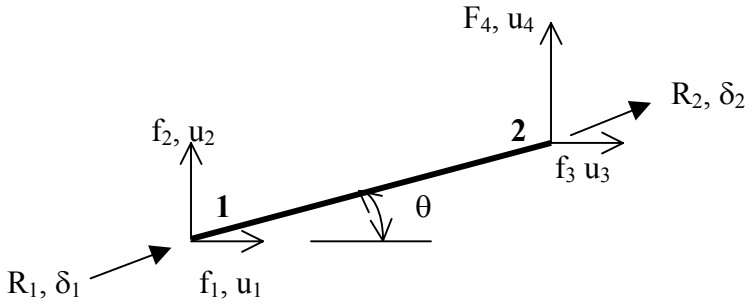


Figure 3.6 Local and Global Forces

3.3.6 Finite Element Equation in Local Coordinate System

Now we will derive the finite element equation in local coordinate system. This equation will be converted to global coordinate system, which can be used to generate a global structural equation for the given structure. Note that, we can not use the element equations in their local coordinate form, they must be converted to a common coordinate system, the global coordinate system.

Consider the element shown below, with nodes 1 and 2, spring constant k, deflections δ_1, and δ_2, and forces R_1 and R_2. As established earlier, the finite element equation in local coordinates is given as,

$$\begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = \begin{pmatrix} k & -k \\ -k & k \end{pmatrix} \begin{pmatrix} \delta_1 \\ \delta_1 \end{pmatrix}$$

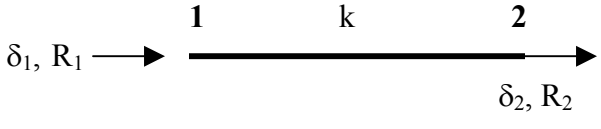


Figure 3.7 A Truss Element

Recall that, for a truss element, k = AE/L

Let k_e = stiffness matrix in local coordinates, then,

$$k_e = \begin{pmatrix} AE/L & -AE/L \\ -AE/L & AE/L \end{pmatrix}$$ *Stiffness matrix in local coordinates*

3.3.7 Finite Element Equation in Global Coordinates

Using the relationships between local and global deflections and forces, we can convert an element equation from a local coordinate system to a global system.

Let k_g = Stiffness matrix in global coordinates.

In local system, the equation is: $R = [k_e]\{\delta\}$ (A)

We want a similar equation, but in global coordinates. We can replace the local force R with the global force f derived earlier and given by the relation:

$$\{f\} = [T^T]\{R\}$$

Replacing R by using equation (A), we get,

$$\{f\} = [T^T]\,[[k_e]\{\delta\}],$$

and δ can be replaced by u, using the relation $\delta = [T]\{u\}$, therefore,

$$\{f\} = [T^T]\,[k_e]\,[T]\{u\}$$

$$\{f\} = [k_g]\ \{u\}$$

Where, $[k_g] = [T^T]\,[k_e]\,[T]$

Substituting the values of $[T]^T$, $[T]$, and $[k_e]$, we get,

$$[k_g] = \begin{pmatrix} c & 0 \\ s & 0 \\ 0 & c \\ 0 & s \end{pmatrix} \begin{pmatrix} AE/L & -AE/L \\ -AE/L & AE/L \end{pmatrix} \begin{pmatrix} c & s & 0 & 0 \\ 0 & 0 & c & s \end{pmatrix}$$

Simplifying the above equation, we get,

$$[k_g] = \begin{pmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{pmatrix} (AE/L)$$

This is the global stiffness matrix of a truss element. This matrix has several noteworthy characteristics:

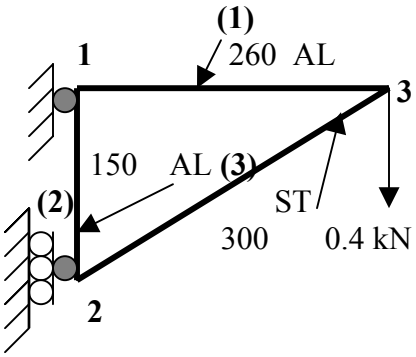
- The matrix is symmetric
- Since there are 4 unknown deflections (DOF), the matrix size is a 4 x 4.
- The matrix represents the stiffness of a single element.
- The terms c and s represent the sine and cosine values of the orientation of element with the horizontal plane, rotated in a counter clockwise direction (positive direction).

The following example will illustrate its application.

Examples

For the truss structure shown:

1. Find displacements of joints 2 and 3
2. Find stress, strain, & internal forces in each member.



$A_{AL} = 200\ mm^2$, $A_{ST} = 100\ mm^2$

All other dimensions are in mm.

Solution

Let the following node pairs form the elements:

Element	**Node Pair**
(1)	1-3
(2)	1-2
(3)	2-3

Using Shigley's Machine Design book for yield strength values, we have,

$S_{y\,(AL)} = 0.0375 kN/mm^2$ (375 Mpa)

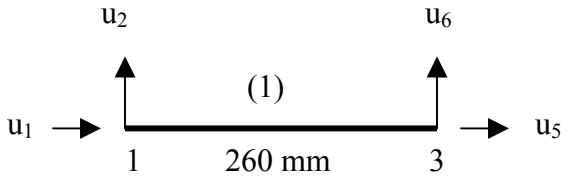
$S_{y\,(ST)} = 0.0586 kN/mm^2$ (586 Mpa)

$E_{(AL)} = 69 kN/mm^2$, $E_{(ST)} = 207 kN/mm^2$

$A^{(1)} = A^{(2)} = 200 mm^2$, $A^{(3)} = 100 mm^2$

Find the stiffness matrix for each element

Element (1)



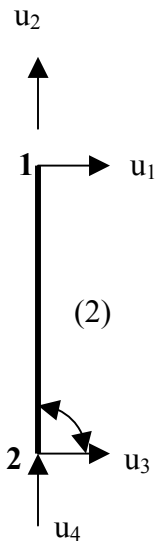
$L^{(1)} = 260$ mm,
$E^{(1)} = 69 kN/mm^2$,
$A^{(1)} = 200 mm^2$

$\theta = 0$
$c = \cos\theta = 1$, $c^2 = 1$
$s = \sin\theta = 0$, $s^2 = 0$
$cs = 0$

$$\frac{EA}{L} = 69 kN/mm^2 \times 200 mm^2 \times \frac{1}{260 mm} = 53.1 kN/mm$$

$$[K_g]^{(1)} = \begin{bmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{bmatrix} \frac{AE}{L}$$

$$[K_g]^{(1)} = \begin{matrix} 1 \\ 2 \\ 5 \\ 6 \end{matrix} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} (53.1) \frac{kN}{mm}$$



