

4. GEOMETRIC MODELING

4.1 THE ROLE OF MODELING AND COMMUNICATION

The concept of the designer working with models is fundamental to the treatment of computer-aided design. It is important to distinguish here between models of the design process, which essentially attempt to describe the pattern that designers follow in the design of products, and models of the designs themselves. At the start of the process, one particular design is an abstract: the physical artefact does not exist, so until it is constructed or manufactured there needs to be some models of the design for those involved to evaluate, manipulate and refine. Such models may exist as different representations. The modeled geometry of an engineering component may for example be represented in different ways. If the design is very simple, these may just be ideas in the mind of the designer, but for all except the most elementary designs some more formal representation is needed.

Models of the design are used for a variety of purposes. At the most basic level, they are used by the designer to record and manipulate ideas and to provide a basis for the analysis and evaluation. The design process is rarely undertaken by a single person, and therefore the models have a major role in the communication between participants in the process, and to those involved in the manufacture, development and sub-sequent use of the product.

In practice, the designer uses a number of different models depending on what property of the design is to be modeled, and who or what is the target, or receiver, for any communication. Of all the modeled properties, form and structure are of particular importance in engineering, and the most appropriate method of representing these has traditionally been graphical. For many engineers - for example the designers of machines, bridges and vehicles - a major part of their task is to define the shape and arrangement of the component parts of the design. This is conventionally achieved by drawings of form. Other engineers are more concerned with the assembly of standard elements to form a design, and with the way these elements are connected together, and with the flows (for example of energy or material) between parts (this approach is often called a systems engineering approach).

Having the above in mind it could be said that geometric modeling to CAD/CAM is as important as governing equilibrium equations to classical engineering fields as mechanics and thermal fluids. It is also a means (tool) to enable useful engineering analysis and judgment. The degree of geometric detail depends on the analysis procedure that utilizes the geometry. Engineers may also agree that there is no model that is sufficient to study all behavioral aspects of an engineering component or a system. A machine part, for example, can be modeled as a lumped mass rigid body on one occasion or as a distributed mass continuum on another occasion.

4.2 WIREFRAME MODELS

A wireframe model of an object is the simplest, but more verbose, geometric model that can be used to represent it mathematically in the computer. It is sometimes referred to as a stick figure or an edge representation of the object. The word ‘wireframe’ is related to the fact that one may imagine a wire that is bent to follow the object edges to generate the model (Fig. 4.1). Typically, a wireframe model consists entirely of points, lines, arcs and circles, conics, and other curves. Wireframe modeling is the most commonly used technique and all

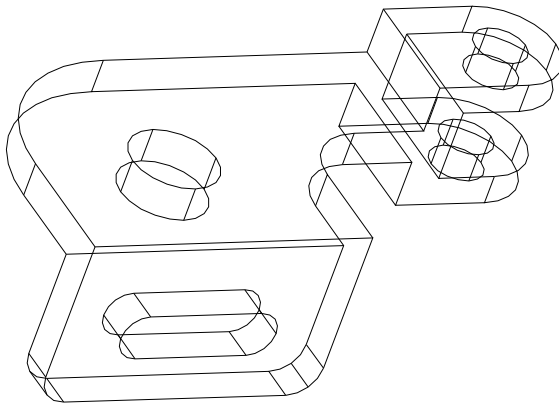


Figure 4.1

commercial CAD/CAM systems have their wireframe-base.

Early wireframe modeling techniques developed in the 1960s were strictly two-dimensional and were designed to automate drafting and simple NC. Users had to construct geometry in the desired various views independently due to the lack of centralization and associativity of the resulting database. Later in the early 1970s, the centralized associative database concept

enabled modeling of three-dimensional objects as wireframe models that can be subject to three-dimensional transformations. Creating geometry in one view is automatically projected and displayed in other views. This represents a substantial saving and flexibility over manual design and drafting. Wireframe modeling is considered a natural extension of traditional methods of drafting. Consequently, it does not require extensive training of users; nor does it demand the use of unusual terminology as surfaces and solids.

Despite its many disadvantages, the major advantage of wireframe modeling is its simplicity to construct. To create the wireframe model one must supply only geometrical information, and only geometrical information is stored in the model's database. Therefore, it does not require as much computer time and memory as does surface or solid modeling. Lastly, the CPU time required to retrieve, edit, or update a wireframe model is usually small compared to surface or solid models.

There are number of disadvantages with wireframe models:

- Primarily, these models are usually ambiguous representations of real objects and rely heavily on human interpretation. A wireframe model of a box offers a typical example where the model may represent more than one orientation of the object depending on which face(s) is assumed to be visible. Models of complex designs having many edges become very confusing and perhaps even impossible to interpret. To overcome this confusion, lines can be hidden,

dashed, or blanked. If done manually, these operations are very tedious, error-prone, and can result in ‘nonsense’ objects.

— Another disadvantage is the lack of visual coherence and information to determine the object profile. The obvious example is the representation of a hole or a curved portion of the object. In most systems, the hole is displayed as two parallel circles separated by the hole length. Some systems may connect a line between the two circles on one side of the hole (Fig. 4.1).

— Wireframe models are also considered lengthy or verbose when it comes to the amount of defining data and command sequence required to construct them. For example, compare the creation of a simple box as a wireframe and as a solid. In the latter, the location of one corner, the length, width, and height are the required input while in the former the drawing of eight lines using the coordinates of eight corners are needed. In this case the user or terminal time needed to prepare and/or input data is substantial and increases rapidly with the complexity of the object being modelled.

Despite the above-mentioned limitations, wireframe models are expected to last and may extend to certain classes of solid modeling. For example, most of the solid modelers (such as Medusa, SolidWorks, Inventor) are based on wireframe input. The simplicity of the geometrical concepts based on wireframe modeling makes them attractive to introduce users to the CAD/CAM field. In addition, at early design stages, designers might just need a sketchpad to try various ideas. Wireframes are ideal to provide them with such a capability. From an industrial point of view, wireframe models may be sufficient to many design and manufacturing needs. From a practical point of view, many companies have large amounts of wireframe databases that are worth millions of dollars and man-hours and therefore make it impossible to get rid of wireframe technology.

Wireframe models form the basis for surface models. Most existing surface algorithms require wireframe entities to generate surfaces.

4.3 SURFACE MODELS OF SOLIDS

A surface model of an object is a more complete and less ambiguous representation than the respective wireframe model. It is also richer in its associated geometric contents, which makes it more suitable for engineering and design applications. Surface models take the modeling of an object one step ahead of wireframe models by providing information on surfaces connecting the object edges. They provide hidden line and shading algorithms to add realism to the displayed geometry.

Surfaces do not exist in conventional engineering drawings. They have not been utilized till development of CAD systems, where it is possible to create numerical objects known as surface. Simply, the surface is mathematical equation, which represents a contour or shape in 3D. The surface is a object which has length, width, but not height. The surface could be thought of as a thin mate-

rial, which could be stretched out and wrapped around any thinkable spatial form.

One complete surface model could be regarded as computer generated tool suitable for a number of engineering applications:

- generating of precise cross-sections for engineering applications;
- output for modeling and exploration of constructions through finite elements applications;
- generating of realistic images for marketing needs via rendering and shading;
- generating the output data for CNC machining;
- creating solids by adding height to the surface or by using the solid cuts – cutting the existing solid by a particular surface to obtain the desired form.

Typically, a surface model consists of wireframe entities that form the basis to create surface entities. Surface description is usually tackled as an extension to the wireframe representation. Analytic and synthetic surface entities are available and provided by most CAD/CAM systems. Databases of surface models are centralized and associative. Thus, manipulating surface entities in one view is automatically reflected in other views. These entities can also be subjected to three-dimensional geometric transformations. Despite their similar look, there is a fundamental difference between surface and solid models. Surface models define only the geometry of their corresponding objects. They store no information regarding the topology of these objects. As an example, if there

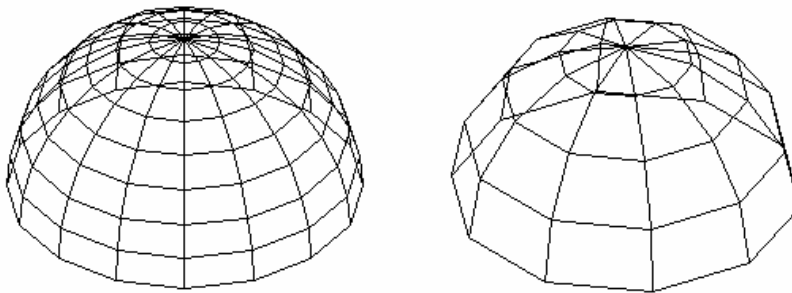


Figure 4.2

are two surface entities that share a wireframe entity (edge), neither the surfaces nor the entity store such information. There is also a fundamental difference between creating surface and solid models.

Visualization of a surface is aided by the addition of artificial lines (called mesh), which criss-cross the surface and so break it up into a network of interconnected patches. Very important thing in constructing surface models is to set the mesh size of a surface entity properly. It should be mentioned here that the finer the mesh size of surface entities in a model, the longer the CPU time to construct the entities and to update the graphics display is needed, and the longer it takes to plot the surface model. In many CAD systems mesh representing object's surface uses planar facets. The mesh density, or number of facets, is defined in terms of matrix of M and N vertices, similar to a grid consisting of col-

umns and rows. Figure 4.2 shows the same surface of revolution drawn with different density of the mesh (mesh size).

To create a surface model, the user begins by constructing wireframe entities and then connecting them appropriately with the proper surface entities. A typical CAD system presents a number of ways to create or generate wireframes. Some of these methods can be listed as follows:

♦ **By using wireframe entities** – lines, arcs, circles, ellipses as well as 2D or 3D splines. Usually wireframe entities must be drawn on different sketch planes in order to form a spatial skeleton.