Element 2

$\theta = 90^0$
$c = \cos 90^0 = 0,\ c^2 = 0$

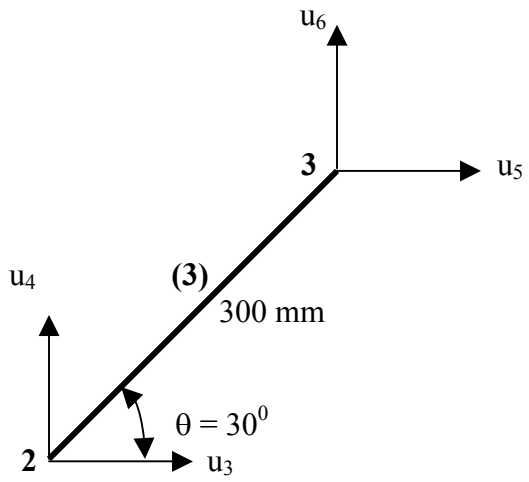
$s = \sin 90^0 = \cos 0^0 = 1,\ s^2 = 1$

$cs = 0$

$EA/L = 69 \times 200 \times (1/150) = 92$ kN/mm

$$[k_g]^{(2)} = (92)\begin{array}{c} \\ 3 \\ 4 \\ 1 \\ 2 \end{array}\begin{array}{c} \begin{array}{cccc} \textit{3} & \textit{4} & \textit{1} & \textit{2} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix} \end{array}$$

Element 3



$\theta = 30^0$
$c = \cos 30^0 = 0.866,\ c^2 = 0.75$

$s = \cos 60^0 = .5,\ s^2 = 0.25$

$cs = 0.433$

$EA/L = 207 \times 100 \times (1/300) = 69$ kN/mm

$$[k_g]^{(3)} = \begin{array}{c} \\ 3 \\ 4 \\ 5 \\ 6 \end{array}\begin{array}{c} \begin{array}{cccc} \textit{3} & \textit{4} & \textit{5} & \textit{6} \end{array} \\ \begin{pmatrix} .75 & .433 & -.75 & -.433 \\ -.433 & .25 & -.433 & -.25 \\ -.75 & -.433 & .75 & .433 \\ -.433 & -.25 & .433 & .25 \end{pmatrix} \end{array} \quad (69)$$

Assembling the stiffness matrices

Since there are 6 deflections (or DOF), u_1 through u_6, the matrix is 6 x 6. Now, we will place the individual matrix element from the element stiffness matrices into the global matrix according to their position of row and column members.

Element [1]

	1	2	3	4	5	6
1	53.1				−53.1	
2						
3						
4						
5	−53.1				53.1	
6						

The blank spaces in the matrix have a zero value.

Element [2]

	1	2	3	4	5	6
1						
2		92		-92		
3						
4		-92		92		
5						
6						

Element [3]

	1	2	3	4	5	6
1						
2						
3			51.7	29.9	-51.7	-29.9
4			29.9	17.2	-29.9	-17.2
5			-51.7	-29.9	51.7	29.9
6			-29.9	-17.2	29.9	17.2

Assembling all the terms for elements [1] , [2] and [3], we get the complete matrix equation of the structure.

$$\begin{matrix}1\\2\\3\\4\\5\\6\end{matrix}\begin{bmatrix} 53.1 & 0 & 0 & 0 & -53.1 & 0 \\ 0 & 92 & 0 & -92 & 0 & 0 \\ 0 & 0 & 51.7 & 29.9 & -51.7 & -29.9 \\ 0 & -92 & 29.9 & 109.2 & -29.9 & -17.2 \\ -53.1 & 0 & -51.7 & -29.9 & 104.8 & 29.9 \\ 0 & 0 & -29.9 & -17.2 & 29.9 & 17.2 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{Bmatrix} = \begin{Bmatrix} 0(F_1) \\ 0(F_2) \\ 0(F_3) \\ 0(F_4) \\ 0(F_5) \\ -0.4(F_6) \end{Bmatrix}$$

Boundary conditions

Node 1 is fixed in both x and y directions, where as, node 2 is fixed in x-direction only and free to move in the y-direction. Thus,

$u_1 = u_2 = u_3 = 0$.

Therefore, all the columns and rows containing these elements should be set to zero. The reduced matrix is:

$$\begin{bmatrix} 109.2 & -29.9 & -17.2 \\ -29.9 & 104.8 & 29.9 \\ -17.2 & 29.9 & 17.2 \end{bmatrix} \begin{Bmatrix} u_4 \\ u_5 \\ u_6 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ -0.4 \end{Bmatrix}$$

Writing the matrix equation into algebraic linear equations, we get,

$$\begin{aligned} 29.9u_4 - 29.9u_5 - 17.2u_6 &= 0 \\ -29.9u_4 + 104u_5 + 29.9u_6 &= 0 \\ -17.2u_4 + 29.9u_5 + 17.2u_6 &= -0.4 \end{aligned}$$

solving, we get $u_4 = -0.0043$
$u_5 = 0.0131$
$u_6 = -0.0502$

Sress, Strain and deflections

Element (1)

Note that u_1, u2, u3, etc. are not coordinates, they are actual displacements.

$$\Delta L = u_5 = 0.0131$$

$$\in = \frac{\Delta L}{L} = \frac{0.0131}{260} = 5.02 \times 10^{-5} \frac{mm}{mm}$$

$$\sigma = E \in = 69 \times 5.02 \times 10^{-5} = 0.00347 \frac{kN}{mm^2}$$

$$Reaction, R = \sigma A = 0.00347 \times 200 = 0.693 kN$$

Element (2)

$$\Delta L = u_4 = 0.0043$$

$$\in = \frac{\Delta L}{L} = \frac{0.0043}{150} = 2.87 \times 10^{-5}$$

$$\sigma = E \in = 69 \times 2.87 \times 10^{-5} = 1.9803 \frac{kN}{mm^2}$$

$$R = \sigma A = (1.9803 \times 10^{-3})(200) = 0.396 kN$$

Element (3)

Since element (3) is at an angle 30^0, the change in the length is found by adding the displacement components of nodes 2 and 3 along the element (at 30^0). Thus,

$$\Delta L = u_5 \cos 30^0 + u_6 \sin 30^0 - u_4 \cos 30^0$$

$$\Delta L = 0.0131 \cos 30^0 - 0.0502 \sin 30^0 + 0.0043 \cos 30^0$$

$$= -0.0116 \quad (element \quad is \quad compressed)$$

$$\in = \frac{\Delta L}{L} = \frac{-0.0116}{300} = -3.87 \times 10^{-5}$$

$$\sigma = E \in = 207 \times -3.87 \times 10^{-5} = -0.0080 \frac{kN}{mm^2}$$

$$Axial \quad force, R = \sigma A = -0.0080 \times 100 = -0.800 kN$$

Factor of Safety

Factor of safety '*n*' is the ratio of yield strength to the actual stress found in the part.

$$Element(1) \quad n = \frac{S_y}{\sigma} = \frac{0.0375}{0.00347} = 10.8$$

$$Element(2) \quad n = \frac{S_y}{\sigma} = \frac{0.0375}{0.00198} = 18.9$$

$$Element(3) \quad n = \frac{S_y}{\sigma} = \frac{0.0586}{0.0080} = 7.325$$

The lowest factor of safety is found in element (3), and therefore, the steel bar is the most likely to fail before the aluminum bar does.