♦ **By copying individual base surface edges.** An individual base surface edge is that portion of the surface's edge that lies between two nodes. Each selected surface edge is copied, and the edge is displayed as either a 3D polyline or a spline, depending on the desired mode (Fig. 4.3).

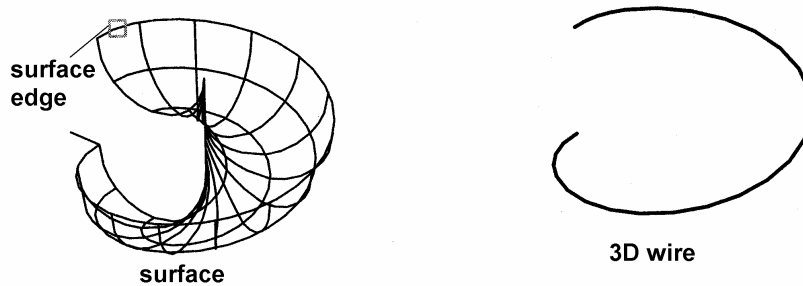


Figure 4.3

♦ **By removing a complete loop** from a particular surface, so that the user can edit its edges. The loop may consist of one or more lines, but this operation converts them to a spline or a 3D polyline for editing (Fig. 4.4).

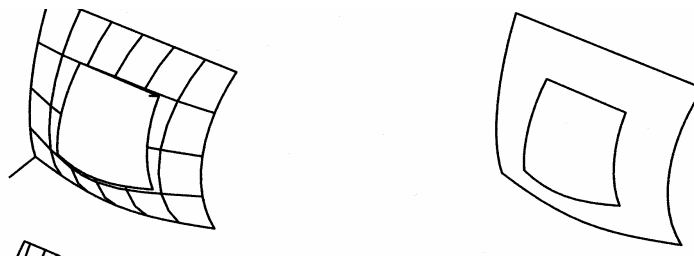


Figure 4.4

♦ **By creating wires** in the M and N directions from an existing surface. These wires (flow wires) are different from surface display lines, which are controlled separately (see above). Flow wires could be created and offset a given distance in a direction normal to the surface (see Fig. 4.5). A positive offset value offsets the flow wires in the same direction as the surface normal. A negative offset value offsets the flow wires in the opposite direction of the surface normal. By examining flow wires, you can verify the quality of the surface or they are sometimes used in manufacturing to generate tool paths.

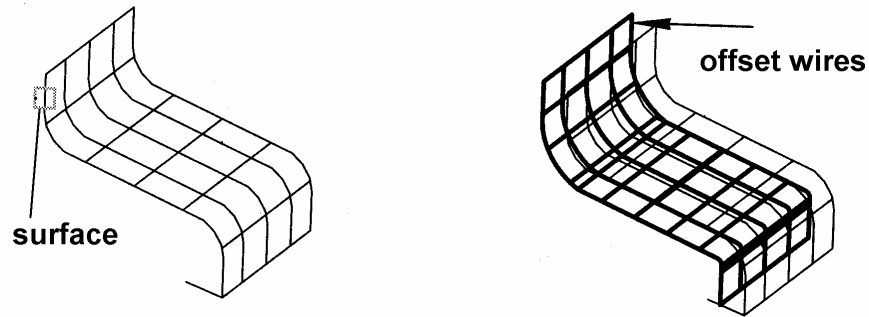


Figure 4.5

♦ *By creating wires as a curve of intersection between two surfaces;*
 ♦ *By cutting a single cross section* or a group of cross sections from a set of selected surfaces. Cross sections can be parallel to one another and to the initial plane, starting at the initial plane and terminate at the specified distance from it. They can also be radial, derived by rotating the initial plane about its X axis (Fig. 4.6).

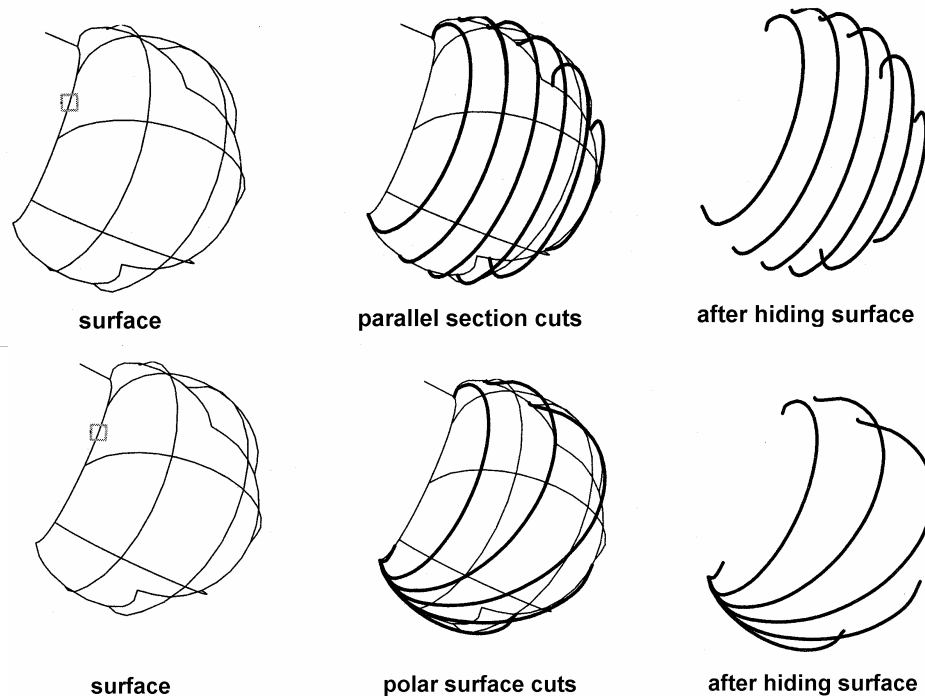


Figure 4.6

4.3.1 SURFACE ENTITIES

Similar to wireframe entities, existing CAD/CAM systems provide designers with both analytic and synthetic surface entities. Analytic entities include plane surface, extruded or ruled surfaces, surface of revolution, tabulated surfaces etc. Synthetic entities include the bicubic Hermite spline surface, B-spline surface, rectangular and triangular Bezier patches, rectangular and triangular Coons patches, and Gordon surface. Some of the most commonly used surface primitives are listed below and given in Table 4.1.

◆ **Plane surface.** This is the simplest surface. It requires three noncoincident points or two parallel or intersecting lines to define an infinite plane. The plane surface is often used for modeling, but can also be used to generate cross-sectional views by intersecting a surface model with it, generate cross sections for mass property calculations, or other similar applications where a plane is needed.

◆ **Surface primitives.** These can be regarded as predetermined 3D shapes as box, cone dish, sphere pyramid torus etc. In order to create such a shape, specific information must be available. For example, to draw a sphere, information about the center of the sphere and radius must be inputted. The procedure for creating 3D shapes is similar to creating 3D solid primitives.

◆ **Surface of revolution.** This is an axisymmetric surface that can model axisymmetric objects. It is generated by rotating a planar wireframe entity in space about the axis of symmetry a certain angle.

◆ **Extruded surface.** A surface created by moving (extruding) a shape along specified direction. The direction could be given by two points or by existing line. One always apply tapering angle to taper the extrusion. For closed curves, entering a positive angle tapers out from the base and entering a negative number tapers in. For other curves, the taper direction is relative to the direction of the curve being extruded. Specifying a taper angle for an object with sharp corners or abrupt changes in curvature might cause the surface to intersect itself. The surfaces of revolution and extruded surfaces are also called 'Motion based surfaces'.

◆ **Swept surface.** A surface created from one or more cross sections swept along a rail (boundary curve). The cross-sections could be dissimilar. In such a case the cross sections are blended as they are swept along the rails. The swept surfaces can also be regarded as motion based surfaces.

When there is an existing wireframe construction, a surface skin over such a wireframe could be created based on several techniques listed briefly below.

◆ **Ruled surface.** This is a linear surface. It interpolates linearly between two nonintersecting boundary curves that define the surface (rails). Rails can be any wireframe entity from straight lines to particular curves. A ruled surface appears as though the user uses a straight edge to blend from one shape to another. This entity is ideal to represent surfaces that do not have any twists or kinks.

◆ **Lofted surface.** In many industries, e.g. shipbuilding, automotive and aircraft construction, the final shape is determined by a process called lofting. In such cases it is necessary to determine the shape of the ship with more accuracy than possible on small 'paper' drawings. Thus, it is common practice to redraw ship lines in full or nearly full size on a large wooden floor located in the 'mold loft'. The skilled craftsmen who perform these tasks are known as 'loftsmen' and the process as 'lofting'. In this process loftsmen use long narrow strips of wood or plastic strips (called 'splines') in order to fit curves through

specified data points. In the present CAD systems a lofted surface is created by wrapping over one or two sets of wires. Each set of wires must have similar attributes, such as being in approximately the same direction.

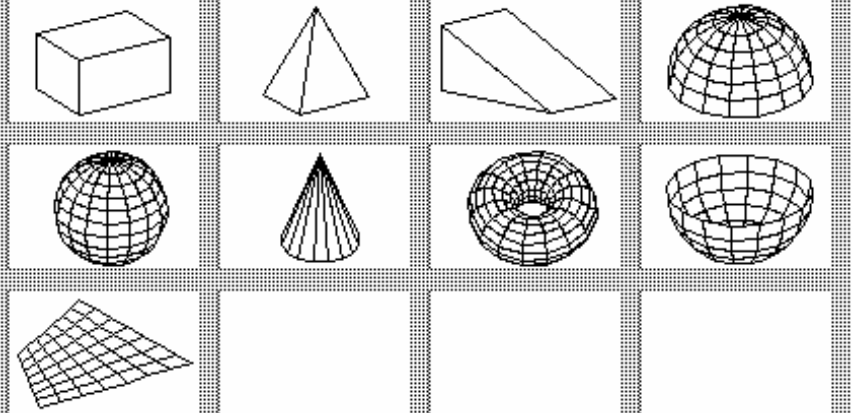
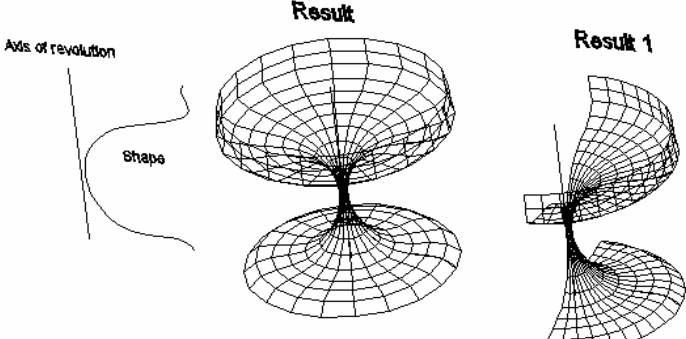
♦ **Fillet surface.** This is a B-spline surface that blends two surfaces together. The two original surfaces may or may not be trimmed.