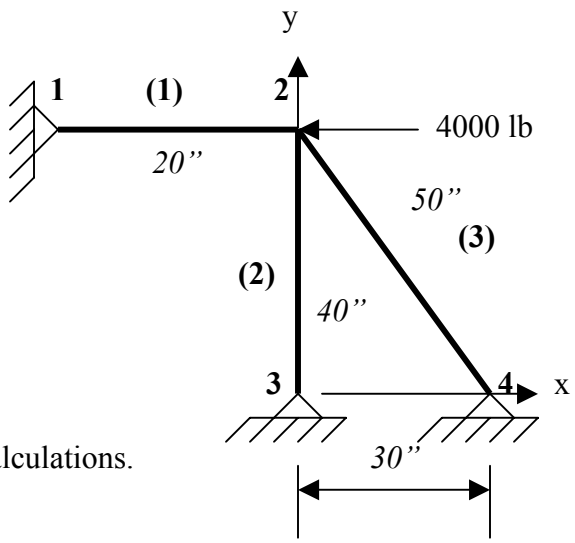
Final Notes

- The example presented gives an insight into how the element analysis works. The example problem is too simple to need a computer based solution; however, it gives the insight into the actual FEA procedure. In a commercial FEA package, solution of a typical problem generates a very large stiffness matrix, which will require a computer assisted solution.

- In an FEA software, the node and element numbers will have variable subscripts so that they will be compatible with a computer-solution

- Direct or equilibrium method is the earliest FEA method.

Example 2



Given:
Elements 1 and 2: Aluminum
Element 3: steel
$A^{(1)} = 1.5in^2$
$A^{(2)} = 1.0in^2$
$A^{(3)} = 1.0in^2$

Required:
Find stresses and displacements using hand calculations.

Solution

Calculate the stiffness constants:

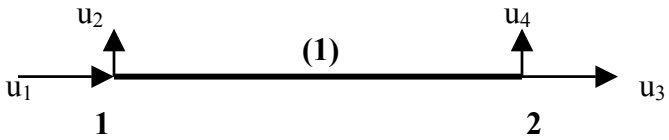
$$K_1 = \frac{AE}{L} = \frac{1.5 \times 10 \times 10^6}{20} = 7.5 \times 10^5 \frac{lb}{in}$$

$$K_2 = \frac{AE}{L} = \frac{1 \times 10 \times 10^6}{40} = 2.5 \times 10^5 \frac{lb}{in}$$

$$K_3 = \frac{AE}{L} = \frac{30 \times 10 \times 10^6}{50} = 6.0 \times 10^5 \frac{lb}{in}$$

Calculate the Element matrix equations.

Element (1)



Denoting the Spring constant for element (1) by k_1, and the stiffness matrix by $K^{(1)}$, the stiffness matrix in global coordinates is given as,

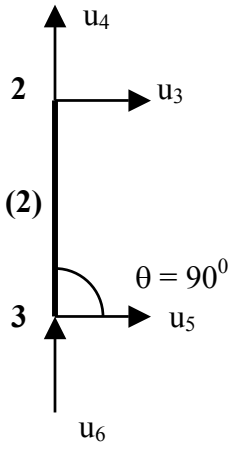
$$[K_g]^{(1)} = K_1 \begin{pmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{pmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix}$$

For element (1), $\theta = 0^{0,}$ therefore

$c = 1$, $c^2 = 1$

$s = 0$, $s^2 = 0$, and $cs = 0$

$$[k^{(1)}] = k_1 \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix}$$



Element (2)

For this element, $\theta = 90^0$, Therefore,

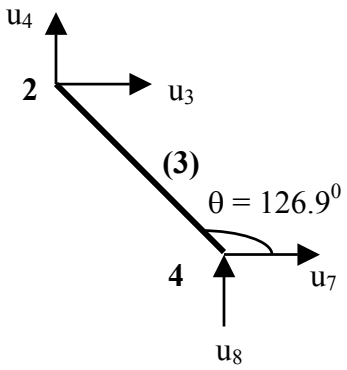
$c = \cos\theta = 0$, $c^2 = 0$
$s = \sin\theta = 1$, $s^2 = 1$
$cs = 0$

The stiffness matrix is,

$$[k_g]^{(2)} = k_2 \begin{pmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{pmatrix} \begin{matrix} 5 \\ 6 \\ 3 \\ 4 \end{matrix}$$

$$[k_g]^{(2)} = k_2 \begin{pmatrix} & \textit{5} & \textit{6} & \textit{3} & \textit{4} & \\ & 0 & 0 & 0 & 0 & \textit{5} \\ & 0 & 1 & 0 & -1 & \textit{6} \\ & 0 & 0 & 0 & 0 & \textit{3} \\ & 0 & -1 & 0 & 1 & \textit{4} \end{pmatrix}$$

Element 3



For element (3), $\theta = 126.9^0$.

$c = \cos(126.9^0) = -0.6$, $c^2 = .36$

$s = \sin(126.9^0$, $s^2 = .64$

$cs = -0.48$

$$[k_g]^{(3)} = k_3 \begin{pmatrix} & \textit{7} & \textit{8} & \textit{3} & \textit{4} & \\ & .36 & -.48 & -.36 & .48 & \textit{7} \\ & -.48 & .64 & .48 & -.64 & \textit{8} \\ & -.36 & .48 & .36 & -.48 & \textit{3} \\ & .48 & -.64 & -.48 & .64 & \textit{4} \end{pmatrix}$$

Assembling the global Matrix

Following the procedure for assembly described earlier, the assembled matrix is,

$$[K_g] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{bmatrix} K_1 & 0 & -K_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -K_1 & 0 & K_1 + .36K_3 & -.48K_3 & 0 & 0 & -.36K_3 & .48K_3 \\ 0 & 0 & .48K_3 & K_2 + .64K_3 & 0 & -K_2 & .48K_3 & -.64K_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -K_2 & 0 & K_2 & 0 & 0 \\ 0 & 0 & -.36K_3 & .48K_3 & 0 & 0 & .36K_3 & -.48K_3 \\ 0 & 0 & .48K_3 & -.64K_3 & 0 & 0 & -.48K_3 & .64K_3 \end{bmatrix}$$

The boundary conditions are:

$u_1 = u_2 = u_5 = u_6 = u_7 = u_8 = 0$

We will suppress the corresponding rows and columns. The reduced matrix is a 2 x2, given below,

$$[K_g] = \begin{matrix} 3 \\ 4 \end{matrix} \begin{bmatrix} K_1 + .36K_3 & -.48K_3 \\ -0.48K_3 & K_2 + .64K_3 \end{bmatrix}$$

The final equation is

$$\begin{bmatrix} K_1 + .36K_3 & -.48K_3 \\ -0.48K_3 & K_2 + .64K_3 \end{bmatrix} \begin{Bmatrix} u_3 \\ u_4 \end{Bmatrix} = \begin{Bmatrix} -4000 \\ 8000 \end{Bmatrix}$$

Substituting values for k_1, k_2, and k_3, we get

$$10^5 \begin{bmatrix} 9.66 & -2.88 \\ -2.88 & 6.34 \end{bmatrix} \begin{Bmatrix} u_3 \\ u_4 \end{Bmatrix} = \begin{Bmatrix} -4000 \\ 8000 \end{Bmatrix}$$

$$u_3 = -0.0000438 in$$
$$u_4 = -0.012414 in$$

$$\sigma_1 = \frac{P}{A} = \frac{K_1 \Delta u}{A_1} = \frac{(7.5 \times 10^5)(-0.0000438)}{1.5} = -214 psi$$
$$\sigma_2 = \frac{P}{A} = \frac{K_2 \Delta u}{A_2} = \frac{(2.5 \times 10^5)(-0.012414)}{1.0} = 3015 psi$$
$$\sigma_3 = \frac{P}{A} = \frac{K_3 \Delta u}{A_{31}} = \frac{(6 \times 10^5)(-0.0000438 \cos 53.1^0 + .012414 \sin 53.1^0)}{1.0} = 6119 psi$$

CHAPTER 4

Beam Element

4.1 Introduction

Beam element is a very versatile line-element, it has six degrees of freedom at each node, which include, translations and rotations along the x, y, and z directions, respectively. Figure 4.1 shows the positive directions of these displacements.

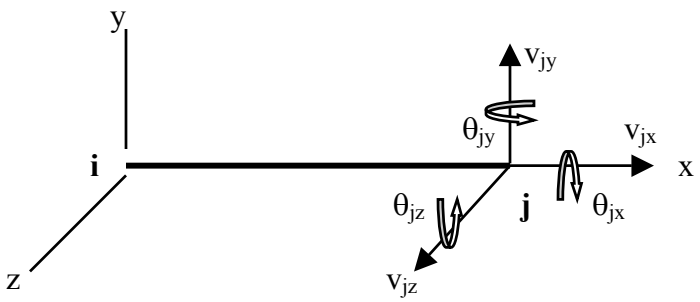


Figure 4.1 Beam Element with six degrees of freedom at each node

Beam element is employed to simulate a slender structure that has an uniform cross section. The element is unsuitable for structures that have complex geometry, holes, and points of stress concentration.

The stiffness constant of a beam element is derived by combining the stiffness constants of a beam under pure bending, a truss element, and a torsion bar. Thus, a beam element can represent a beam in bending, a truss element, and a torsion bar. In FEA it's a common practice to use beam elements to represent all or any of these three loads.

We will derive the element stiffness equation for a beam element by first deriving the stiffness equation of a beam in bending, and then superimposing the stiffness of a truss and a torsion bar element.

4.1 Derivation of a Stiffness Equation for a Beam Element Under Pure Bending

A beam, such as, a cantilever beam, under pure bending (without axial loads or torsional loads), has two-degrees of freedom at any point, transverse deflection v and rotation θ, as shown in Figure 4.2.

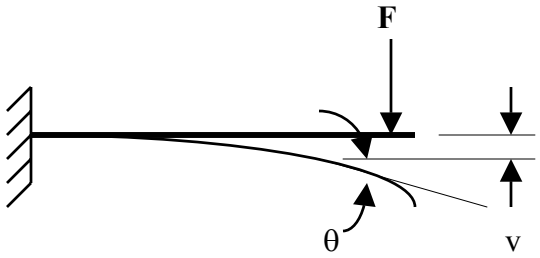


Figure 4.2 Cantilever Beam with it's DOF, v and θ

A beam element has a total of four degrees of freedom, two at each node. Since there are four degrees of freedom, the size of the stiffness matrix of a beam element has the size 4 x 4.

We will derive the stiffness matrix equation using a simple method, known as Stiffness Influence Coefficient Method. In this procedure, a relationship between force and the coefficients that influence stiffness is established. For a beam element, these coefficient consist of: the modulus of elasticity, moment of inertia, and length of the element. For a two-node beam element, there are two deflections and two rotations, namely, v_1, θ_1, v_2, and θ_2. Force and influence coefficient relationship is established by setting each of the four deflection values to unity, with the remaining deflection values equal to zero. The procedure follows.

Consider a beam element, loaded in such a way that it has the deflection values: $v_i = 1$, $\theta_i = 0$, $v_j = 0$, $\theta_j = 0$

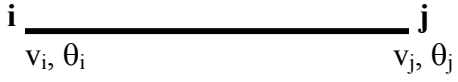
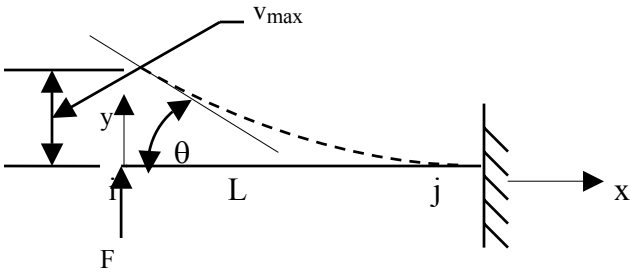


Figure 4.3 Beam Element

The above deflections can be produced by a combination of load conditions, shown in figure 4.4.

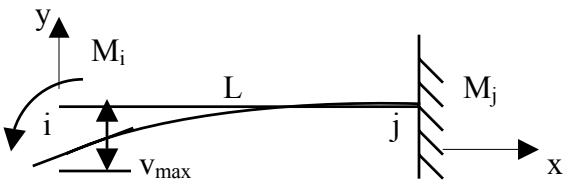
The following deflection relationships for loading of Figures 4.4 (a) and (b) can be found in any Machine Design Handbook, and is given as,



(a)

$$v_{max} = (FL^3)/(3EI)$$

$$\theta = -(FL^2)/(2EI)$$



(b)

$$v_{max} = -(ML^2)/(2EI)$$

$$\theta = (ML)/(EI)$$

Figure 4.4

Applying these relationships to the beam of Figure 4.3, we get,

$1 = v_i = (v_i)_F + (v_i)_M$

$$1 = v_i = (F_i L^3)/3\,EI - (M_i L^2)/2EI \qquad (4.1)$$

and $\theta = 0 = (\theta)_F + (\theta)_M$

$$0 = -(F_i L^2)/2EI + (M_i L)/EI \qquad (4.2)$$

Solving Equations (4.1) and (4.2), we get,

$F_i = (12EI)/L^3$ (A)

$F_j = -F_i = -(12EI)/L^3$ (B)

$M_i = (6EI)/L^2$ (C)

From Figure 4. 4 (a) and (b),

$$M_j = F_i L - M_i$$
$$= (12EI)/L^2 = (6EI)/ L^2$$
$$= (6EI)/ L^2 \qquad (D)$$

Writing equations (A) through (D) in a matrix form we get,

$$\begin{pmatrix} F_i \\ M_i \\ F_j \\ M_j \end{pmatrix} = \begin{pmatrix} (12EI)/L^3 \\ (6EI)/ L^2 \\ -(12EI)/ L^3 \\ (6EI)/ L^2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} (12EI)/ L^3 & 0 & 0 & 0 \\ (6EI)/ L^2 & 0 & 0 & 0 \\ -(12EI)/ L^3 & 0 & 0 & 0 \\ (6EI)/ L^2 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Using a similar procedure and setting the following deflection values:

$v_i = 0$, $\theta_i = 1$, $v_j = 0$, $\theta_j = 0$, we get,

$$\begin{pmatrix} F_i \\ M_i \\ F_j \\ M_j \end{pmatrix} = \begin{pmatrix} (6EI)/L^2 \\ (4EI)/ L \\ -(6EI)/ L^2 \\ (2EI)/ L \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & (6EI)/ L^2 & 0 & 0 \\ 0 & (4EI)/ L & 0 & 0 \\ 0 & -(6EI)/ L^2 & 0 & 0 \\ 0 & (2EI)/ L & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad (4.6)$$

Similarly, setting $v_j = 1$ and , $\theta_j = 1$, respectively, and keeping all other deflection values to zero, we get the final matrix as,

$$\begin{pmatrix} F_i \\ M_i \\ F_j \\ M_j \end{pmatrix} = \begin{pmatrix} (12EI)/L^3 & (6EI)/ L^2 & -(12EI)/ L^3 & (6EI)/ L^2 \\ (6EI)/ L2 & (4EI)/ L & -(6EI)/ L^2 & (2EI)/ L \\ -(12EI)/ L^3 & -(6EI)/ L^2 & (12EI)/L^3 & -(6EI)/ L^2 \\ (6EI)/ L^2 & (2EI)/ L & -(6EI)/ L^2 & (4EI)/ L \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad (4.7)$$

Note that, the first term on the RHS of the above equation is the stiffness matrix and the second term is the deflection. In the case where deflections are other than unity, the above equation will provide an element equation for a beam (in bending), which can be written as,

$$\begin{Bmatrix} F_i \\ M_i \\ F_j \\ M_j \end{Bmatrix} = \begin{pmatrix} (12EI)/L^3 & (6EI)/L^2 & -(12EI)/L^3 & (6EI)/L^2 \\ (6EI)/L2 & (4EI)/L & -(6EI)/L^2 & (2EI)/L \\ -(12EI)/L^3 & -(6EI)/L^2 & (12EI)/L^3 & -(6EI)/L^2 \\ (6EI)/L^2 & (2EI)/L & -(6EI)/L^2 & (4EI)/L \end{pmatrix} \begin{Bmatrix} v_i \\ \theta_i \\ v_j \\ \theta_j \end{Bmatrix} \quad (4.7)$$

Where F_i, M_i, F_j, M_j are the loads corresponding to the deflections v_i, θ_i, v_j, θ_j.

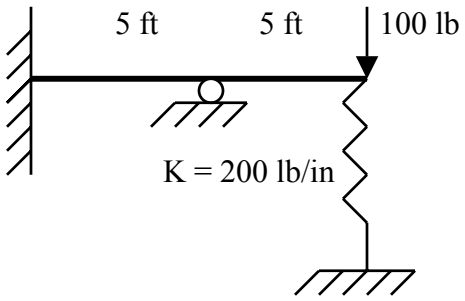
Equation (4.7) is the equation of a beam element, which is under pure bending load (no axial or torsion loads). The stiffness matrix is a 4 x 4, symmetric matrix. Using this equation, we can solve problems in which several beam elements are connected in an uni-axial direction. The assembly procedure is identical to the truss elements. However, if the beam elements are oriented in more than one direction, we will have to first transform the above equation (4.7) in to a global stiffness matrix equation (analogues to the procedure used for truss elements).

For a beam element, transformation of a local stiffness matrix into a global equation involves very complex trigonometric relations, and therefore, we will defer the derivations at this time. However, Equation (4.7) can be used for solving a beam problem, loaded under bending loads. In order to understand the application of this equation, we will apply it to solve some statically indeterminate problems.

Example 1

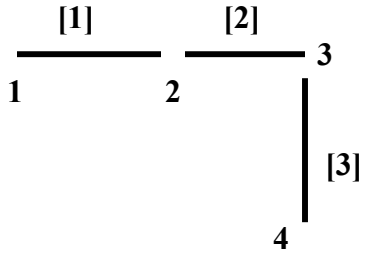
For the beam shown, determine the displacements and slopes at the nodes, forces in each element, and reactions at the supports.

$E = 1.4 \times 10^6$ psi, $I = 2.4$ in^4



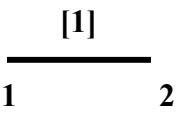
Solution

The beam structure is descritized into three elements and 4-nodes, as shown.



First, we will find the element stiffness matrix for each element, next we will assemble the stiffness matrices, apply the boundary conditions, and finally, solve for node deflection. Internal forces and reactions are calculated by back-substituting the deflections in the structural equation.

Element 1

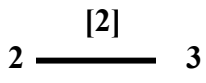


$EI/L^3 = (1.4 \times 10^6) \times (2.4)/(5 \times 12)^3 = 15.55$

The general equation of a stiffness matrix is given as,

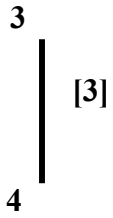
$$[K_e]^{(1)} = (EI/L^3) \begin{pmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{pmatrix} \begin{matrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{matrix}$$

Element 2



$$[K_e]^{(1)} = (EI/L^3) \begin{pmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{pmatrix} \begin{matrix} v_2 \\ \theta_2 \\ v_3 \\ \theta_3 \end{matrix}$$

Element 3



$$[K_e]^{(3)} = \begin{pmatrix} K & -K \\ -K & K \end{pmatrix} \begin{matrix} v_3 \\ v_4 \end{matrix}$$

To get the global stiffness matrix, we will use the same procedure used for assembling truss element stiffness equations. In terms of E, L, and I the assembled global stiffness matrix is,

	v_1	θ_1	v_2	θ_2	v_3	θ_3	v_4
v_1	12	6L	-12	6L	0	0	0
θ_1		$4L^2$	-6L	$2L^2$	0	0	0
v_2			24	0	-12	6L	0
θ_2				$8L^2$	-6L	$2L^2$	0
v_3					12 +K'	-6L	- K'
θ_3						$4L^2$	0
v_4	SYMMETRY						K'

x (EI) /(L^3)

Where K' = (K) x [L^3 / (EI)]

Our next step is to write the structural equation; however, we can reduce the size of the stiffness matrix by applying the given boundary conditions:

$v_1 = \theta_1 = 0$ node 1 is fixed

$v_2 = 0$ node 2 has no vertical deflection, but it's free to rotate.

$V_4 = 0$ node 4 is fixed.