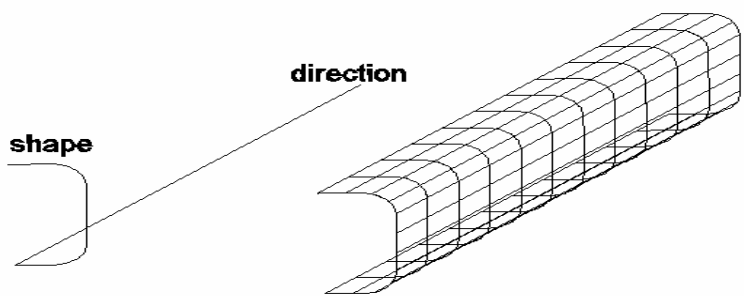
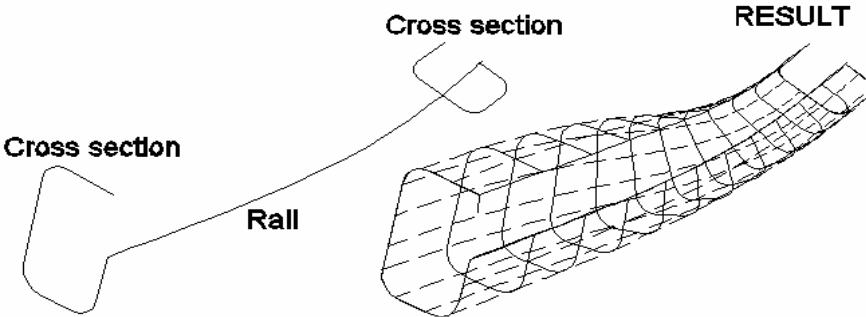
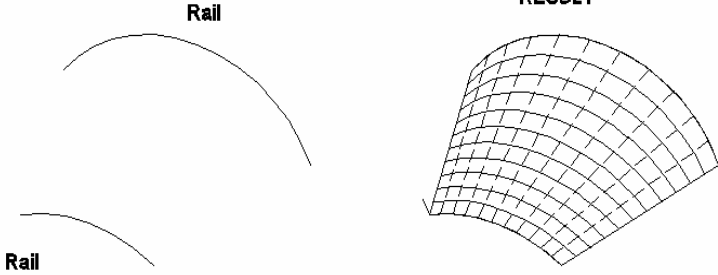
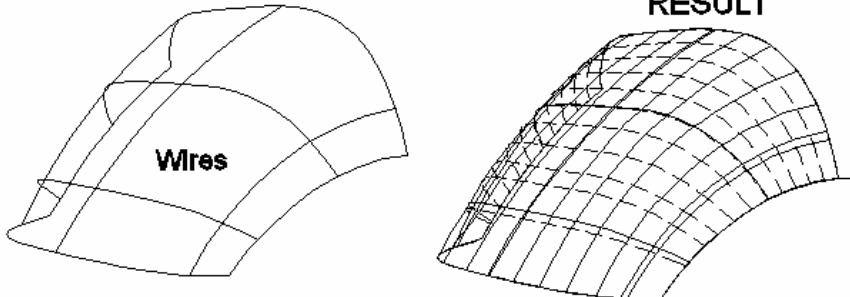
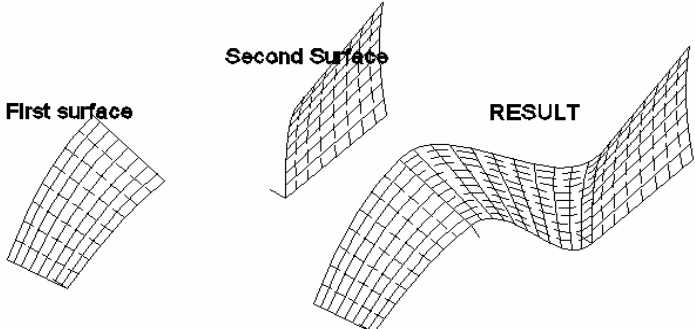
♦ **Bezier surface.** This is a surface that approximates given input data. It is different from the previous surface in that it is a synthetic surface. Similarly to the Bezier curve, it does not pass through all given data points. It is a general surface that permits, twists, and kinks. The Bezier surface allows only global control of the surface.

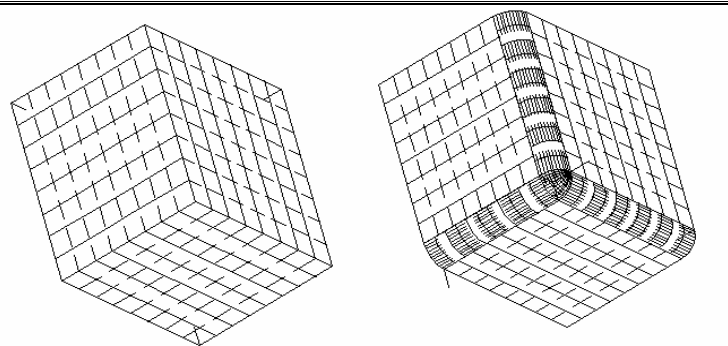
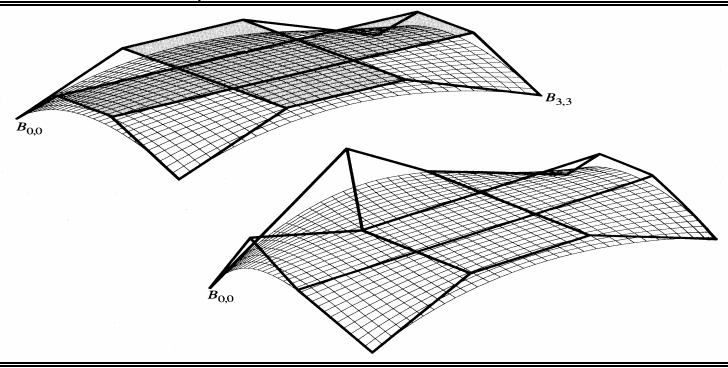
♦ **B-spline surface.** This is a surface that can approximate or interpolate given input data. It is a synthetic surface. It is a general surface like the Bezier surface but with the advantage of permitting local control of the surface.

♦ **Offset surface.** Existing surfaces can be offset to create new ones identical in shape but may have different dimensions. It is a useful surface to use to speed up surface construction. For example, to create a hollow cylinder, the outer or inner cylinder can be created using a cylinder command and the other one can be created by an offset command. Offset surface command becomes very efficient to use if the original surface is a composite one.

Table 4.1

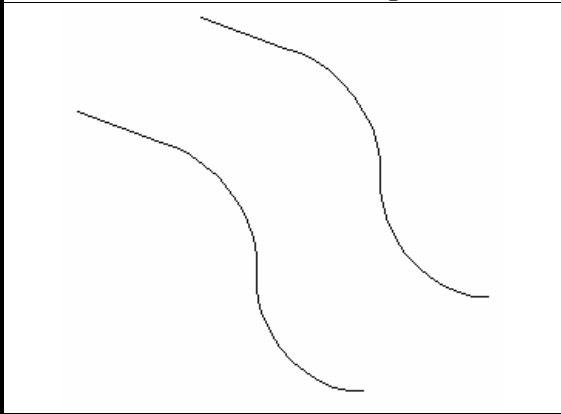
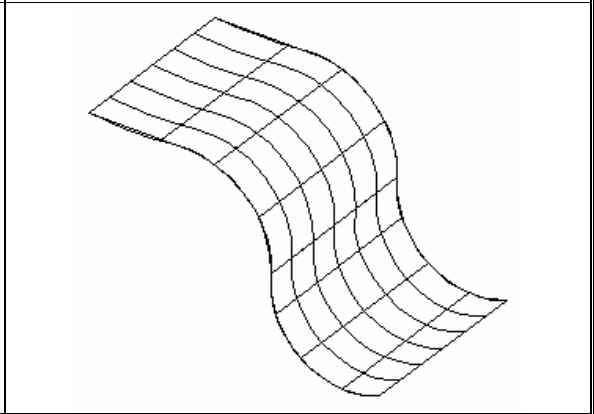
SURFACE PRIMITIVES	
MOTION BASED SURFACES Surface of revolution	