The reduced stiffness matrix is

$$K_G = EI / (L^3) \begin{pmatrix} 8L^2 & -6L & 2L^2 \\ -6L & 12+K' & -6L \\ 2L^2 & -6L & 4L^2 \end{pmatrix}$$

Substituting the values of E, L, and I the structural equation can be written as,

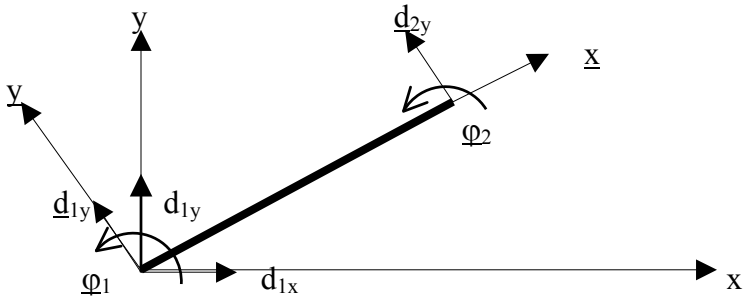
$$\begin{pmatrix} 0 \\ -100 \\ 0 \end{pmatrix} = (15.55) \begin{pmatrix} 1152 & -72 & 288 \\ -72 & 16.11 & -72 \\ 288 & -72 & 576 \end{pmatrix} \begin{pmatrix} \theta_2 \\ v_3 \\ \theta_3 \end{pmatrix}$$

Solving, we get

$\theta_2 = -\ 0.0032$ rad
$v_3 = -\ 0.4412$ in
$\theta_3 = -0.0095$ rad

4.3 Arbitrarily Oriented 2-D Beam Element

The stiffness equation for an arbitrarily oriented beam element can be derived with a procedure similar to the truss element.



$\underline{d}_{1y} = d_{1y} \cos\theta - d_{1x} \sin\theta = d_{1y}\, c - d_{1x}\, s$

$\underline{d}_{2y} = d_{1y} \cos\theta - d_{2x} \sin\theta = d_{2y}\, c - d_{2x}\, s$

and $\underline{\varphi}_1 = \varphi_1,\ \underline{\varphi}_2 = \varphi_2$

Note: The underscored terms represent local coordinate values. Thus, $\underline{x}$ and $\underline{y}$ are local coordinates and x and y are global coordinates.

The above equations can be written in a matrix form,

$$\begin{pmatrix} \underline{d}_{1y} \\ \underline{\varphi}_1 \\ \underline{d}_{2y} \\ \underline{\varphi}_2 \end{pmatrix} = \begin{pmatrix} -s & c & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -s & c & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} d_{1x} \\ d_{1y} \\ \varphi_1 \\ d_{2x} \\ d_{2y} \\ \varphi_2 \end{pmatrix}$$

Let $T = \begin{pmatrix} -s & c & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -s & c & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$, the transformation matrix.

Thus, $\{\underline{d}\} = [T]\ \{d\}$

Local ← Global →

Note that angle φ is independent of the coordinate systems, and $\underline{\varphi}_1 = \varphi_1,\ \underline{\varphi}_2 = \varphi_2$

As derived in the case of the truss element, relationship between local and global stiffness matrices is given as

$$[k_g] = [T]\,[\underline{k}]\,[T]$$

Where, $[k_g]$ = Global stiffness matrix of an element
$[T]$ = Transformation matrix
$[\underline{k}]$ = Local stiffness matrix of the element

Substituting the values of [T] and $[\underline{k}]$, we get the global equation of a beam element oriented arbitrarily at an angle θ as,

$$k = EI/L \begin{pmatrix} 12S^2 & -12SC & -6LS & -12S^2 & -12SC & -6LS \\ & 12C^2 & 6LC & 12SC & -12C^2 & 6LC \\ & & 4L^2 & 6LS & -6LC & 2L^2 \\ & & & 12S^2 & -12SC & 6LS \\ \text{Symmetry} & & & & 12C^2 & 4L^2 \end{pmatrix}$$

This is the equation of a beam element (without axial or torsional load, and oriented at an angle θ.

Also, $S = \sin\theta$, $C = \cos\theta$ in the above equation.

4.4 Beam Element with Combined Bending and Axial loads

First, we will derive the stiffness matrix in local coordinates and then convert it in to global coordinates.

4.4.1 Stiffness matrix of a beam element with bending and axial loads in local coordinates

The stiffness equation for the combined bending and axial load can be written by superimposing the axial stiffness terms over the bending stiffness.

For axial loading, the structural equation is,

$$\begin{pmatrix} f_{1x} \\ f_{2x} \end{pmatrix} = AE/L^3 \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} d_{1x} \\ d_{2x} \end{pmatrix}$$

And for bending loading, the structural equation is,

$$\begin{pmatrix} f_{1y} \\ m_1 \\ f_{2y} \\ m_2 \end{pmatrix} = AE/L^3 \begin{pmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{pmatrix} \begin{pmatrix} d_{1y} \\ \varphi_1 \\ d_{2y} \\ \varphi_2 \end{pmatrix}$$

Therefore, the combined loading equation is

$$\begin{pmatrix} f_{1x} \\ f_{1y} \\ m_1 \\ f_{2x} \\ f_{2y} \\ m_2 \end{pmatrix} = \begin{pmatrix} C_1 & 0 & 0 & -C_1 & 0 & 0 \\ 0 & 12C_2 & 6C_2L & 0 & -12C_2 & 6C_2L \\ 0 & 6C_2L & 4C_2L^2 & 0 & -6C_2L & 2C_2L^2 \\ -C_1 & 0 & 0 & C_1 & 0 & 0 \\ 0 & -12C_2 & -6C_2L & 0 & 12C_2 & -6C_2L \\ 0 & 6C_2L & 2C_2L^2 & 0 & -6C_2L & 4C_2L^2 \end{pmatrix} \begin{pmatrix} d_{1x} \\ d_{1y} \\ \varphi_1 \\ d_{2x} \\ d_{2y} \\ \varphi_2 \end{pmatrix}$$

$$\text{And, } [k] = \begin{bmatrix} C_1 & 0 & 0 & -C_1 & 0 & 0 \\ 0 & 12C_2 & 6C_2L & 0 & -12C_2 & 6C_2L \\ 0 & 6C_2L & 4C_2L^2 & 0 & -6C_2L & 2C_2L^2 \\ -C_1 & 0 & 0 & C_1 & 0 & 0 \\ 0 & -12C_2 & -6C_2L & 0 & 12C_2 & -6C_2L \\ 0 & 6C_2L & 2C_2L^2 & 0 & -6C_2L & 4C_2L^2 \end{bmatrix}$$

Where, $C_1 = AE/L$, and $C_1 = EI/L^3$

4.4.2 Transformation matrix for combined Bending and Axial loading.

For the axial loading, the relationship between the local and global coordinates was derived earlier, as

$$\begin{aligned}\hat{d}_{1x} &= d_{1x}\cos\theta + d_{1y}\sin\theta \\ &= d_{1x}C + d_{1y}S \\ \hat{d}_{2x} &= d_{2x}C + d_{2y}S\end{aligned}$$