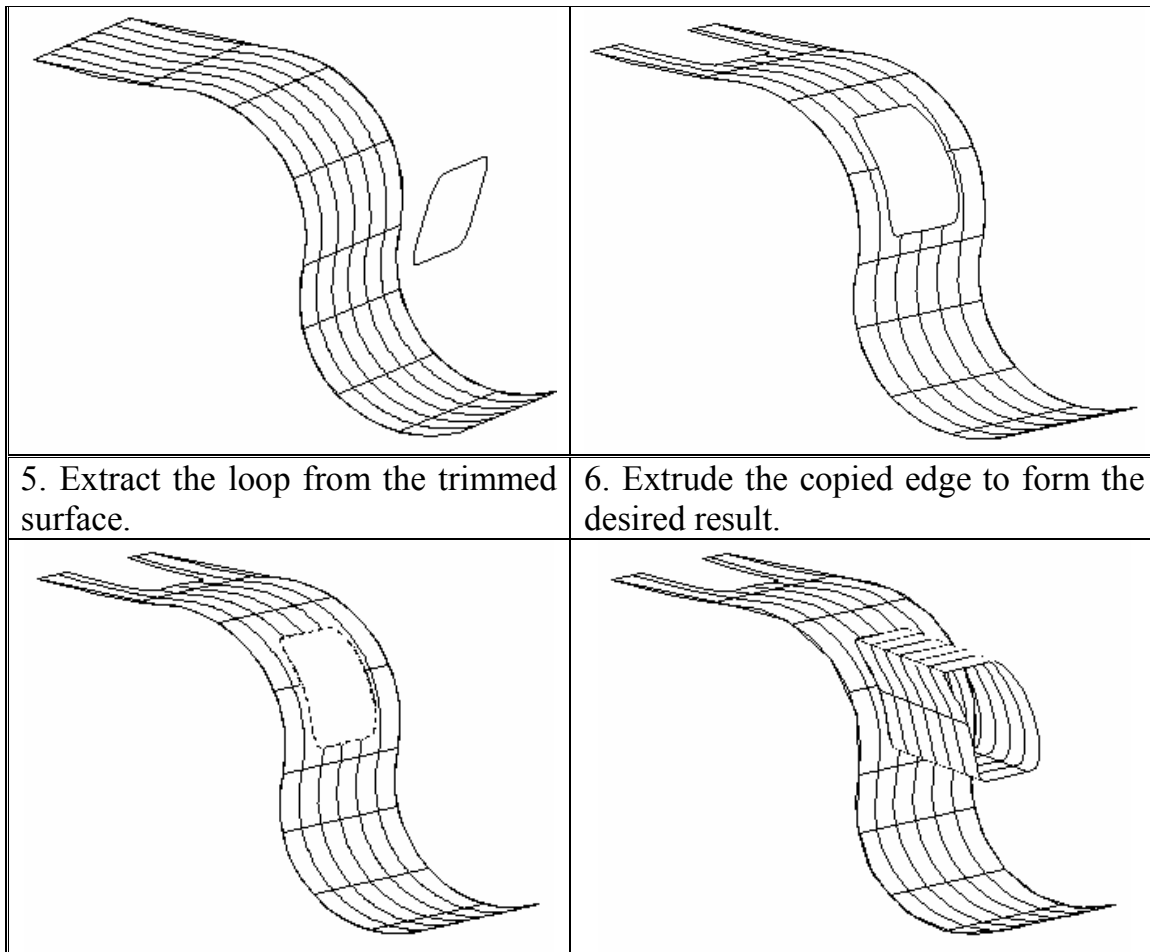
MOTION BASED SURFACES Extruded Surface	RESULT 
MOTION BASED SURFACES Swept Surface	RESULT 
SKIN SURFACES Ruled Surface	RESULT 
SKIN SURFACES Lofted Surface	RESULT 
DERIVED SURFACES Blended Surface	RESULT 

DERIVED SURFACES Fillet Surfaces	
SYNTHETIC SURFACE Bezier Surface	

One brief particular example of forming a surface model is given in Table 4.2.

Table 4.2

1. Start with existing wires (curves) created in different sketch planes	2. Create a ruled surface to wrap over the frames
	
3. Create an additional closed wire-frame on a third sketch plane	4. Project the wireframe on the surface along a particular vector. Trim the surface



4.4 TYPES AND REPRESENTATIONS OF SOLIDS - INTRODUCTION

Solid modeling techniques are based on informationally complete, valid, and unambiguous representations of objects. Simply stated, a complete geometric data representation of an object is one that enables points in space to be classified relative to the object, if it is inside, outside, or on the object. This classification is sometimes called **spatial addressability**.

Solid modelers store more information (geometry and topology) than wireframe or surface modelers (geometry only). Both wireframe and surface modelers are incapable of handling spatial addressability as well as verifying that the model is well formed, the latter meaning that these models cannot verify whether two objects occupy the same space. The completeness and unambiguity of solid models are attributed to the information that the related databases of these models store. Unlike wireframe and surface models, which contain only geometric data, solid models contain both geometric data and topological information of the corresponding objects. The difference between geometry and topology is illustrated in Fig. 4.12. Geometry (sometimes called metric informa-

tion) is the actual dimensions that define the entities of the object. The geometry that defines the object shown in Fig. 4.7 is the lengths of lines L_1 , L_2 , and L_3 , the angles between the lines and the radius R and the center P , of the half-circle. Topology (sometimes called combinatorial structure), on the other hand, is the connectivity and associativity of the object entities. It has to do with the notion of neighborhood; that is, it determines the relational information between object entities. The topology of the object

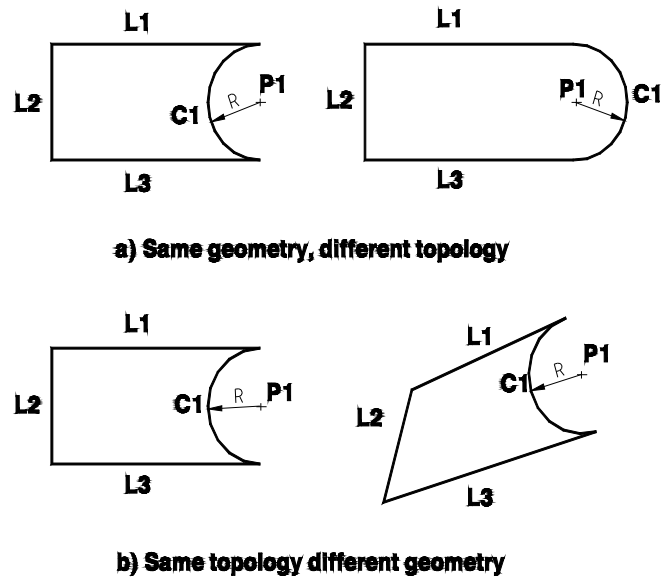


Figure 4.7

shown in Fig. 4.7b can be stated as follows: L_1 shares a vertex (point) with L_2 and C_1 , L_2 shares a vertex with L_1 and L_3 , L_3 shares a vertex with L_2 and C_1 , L_1 and L_3 do not overlap, and P_1 lies outside the object. Based on these definitions, neither geometry nor topology alone can completely model objects. Wireframe and surface models deal only with geometrical information of object, and are therefore considered incomplete and ambiguous. From a user point of view geometry is visible and topology is considered to be nongraphical relational information that is stored in solid model databases and is not visible to users

The properties that a solid model or an abstract solid should capture mathematically can be stated as follows:

1. Rigidity. This implies that the shape of the model is invariant and does not depend on the model location or orientation in space;
2. Homogeneous three-dimensionality. Solid boundaries must be in contact with the interior. No isolated dangling boundaries should be permitted;
3. Finiteness and finite describability. The former property means that the size of the solid is not infinite, while the latter ensures that a limited amount of information should be used to describe the solid. It is connected with the need of limited storage space the computers utilize to store data. It should be noted that the former property does not include the latter and vice versa. For example a cylinder which have a limited height and radius may be described with infinite number of planar faces;
4. Closure under rigid motion and Boolean operations. This property means that manipulation of a solids by moving them in space or changing via Boolean operations must provide other valid solids;
5. Boundary determinism. The boundary of a solid must contain the solid and hence must determine distinctively the interior of the solid.

4.5 AN OVERVIEW OF SOLID MODELING METHODS

A number of different schemes have been used to represent solids. The overview below briefly describes the following schemes: constructive solid geometry (CSG), and boundary representation. Such techniques as primitive instancing, cell decomposition, sweep representations also exist.

These methods are not mutually exclusive; in fact, currently most commercial solid modelers, such as AutoCAD use constructive solid geometry together with boundary representation. Also, AutoCAD uses sweep representations to construct primitives for use with the CSG operations.

4.5.1 BOUNDARY REPRESENTATION (B-REP)

A solid may be defined by providing a representation of its boundary (with the solid occurring inside this boundary). The boundary may be an approximate one, made up of faces (faceted boundary), or it may be defined precisely by formulas. For example, the equation for the surface of a sphere provides an exact boundary representation of a sphere.

A B-rep model or boundary model is based on the topological notion that a physical object is bounded by a set of faces. These faces are regions or subsets of **closed and orientable surfaces**. A closed surface is one that is continuous without breaks. An orientable surface is one in which it is possible to distinguish two sides by using the direction of the surface normal to point to the inside or outside of the solid model under construction. Each face is bounded by edges and each edge is bounded by vertices. Thus, topologically, a boundary model of an object is comprised of faces, edges, and vertices of the object linked together in such a way as to ensure the topological consistency of the model.

The database of a boundary model contains both its topology and geometry. Topology is created by performing Euler operations and geometry is created by performing euclidean calculations. Euler operations are used to create, manipulate, and edit the faces, edges, and vertices of a boundary model as the set (boolean) operations create, manipulate, and edit primitives of CSG models. Euler operators, as boolean operators, ensure the integrity of boundary models. They offer a mechanism to check the validity of these models. Other validity checks may be used as well. Geometry includes coordinates of vertices and metric information such as distances, angles, areas, volumes and inertia tensors. It should be noted that topology and geometry are interrelated and cannot be separated entirely. Both must be compatible otherwise a nonsense object could result.

While B-rep systems store only the bounding surfaces of the solid, it is still possible to compute volumetric properties such as mass properties (assuming uniform density) by virtue of the Gauss divergence theorem which relates volume integrals to surface ones. The speed and accuracy of these calculations depend on the types of surfaces used by the models.