Also, for bending load, derived previously,

$$\begin{aligned}\hat{d}_{1y} &= d_{1y}C - d_{1x}S \\ \hat{\phi}_1 &= \phi \\ \hat{d}_{2y} &= d_{2y}C - d_{2x}S \\ \hat{\phi}_2 &= \phi\end{aligned}$$

Therefore, the relationship for the combined bending and axial loading can be written as

$$\begin{Bmatrix} \hat{d}_{1x} \\ \hat{d}_{1y} \\ \hat{\phi}_1 \\ \hat{d}_{2x} \\ \hat{d}_{2y} \\ \hat{\phi}_2 \end{Bmatrix} = \begin{bmatrix} C & S & 0 & 0 & 0 & 0 \\ -S & C & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & C & S & 0 \\ 0 & 0 & 0 & -S & C & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} d_{1x} \\ d_{1y} \\ \phi_1 \\ d_{2x} \\ d_{2y} \\ \phi_2 \end{Bmatrix}$$

Or,

$$\{\hat{d}\} = [T]\{d\}$$

$$\text{Where, } [T] = \begin{bmatrix} C & S & 0 & 0 & 0 & 0 \\ -S & C & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & C & S & 0 \\ 0 & 0 & 0 & -S & C & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

4.4.3 2-D Beam Element Equation for Combined Loading – Axial and Bending – at an Arbitrary Orientation θ

Substituting the values of $\left[\hat{K}\right]$ and [T] into the equation $[K] = [T]^T [\hat{K}][T]$, we get

$$K = \frac{E}{L}\begin{bmatrix} AC^2 + \frac{12I}{L^2}S^2 & \left(A - \frac{12I}{L^2}\right)CS & -\frac{6I}{L}S & -\left(AC^2 + \frac{12I}{L^2}S^2\right) & -\left(A - \frac{12I}{L^2}\right)CS & -\frac{6I}{L}S \\ & AS^2 + \frac{12I}{L^2}C^2 & \frac{6I}{L}C & -\left(A - \frac{12I}{L^2}\right)CS & -\left(AS^2 + \frac{12I}{L^2}C^2\right) & \frac{6I}{L}C \\ & & 4I & \frac{6I}{L}S & -\frac{6I}{L}C & 2I \\ & & & AC^2 + \frac{12I}{L^2}S^2 & \left(A - \frac{12I}{L^2}\right)CS & \frac{6I}{L}S \\ & \textit{symmetry} & & & AS^2 + \frac{12I}{L^2}C^2 & -\frac{6I}{L}C \\ & & & & & 4I \end{bmatrix}$$

4.5 2-D Beam Element with combined loading Bending, Axial, and Torsion (θ = 0)

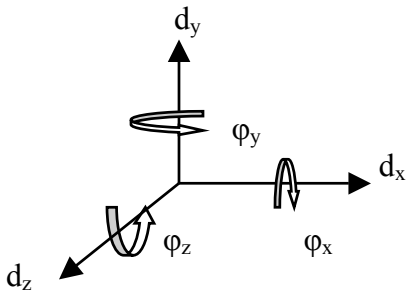
The torsional loads are m_{1x} and m_{2x}, and the corresponding deflections are, ϕ_{1x} and ϕ_{2x}

The torsional structural equation is:

$$\begin{pmatrix} m_{1x} \\ \\ m_{1x} \end{pmatrix} = JG/L \begin{pmatrix} 1 & -1 \\ \\ -1 & 1 \end{pmatrix} \begin{pmatrix} \varphi_{1x} \\ \\ \varphi_{2x} \end{pmatrix}$$

These terms can be superimposed on the stiffness equation derived previously for the combined bending and axial loads.

3-D Beam Element:



A 3-D beam element has 6 DOF at each node, and 12 DOF for each element. The stiffness matrix can be derived by super-imposing the axial, bending, and torsion loadings in the XY, XZ, and YZ planes. The equation is,

$$
K = \begin{bmatrix}
\hat{d}_{1x} & \hat{d}_{1y} & \hat{d}_{1z} & \hat{\phi}_{1x} & \hat{\phi}_{1y} & \hat{\phi}_{1z} & \hat{d}_{2x} & \hat{d}_{2y} & \hat{d}_{2z} & \hat{\phi}_{2x} & \hat{\phi}_{2y} & \hat{\phi}_{2z} \\
\frac{AE}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{AE}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} & 0 & -\frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} \\
0 & 0 & \frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 & 0 & 0 & -\frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 \\
0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 \\
0 & 0 & -\frac{6EI_y}{L^2} & 0 & \frac{4EI_y}{L} & 0 & 0 & 0 & \frac{6EI_y}{L^2} & 0 & \frac{2EI_y}{L} & 0 \\
0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{4EI_z}{L} & 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{2EI_z}{L} \\
-\frac{AE}{L} & 0 & 0 & 0 & 0 & 0 & \frac{AE}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{12EI_z}{L^3} & 0 & 0 & 0 & -\frac{6EI_z}{L^2} & 0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & -\frac{6EI_z}{L^2} \\
0 & 0 & -\frac{12EI_y}{L^3} & 0 & \frac{6EI_y}{L^2} & 0 & 0 & 0 & \frac{12EI_y}{L^3} & 0 & \frac{6EI_y}{L^2} & 0 \\
0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 \\
0 & 0 & -\frac{6EI_y}{L^2} & 0 & \frac{2EI_y}{L} & 0 & 0 & 0 & \frac{6EI_y}{L^2} & 0 & \frac{4EI_y}{L} & 0 \\
0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{2EI_z}{L} & 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{4EI_z}{L}
\end{bmatrix}
$$

APPENDIX A

Matrices

Introduction

- FEA solves structural problems by defining its behavior in terms of differential equations. These equations are converted into a set of linear algebraic equations, which are represented in the form of matrix equations.
- Matrix equations are easy to solve with computers, and therefore it's important to understand the matrix algebra.
- We must always remember that when we solve matrix equations, we are, in fact, solving the simultaneous algebraic equations (and the differential equations) of the structure.
- Since all the FEA calculations are based on matrix algebra, it's important to understand the matrix operations: addition, subtraction, multiplication, inversion, transpose, etc.

Relationship between Algebraic and Matrix equations

$$\begin{aligned} a_{11}x_{11} + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_{11} + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_{11} + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned} \qquad \text{(A)}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} b1 \\ b2 \\ b3 \end{Bmatrix} \quad \text{or} \quad [A]\{x\} = \{b\} \qquad \text{(B)}$$

- Equations (A) & (B) represents the same system of equations.

Transpose Matrix

Let $[A] = \begin{pmatrix} 3 & 8 \\ 5 & 2 \\ 6 & 3 \end{pmatrix}$ then

$[A]^T = \begin{pmatrix} 3 & 5 & 6 \\ 8 & 2 & 3 \end{pmatrix}$ The rows & column position of elements is interchanged.