The desired properties of a representation scheme discussed above apply to B-rep schemes. These schemes are unambiguous if faces are represented unambiguously, that is, as regions of closed orientable surfaces. The validity of B-rep models is ensured via Euler operations which can be built into the syntax of a CAD/CAM system. However, these models are not unique because the boundary of any object can be divided into faces, edges, and vertices in many ways. Verification of uniqueness of boundary models is computationally expensive and is not performed in practice.

Objects that are often encountered in engineering applications can be classified as either polyhedral or curved objects. A polyhedral object (plane-faced polyhedron) consists of planar faces (or sides) connected at straight (linear) edges which, in turn, are connected at vertices. A cube or a tetrahedron is an obvious example. A curved object (curved polyhedron) is similar to a polyhedral object but with curved faces and edges instead. The identification of faces, edges, and vertices for curved closed objects such as a sphere or a cylinder needs careful attention, as will be seen later in this section. Polyhedral objects are simpler to deal with and are covered first.

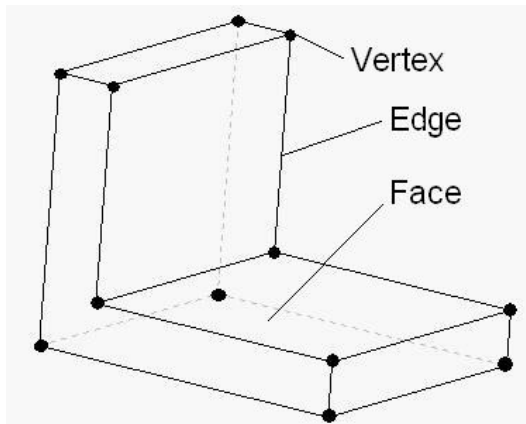
It could be stated that the primitives of a B-rep scheme are faces, edges, and vertices. In addition to this however the following two questions must be answered:

- first, what is a face, edge, or a vertex?
- second, knowing the answer to the first question, how can we know that when we combine these primitives we would create valid objects?

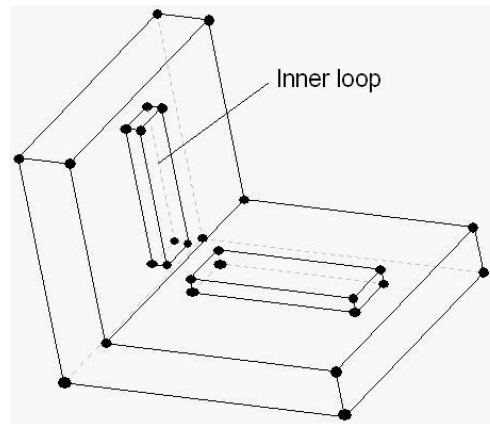
Answers to these questions can help users to create B-rep solid models of objects successfully. To show that these answers are not always simple, consider the polyhedral objects shown in Fig. 4.16. Polyhedral objects can be classified into four classes. The first class (Fig. 4.16a) is the simple polyhedra. These do not have holes (through or not through) and each face is bounded by a single set of connected edges, that is, bounded by one loop of edges. The second class (Fig. 4.16b) is similar to the first with the exception that a face may be bounded by more than one loop of edges (inner loops are sometimes called rings). The third class (Fig. 4.16c) includes objects with holes that do not go through the entire object. For this class, a hole may have a face coincident with the object boundary; in this case we call it a boundary hole. On the other hand, if it is an interior hole (as a void or crack inside the object), it has no faces on the boundary. The fourth and the last class (Fig. 4.16d) includes objects that have holes going through the entire objects. Topologically, these through holes are called handles.

Let us further define the primitives of a B-rep scheme and other related topological items that enable a user to create the boundary model of an object. They apply to both polyhedral and curved objects.

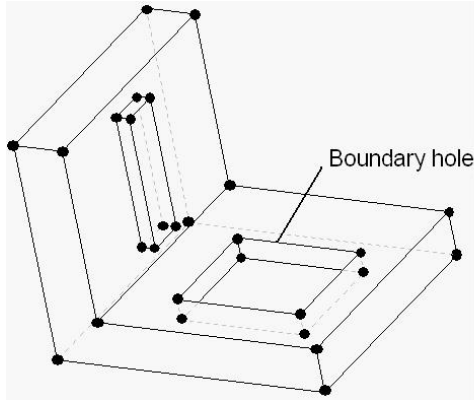
- a **vertex** is an unique point (an ordered triplet) in space;
- an **edge** is a finite, non-self-intersecting, directed space curve bounded by two vertices that are not necessarily distinct;



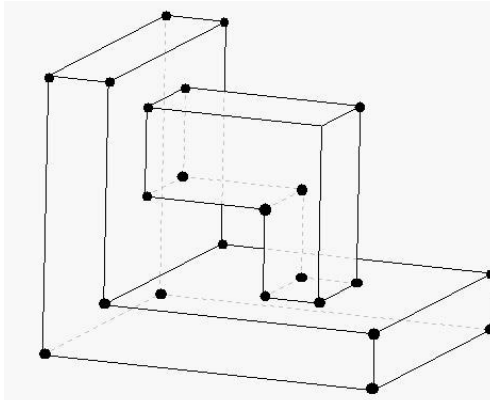
a. Simple polyhedra



b. Polyhedra with faces of inner loops



c. Polyhedra with no through holes



d. Handles (through holes)

Figure 4.16

— **a face** is defined as a finite connected, non-self-intersecting, region of a closed oriented surface bounded by one or more loops. At each point on the face, there is a surface normal N that has a sign associated with it to indicate whether it points into or away from the solid interior. One convention is to assume N positive if it points away from the solid. It is desirable, but not required, that a face has a constant surface normal;

— **a loop** is an ordered alternating sequence of vertices and edges. A loop defines a non-self-intersecting, piecewise, closed space curve, which in turn, may be a boundary of a face. In Fig. 4.16a, each face has one loop while the top and the right side faces of the object shown in Fig. 4.16b have two loops each (one inner and one outer);

— **a ‘not’ through hole** is defined as a depression in a face of an object;

— **a handle** (or through hole) is defined as a passageway that pierces the object completely. The topological name for the number of handles in an object is genus;

— **a body** (sometimes called a shell) is a set of faces that bound a single connected closed volume. Thus a body is an entity that has faces, edges, and vertices. Such an entity may be a useful solid or an intermediate polyhedron. A minimum body is a point. Topologically this body has one face, one vertex, and no edges. It is called a seminal or singular body.

Having defined the boundary model primitives, we now return to the question of how they can be combined to generate topologically valid models. Euler (in 1752) proved that polyhedra that are homo-morphic to a sphere (i.e., their faces are non-self-intersecting and belong to closed orientable surfaces) are topologically valid if they satisfy the following equation:

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

The equation above relates the number of:

V - vertices; E - edges; F - faces; L - faces' inner loops; B - bodies; G - genus (handles or through holes) of the object.

Table 4.3 shows the counts of the various variables of the above equation for polyhedra shown in Fig. 4.16. The numbering of these polyhedra in the table is taken from left to right and top to bottom with the top left cube being polyhedron number 1 and the bottom right object being number 9.

Table 4.3

Object	F	E	V	L	B	G
a	8	18	12	0	1	0
b	18	42	28	2	1	0
c	18	42	28	2	1	0
d	14	36	24	2	1	1

Further on, curved objects such as cylinders and spheres must be considered. The same rules and guidelines for boundary modeling discussed thus far for the former objects apply to the latter. The major difference between the two types of objects results if closed curved edges or faces exist. Consider, for example, the closed cylinder shown in Fig. 4.17. As shown in Fig. 4.17, a closed

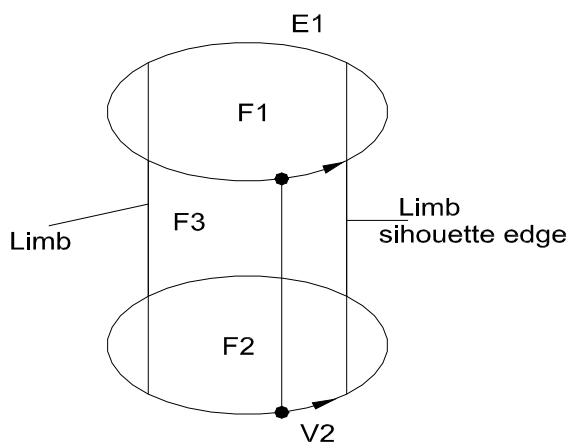


Figure 4.17

cylindrical face (and alike) has one edge and two vertices and a spherical face (and alike) has one vertex and no edges. The boundary model of a cylinder has three faces (top, bottom, and cylindrical face itself), two vertices, and three edges connecting the two vertices. The other 'edges' are for visualization purposes. They are called limbs, virtual edges, or silhouette edges. The boundary model of a sphere, on the other hand, consists of one face, one vertex, and no edges. Notice that both models satisfy Euler laws $F - E + V = 2$ for simple polyhedra.

The representation of curved edges is more complex than representing piecewise linear edges. There are direct and indirect schemes. In direct schemes, an edge is represented by a curve equation and ordered endpoints. In indirect

schemes, the edge is represented by the intersection of two surfaces. In practice, indirect schemes are probably preferred because the intersection of two underlying surfaces of two faces produces the curved edge of the two faces.

If the curved objects are represented by storing the equations of the underlying curves and surfaces of the object edges and faces respectively, the resulting boundary scheme is known as an exact B-rep scheme. Another alternative is the approximate or faceted B-rep (sometimes called tessellation rep). In this scheme, any curved face is divided into planar facets—hence the name faceted B-rep. The faceted cylinder for example is generated by rotating a line incrementally about the cylinder axis the desired total number of facets. This is accomplished via a rotational sweep operator. A faceted sphere is formed in a similar way by rotating m connected line segments (edges) about the sphere axis for a total of n sides.

4.5.2 CONSTRUCTIVE SOLID GEOMETRY (CSG)

Objects are built up with CSG by starting with simple objects called primitives and combining them with Boolean operations. A CSG model is based on the topological notion that a physical object can be divided into a set of primitives (basic elements or shapes) that can be combined in a certain order following a set of rules (Boolean operations) to form a object. Primitives themselves are considered valid CSG models. Each primitive is bounded by a set of surfaces; usually closed and orientable. The primitive's surfaces are combined via a boundary evaluation process to form the boundary of the object, that is, to find its faces, edges, and vertices. In addition to degenerating an object to a collection of primitives, a CSG model is fundamentally and topologically different from a B-rep model in that the former does not store explicitly the faces, edges, and vertices. Instead, it evaluates them whenever they are needed by applications' algorithms, e.g., generation of line drawings.

A CSG solid can be constructed using succession of activities and following three simple steps (refer also to the practical example given in Table 4.4):

- create a number of solid primitives;
- locate and orient primitives in accordance to each other by moving in three dimensions and rotating them about constructive axes;
- apply Boolean operation to add, subtract or intersect primitives in order to obtain the final shape.

While solid models are complete and unambiguous, they are not unique. An object may be constructed in various ways. The above mentioned three steps could be applied in different order with the same result. Regardless of the order and method of construction as well as the representation scheme utilized, the resulting solid model of the object is always complete and unambiguous. However, there will always be a more efficient way than others to construct the solid models as in the case with wireframe and surface models.

Most commercially available solid modeling packages have a CSG-compatible user input and therefore provide users with a certain set of building

blocks, often called primitives. Primitives are simple basic shapes and are considered the solid modeling entities which can be combined by a mathematical set of Boolean operations to create the solid. Primitives themselves are considered valid 'off-the-shelf' solids. In addition, some packages, especially those that support sweeping operations, permit users to utilize wireframe entities to create faces that are swept later to create solids. The user usually positions primitives as required before applying Boolean operations to construct the final solid.

There is a wide variety of primitives available commercially to users. However, the four most commonly used are the block, cylinder, cone, and sphere. These are based on the four natural quadrics: planes, cylinders, cones, and spheres. For example, the block is formed by intersecting six planes. These quadrics are considered natural because they represent the most commonly occurring surfaces in mechanical design which can be produced by rolling, turning, milling, cutting, drilling, and other machining operations used in industry. Planar surfaces result from rolling, chamfering, and milling; cylindrical surfaces from turning or filleting; spherical surfaces from cutting with a ball-end cutting tool; conical surfaces from turning as well as from drill tips and countersinks. Natural quadrics are distinguished by the fact that they are combinations of linear motion and rotation. Other surfaces, except the torus, require at least dual axis control.

From a user-input point of view and regardless of specific system syntax, primitive requires a set of location data, a set of geometric data, and a set of orientation data to define it completely. Location data entails a primitive local coordinate system and an input point defining its origin. Geometrical data differs from one primitive to another and are user-input. Orientation data is typically used to orient primitives properly relative to some Current Coordinate System or World Coordinate System (CCS or WCS) of the solid model under construction. Again, primitives are usually translated and/or rotated to position and orient them properly before applying Boolean operations.

Two or more primitives can be combined to form the desired solid. To ensure the validity of the resulting solid, the allowed combinatorial relationships between primitives are achieved via Boolean (or set) operations. The available Boolean operators are union ($\cup$ or +), intersection ($\cap$ or I), and difference (-). The union operator is used to combine or add together two objects or primitives. Intersecting two primitives gives a shape equal to their common volume. The difference operator is used to subtract one object from the other and results in a shape equal to the difference in their volumes.

Figure 4.18 shows some Boolean operations between two sets as follows:

The union of two sets $P \cup Q$ is subset of elements of W that are members either P or Q .

The intersection of two sets $P \cap Q$ is subset of W that are simultaneously element of both P and Q .

The difference of two sets $P - Q$ is the subset of elements of W that belong to P and not to Q .

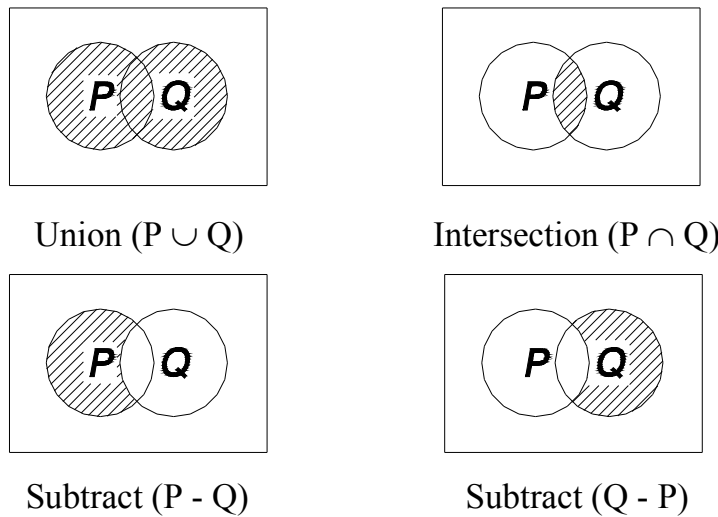


Figure 4.18

Questions:

1. What are the advantages and disadvantages of wireframe models?
2. What wireframe models are used for?
3. What is surface object?
4. List some applications of the surface models.
5. What are extruded surface and surface of revolution?
6. What are swept and ruled surfaces?
7. What is spatial addressability.
8. Give an example for an object topology.
9. What is homogeneous three-dimensionality?
10. What is Finiteness and finite describability?
11. What is closed and orientable surface?
12. What are vertex and edge?
13. What is a face?
14. What is a loop?
15. Write the Euler formula and explain the elements.
16. Prove the validity of the Euler formula for some of the polyhedral objects shown in figure 4.16
17. Prove the validity of the Euler formula for a cylinder.
18. Explain the steps to construct CSG object.
19. Draw a figure and explain the Boolean operations

5. FEATURE BASED MODELLING

5.1 INTRODUCTION

One of the basic skills of an expert designer is an ability to manipulate geometric concepts. The packaging of an aircraft's undercarriage mechanism into a compartment in the wing, the recognition of a stress-raising feature likely to lead to component failure, and the development of a plan of the machining operations to manufacture a product all require a very sophisticated geometric reasoning ability. Going back to CAD/CAM applications, any truly expert computer program will have to emulate such ability in order to bring the system close to the engineering activities.

Part of the difficulty in geometric interpretation is that the methods used for modeling of geometry in CAD are not semantically very rich. There is no information, for example, to say that a collection of lines and arcs, or of cylinders and cones, represents a drilled and tapped hole. A trained human observer can, nevertheless, interpret such features on a drawing without difficulty. A route that is currently being explored in CAD research is to represent components in terms of higher-level entities that do have some engineering meaning. Such entities are called **features**.

5.1.1 WHAT IS A FEATURE?

There are almost as many definitions of features as there are workers in the features research field! Originally, features were thought of almost exclusively in their geometric sense. Shapes such as drilled holes, ribs or bosses in castings, grooves in shafts and so on were regarded as typical features. In addition to a geometric meaning, they also had some engineering meaning, and in fact in some work almost any attribute of a component (or part of a component) came to be regarded as a feature. We shall mention Brown's definition (1992):

a feature is any perceived geometric or functional element or property of an object useful in understanding the function, behavior or performance of that object.

Some of the early work in features came from the area of process planning, which is the procedure whereby a plan is determined for the sequence of operations required to manufacture a component. In order to automate this task, it is necessary for computers to be able to recognize features on a component that may be produced by given machining operations. More recently, features have been used as a basis for a number of design activities, including the rapid design of components using standard shape features, and assistance with the interface between CAD and analysis and manufacture.

Although the definition of the feature could seem some philosophical, there are some underlying practical meanings which are listed bellow:

Features are the building blocks of your design. Every geometric model consists of number of features, starting with a base (boss) feature and adding additional ones as required;

Features are mainly three dimensional, although 2D sketches can also be considered features;

Features are named objects and each has a unique editable name given by the system or by the user;

Features preserve their initial geometry and topological information despite the subsequent model changes which information could be edited on any stage of the model creation;

Features form a hierarchical dependency. Earlier created features can be a base for adding new ones. In this case editing (deleting) a given feature can affect the dependent ones.

5.2 DESIGN BY FEATURES

According to the latest developments features can include industry standard objects like holes, chamfers, and fillets. The designer can position these features precisely with parametric dimensions and geometric constraints. In addition, the designer can sketch own features as elements of the part design.

Thus most of modern CAD/CAM systems adopted the method where user may assemble a representation of an object as a collection of features, by either adding or subtracting features from the model. The method is therefore sometimes called 'constructional design by features'. A number of programs have been developed along these lines. Some are very closely integrated with solid modelers and some less so, but are nevertheless able to translate the feature representation into solids. A variety of underlying representations are used, encompassing the all major solid modeling approaches – B-rep, CSG and dual representation.

5.2.1 TYPES OF FEATURES

Any feature based CAD system recognizes some major types of features:

1. **Sketch features.** Features based on sketch geometry – a planar closed regions or open succession of wires. They are two basic types:
 - ♦ *profile sketches* allowing creation of 3D objects with arbitrary shapes based on planar closed regions – profiles. To such profiles can be added 3-th dimension via one of the following methods:

- extruding in third dimension;
 - sweeping along the 2D or 3D path;
 - revolving about an arbitrary axis;
 - lofting.
- ♦ *cutting line sketches* - used to define cross sections for drawing views.
- 2. Placed features.**
- holes, fillets; chamfers, surfcuts shells.
- 3. Standard parts.**
- bearings, standard holes keys shaft generators gear generators.
- 4. Construction features (reference geometry)**
- work planes;
 - work axes;
 - work points.