Also, $[AB]^T = [B]^T [A]^T$

Orthogonal Matrix

Orthogonal matrix is a square matrix. For an orthogonal matrix, the inverse of the matrix is equal to it's transpose, thus,

$$[A]^{-1} = [A]^T$$

Also $[A][A]^{-1} = [I]$

$$[A^T][A] = [I]$$

$$[A][A^T] = [I] \quad \text{Thus,}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} a11 & a_{21} & a_{31} \\ a12 & a_{22} & a_{32} \\ a13 & a_{23} & a_{33} \end{pmatrix} = [I]$$

Matrix partitioning

Matrix partitioning is the sub division of a matrix into several smaller matrices, called submatrices.

- In FEA, a matrix is usually partitioned into two or four submatrices.
- A matrix is partitioned so that the number of columns to the left of the vertical partition in the coefficient matrix equals the number of rows above the horizontal partition.
- The submatrices can be manipulated with matrix operations in the same manner as the original matrix.

EX:

$$\left(\begin{array}{cc|c} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \hline a_{31} & a_{32} & a_{33} \end{array}\right) \left\{\begin{array}{c} x_1 \\ x_2 \\ \hline x_3 \end{array}\right\} = \left\{\begin{array}{c} b_1 \\ b_2 \\ \hline b_3 \end{array}\right\}$$

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{Bmatrix} X_1 \\ X_2 \end{Bmatrix} = \begin{Bmatrix} B_1 \\ B_2 \end{Bmatrix}$$

$$[A_{11}]\{X_1\} + [A_{12}]\{X_2\} = [B_1]$$
$$[A_{21}]\{X_1\} + [A_{22}]\{X_2\} = [B_2]$$

- The advantage of partitioning a matrix is that the resulting sub matrices can be manipulated with matrix operations in the same way as the original.
- The ability to deal with a system of equations, either as one matrix equation or two matrix equations provides a convenient means for solving an equation system when some of the unknowns are in the force vector [B] and the remainder are in the displacement vector X.
- This is frequently the case when applying the FEA method to structures, where the number of algebraic equations is very large and contains unknown forces.

Inverse Matrix

$[A]\,[A^{-1}] = [I]$
$[A^{-1}]$ is the inverse of matrix [A].

The equation [A][X] = [B] is solved by pre-multiplying the L.H.S with A^{-1}, thus

$[A^{-1}][A][X] = [A^{-1}][B]$
or $[X] = [A^{-1}][B]$

* [A] must be a square matrix. Non-square matrix cannot be inverted.

Orthogonal Matrix

If $A^{-1} = A^{T}$, the matrix [A] is called an orthogonal matrix.

Determinants

The determinant of a square matrix is a single number, a scalar quantity. Its symbol is

det[A] or $\left| A \right|$

Example:

$$\det[A] = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = (a_{11})(a_{22})-(a_{12})(a_{21})$$

Sub-matrix solution procedure:

- In FEA, generally, some of the displacements and most of the forces are known and the structural matrix equation is best solved by partitioning it.
- The rows and columns of the stiffness matrix are arranged so that the known forces and known displacements are placed in submatrices.

Example:

$$\begin{pmatrix} K_{11} & K_{12} & K_{13} & K_{14} & K_{15} \\ K_{21} & K_{22} & K_{23} & K_{24} & K_{25} \\ K_{31} & K_{32} & K_{33} & K_{34} & K_{35} \\ K_{41} & K_{42} & K_{43} & K_{44} & K_{45} \\ K_{51} & K_{52} & K_{53} & K_{54} & K_{55} \end{pmatrix} \begin{Bmatrix} q_1 \\ q_{02} \\ q_3 \\ q_{04} \\ q_5 \end{Bmatrix} = \begin{Bmatrix} f_{01} \\ f_2 \\ f_{03} \\ f_4 \\ f_{05} \end{Bmatrix}$$

- The displacements and the forces with subscript '0' are known. [q_{02}, q_{04} & f_{01}, f_{03}, f_{05}]

- Before partitioning, the rows of matrix K are rearranged so that the known forces are grouped in the first three rows, as shown below.

$$\begin{pmatrix} K_{11} & K_{12} & K_{13} & K_{14} & K_{15} \\ K_{31} & K_{32} & K_{33} & K_{34} & K_{35} \\ K_{51} & K_{52} & K_{53} & K_{54} & K_{55} \\ K_{21} & K_{22} & K_{23} & K_{24} & K_{25} \\ K_{41} & K_{42} & K_{43} & K_{44} & K_{45} \end{pmatrix} \begin{Bmatrix} q_1 \\ q_{02} \\ q_3 \\ q_{04} \\ q_5 \end{Bmatrix} = \begin{Bmatrix} f_{01} \\ f_{03} \\ f_{05} \\ f_2 \\ f_4 \end{Bmatrix}$$

Next, the columns of the K matrix are rearranged so that the unknown displacements are grouped on the top.

- Note that, the validity of the algebraic equations represented by the matrix equations is maintained only if the change in position of q values is compensated by swapping the corresponding columns in the K matrix.
- We will change the order of displacements to:

$$\begin{Bmatrix} q_1 \\ q_3 \\ q_5 \\ q_{02} \\ q_{04} \end{Bmatrix}$$

Therefore, we must swap the columns in K matrix as:
Column 1: unchanged
Column 2: Replace by column 3
Column 3: Replace by column 5
Column 4: Replace by column 2
Column 5: Replace by column 4

The final matrix eqn. is

$$\left(\begin{array}{ccc|cc} K_{11} & K_{13} & K_{15} & K_{12} & K_{14} \\ K_{31} & K_{33} & K_{35} & K_{32} & K_{34} \\ \underline{K_{51}} & \underline{K_{53}} & \underline{K_{55}} & \underline{K_{52}} & \underline{K_{54}} \\ K_{21} & K_{23} & K_{25} & K_{22} & K_{24} \\ K_{41} & K_{43} & K_{45} & K_{42} & K_{44} \end{array}\right) \left\{\begin{array}{c} q_1 \\ q_3 \\ \underline{q_5} \\ q_{02} \\ q_{04} \end{array}\right\} = \left\{\begin{array}{c} f_{01} \\ f_{03} \\ \underline{f_{05}} \\ f_2 \\ f_4 \end{array}\right\}$$

Now we can partition the matrix as shown. The submatrices can be written as

$$\begin{pmatrix} K_{ff} & K_{fs} \\ K_{sf} & K_{ss} \end{pmatrix} \begin{Bmatrix} q_f \\ q_o \end{Bmatrix} = \begin{Bmatrix} F_o \\ F_s \end{Bmatrix}$$

$$[K_{ff}]\ \{q_f\} + [K_{fs}]\ \{q_o\} = F_o \qquad (1)$$
$$[K_{sf}]\ \{q_f\} + [K_{ss}]\ \{q_o\} = F_s \qquad (2)$$

The equations can be solved for q_f & Fs as

$$\{q_f\} = [K_{ff}^{-1}]\ (\{F_o\} - [K_{fs}]\ \{q_o\})$$

$$\{F_s\} = [K_{sf}]\{q_f\} + [K_{ss}]\{q_o\}$$