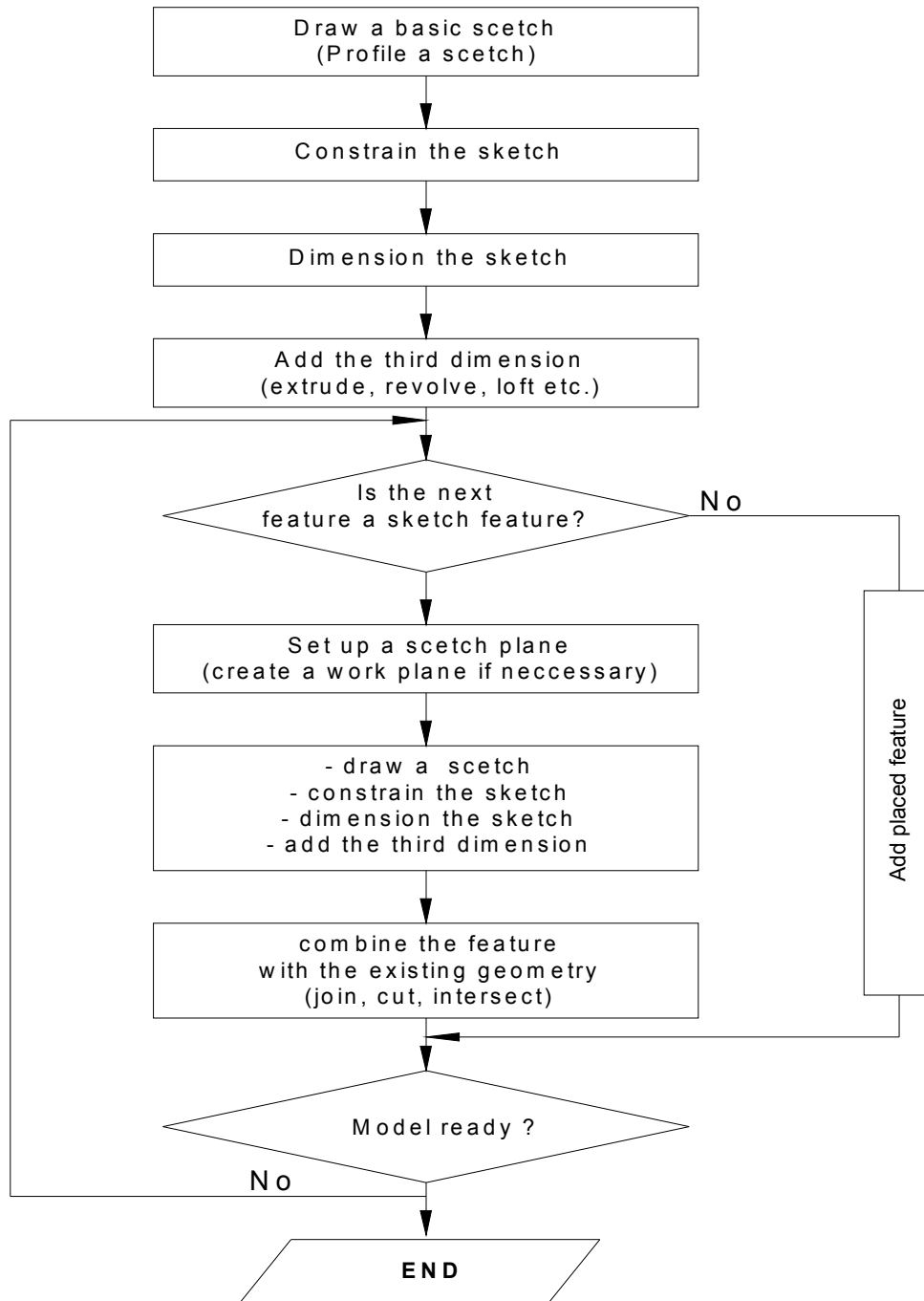
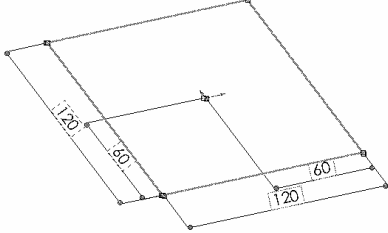
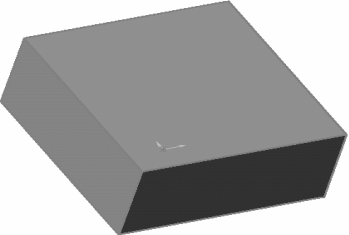
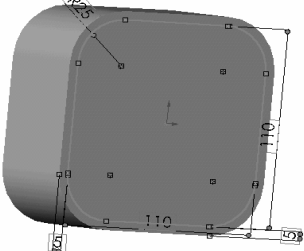
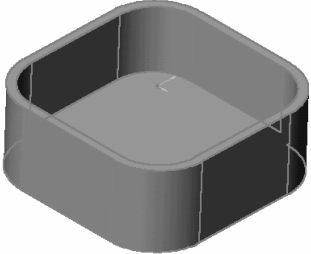


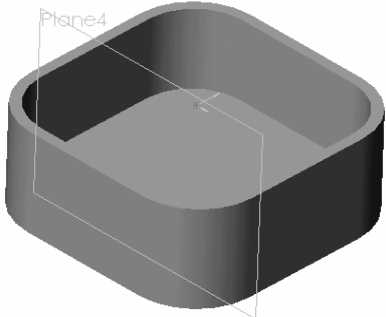
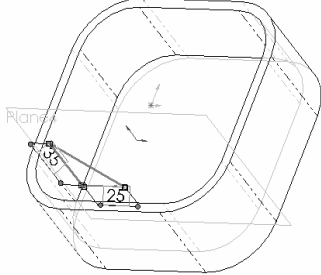
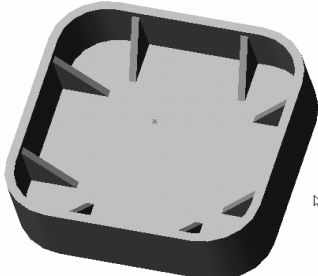
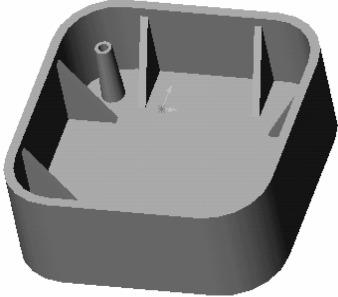
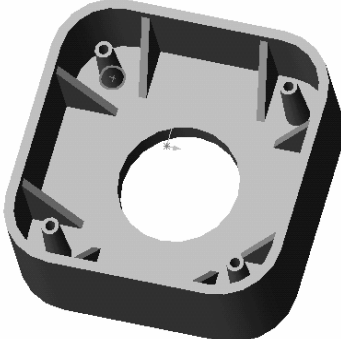
Figure 5.1

As it was said above, the feature based parametric design is quite straightforward and takes the generic form given on the flowchart above (Fig. 5.1).

An illustration of the construction of a component using features is shown in Figure 5.1 – a plastic injection molding that has been defined using instances of features of six basic types.

Table 5.1

OPERATION	RESULT
Create the base sketch on the default sketch plane. This sketch must be as simple as possible	
Add the third dimension (to form a 3D basic feature) by extruding the sketch in positive direction	
Add placed features – fillets to round the vertical edges	
Select the upper base as a sketch plane. Add a second sketch on a sketch plane. This sketch must be dimensioned and positioned with the accordance to the base feature.	
Extrude the second sketch towards the solid basic feature with operation 'cut' in order to produce the walls	

Define a construction feature (reference geometry) – a sketch plane for placing the third sketch – rib. This sketch plane is offset from the wall a certain distance	
Place the third sketch representing the rib on the sketch plane. This sketch is fixed according to the existing geometry	
Extruding the rib using the 'join' operation at midplane and multiplying it in a circular pattern	
Adding the boss with a hole by using the following procedure: - select the bottom of the box as a sketch plane; - draw a new sketch – circle; - dimension the sketch; - extrude the sketch at certain distance with a given tapering angle using the 'join' operation; - apply the 'hole' operation at the upper basis of the cone.	
Multiplying it the boss with a hole in a circular pattern and applying the 'hole' operation in the middle of the bottom	

5.3 CREATING SKETCH GEOMETRY

As it is clear from the examples given above, each sketch type is created similarly. First, you create sketch geometry using available for a given CAD system drawing technique. Then the sketch has to be converted to a parametric system whose dimensional values determine changes in part geometry. Some systems perform the conversion automatically when a closed region is created.

There are two essential recommended practices to be kept in mind while creating a sketch:

- ◆ Study the part and determine how to keep your sketches simple. Simple sketches are not only easier to create; they provide more control over the part when you need to modify them. Drawing a complex sketch may reduce the number of sketches you need to create. However, a few simple sketches are easier to draw and they give you more ways to manipulate the model. To keep your sketches simple, look for ways to decompose the part into its simplest shapes. Also, consider how you can create a complex feature by creating a simple sketch and manipulating the resultant feature by arraying it into a more complex form;

- ◆ Draw sketches to be roughly correct in size and shape. To do this, you can use the coordinate snapping and polar tracking modes where provided. Also consider drawing a rectangle that serves as a boundary for the size of the base feature. Then, you can create your sketch within the bounds of this rectangle. This is a useful technique because it sets the relative size for the base feature and, therefore, for features that are added afterward. After creating the sketch, delete the rectangle.

5.3.1 DEFINING SKETCH CONSTRAINTS

Completing the definition of the sketch is a very important modeling step. Because the system must control and maintain parametric relationships within the part, it should fully understand each feature and how potential changes can affect it. In other words, the ability of the systems to accurately maintain these relationships depends upon its ability to predict how features can legitimately change.

The purpose of constraining a sketch is to define how a sketch can change shape or size. These ways are known as the sketch's **degrees of freedom (DOF)**. Constraining a sketch means that you are defining the acceptable ways in which a feature can exist or change. Because constraints can be fixed or variable, you can control the amount and type of change that a feature can experience. This type of control tells the system what changes are acceptable and prevents it from making unwanted changes to a feature as you modify your part.

A simplest example could be given considering the degrees of freedom for a circle and an arc. A circle has maximum three degrees of freedom, the location of its center (2 DOF) and its radius (although if the circle is a basic feature the exact position of the centre could not be required). If these attributes are

defined, then the circle is fully understood and those values can be maintained. Similarly, an arc has seven degrees of freedom – positions of its center, and the endpoints of the arc segment as well as its radius.

Many of the CAD systems utilize means to graphically visualize the sketch DOF. Figure 5.2a shows a freely located circle with all degrees of freedom.

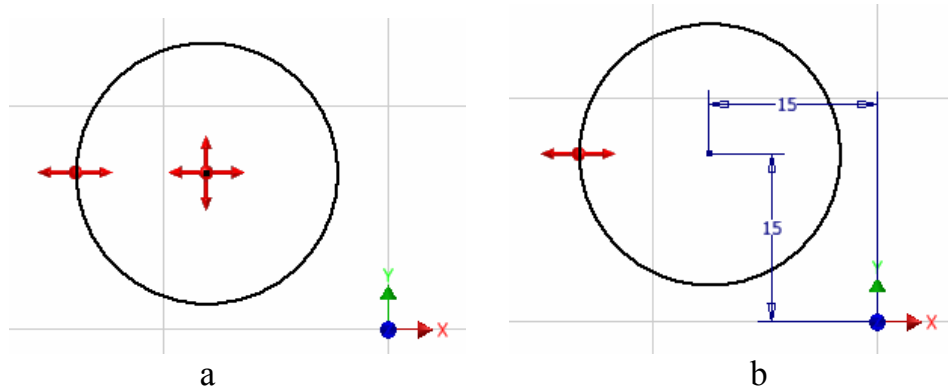


Figure 5.2

As you constrain the center of the circle in respect to the coordinate system origin by adding dimensions, the number of DOF is reduced (Figure 5.2 b). Adding the value for the radius will fully constrain the sketch.

The number of degrees of freedom defined determines the degree to which the sketch is constrained. If one defines all of the attributes for an arc as specified above, the arc is **fully constrained**. Defining a subset of these attributes causes the sketch to be **underconstrained**, while defining the attributes in more than one way causes the sketch to be **overconstrained**.

The goal when constraining a sketch is to fully constrain it. You can underconstrain your sketches but this is not recommended because an underconstrained sketch causes the resulting feature to be unpredictable. This creates the potential for unexpected changes to occur to your part. Most of the systems will not let you overconstrain a sketch because it results in a potentially ambiguous definition of the feature. If you attempt to overconstrain a sketch, the system displays an error message and ignores the constraint.

There are at least two types of constraints, **geometric** and **dimensional**. Geometric constraints define the shape and the interrelationships of the sketch geometry. This is the type of constraint that could be applied automatically to the sketch when it is converted to a profile or a path.

Dimensional constraints specify the size and placement of sketch geometry. Because of their parametric nature, dimensions are very powerful and can be defined in a number of ways.

Before constraining it, the designer needs to study the sketch and consider the following:

- ◆ Determine the degrees of freedom that your sketch contains and the interrelationships of its geometric elements.

- ◆ Identify the geometric constraints that the system has already assumed and which degrees of freedom they define.
- ◆ Decide if these geometric constraints need to be added to or modified to properly represent the shape and relationships of the geometric elements.
- ◆ Identify the remaining degrees of freedom that need to be constrained.
- ◆ Decide the type of constraints needed to define the remaining degrees of freedom. You will need to use a combination of geometric constraints and dimensions to complete the sketch definition.

Usually, there are many different combinations or techniques one can use to do this. The designer should choose one that is simple but that properly represents the relationship of the geometric elements. The power of parametric design lies in the ability to create variable definitions that relate geometric elements through expressions or equations. This capability makes control and maintenance scalable as well as flexible since these parametric relationships can be applied not only to single parts but to part families as well.

5.3.2 APPLYING GEOMETRIC CONSTRAINTS

When constraining a sketch it is best to begin by defining its overall shape before defining its size. Geometric constraints help determine the shape of the sketch as they specify the orientation and relationship of its geometric elements. Geometric constraints that specify orientation indicate whether an element is horizontal or vertical. Geometric constraints that determine relationships specify whether two elements are perpendicular, parallel, tangent, collinear, concentric, projected together, joined, have the same *X* or *Y* coordinate location, or have the same radius.

Different CAD systems utilize different ways to display geometric constraints. In and case however, the user has full control over them in terms of adding, deleting or modifying these constraints for the optimum results.

When applying geometric constraints, keep the following recommended practices in mind:

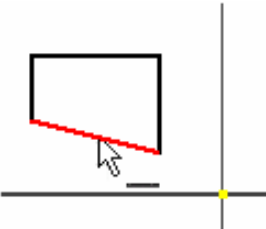
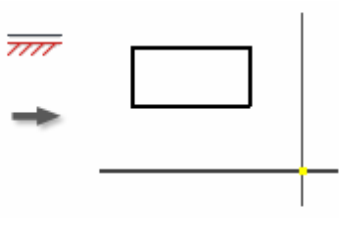
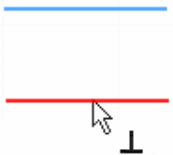
- ◆ Determine the geometric constraints your sketch needs. Again, the key to defining geometric constraints is to study your sketch and determine how you want the elements to interrelate.
- ◆ Review the geometric constraints the system determined when it translated your sketch into a profile or path. Make sure these constraints are consistent with the ones you predetermined. If they are not, you need to delete the ones that do not apply.
- ◆ Add any necessary, supplemental geometric constraints. Add the minimal number of geometric constraints you need to define the shape. Since they often define more than one degree of freedom, your sketch needs fewer geometric constraints than it has degrees of freedom.

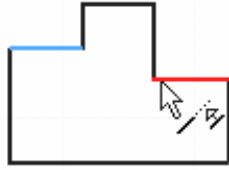
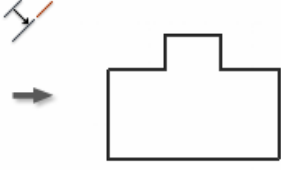
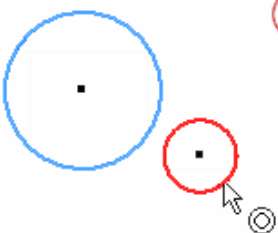
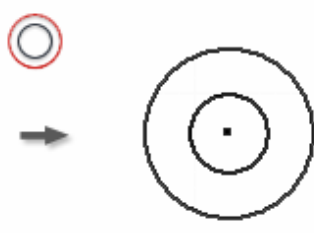
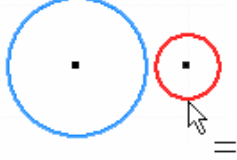
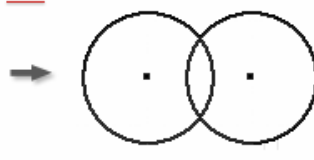
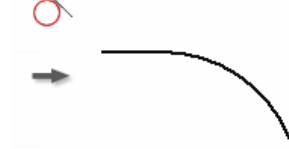
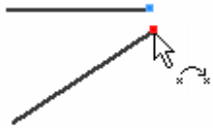
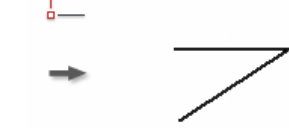
◆ Consider substituting dimensional constraints for geometric constraints when the shape of the sketch becomes misshapen or contorted. When constraining your sketch, you need to select a proper combination of geometric and dimensional constraints. You cannot rely on one type of constraint to define your sketch.

As additional geometric constraints are added, some combinations of these constraints may cause the sketch to become twisted so lines overlap. When the designer fixes points around the sketch, he may be forcing the sketch to move. Since it moves in areas that are not fixed, this movement can force the sketch into an improper shape. When this happens, the last assigned geometric constraint must be deleted. Then one good approach is either to try another combination of geometric constraints or to use a dimensional rather than a geometric one. Depending on the sketch, a dimensional constraint may not contort its shape.

Table 5.2 gives examples for typical geometric constraints.

Table 5.2

Constraint	Before applying the constraint	After applying the constraint
HORIZONTAL The horizontal constraint causes lines, ellipse axes, or pairs of points to lie parallel to the X axis of the sketch coordinate system. VERTICAL The vertical constraint causes lines, ellipse axes, or pairs of points to lie parallel to the Y axis of the sketch coordinate system.		
PERPENDICULAR The perpendicular constraint causes selected curves or ellipse axes to lie at right angles to one another.		
PARALLEL The parallel constraint causes two or more lines or ellipse axes to be constrained parallel to one another.		

Collinear Allows two line segments or ellipse axes to fall on the same line.		
Concentric The concentric constraint causes two arcs, circles, or ellipses to be constrained to the same center point. The result is the same as a coincident constraint applied to the center points of the curves.		
Equal The equal radius or length constraint resizes selected arcs and circles to the same radius, or selected lines to the same length.		
Tangent The tangent constraint causes two curves to be tangent to one another, even if they do not physically share a point.		
Coincident The coincident constraint causes two points to be constrained together or one point to a curve.		

5.3.3 APPLYING DIMENSIONAL CONSTRAINTS

After defining the shape of the sketch, the size must also be specified. Dimensional constraints, otherwise known as dimensions, tell the system such measurements as length, radius, or rotation angle of geometric elements within the sketch. Unlike geometric constraints, dimensions are parametric. This means that they are variable and that changing their values causes the geometry to change. This gives the designer the ability to have full control over the model dimensions, which could be changed at any time during the design process.

Dimensions can be specified as **numeric** constants or as **equations**. Although one can use them interchangeably, they each have specific uses. Numeric constants are useful when a geometric element is independent. That is, its size is static and is not related to any other geometric element. For example, in the Fig. 5.3a, all of the lines and the angle are constant and are stated as numeric values.

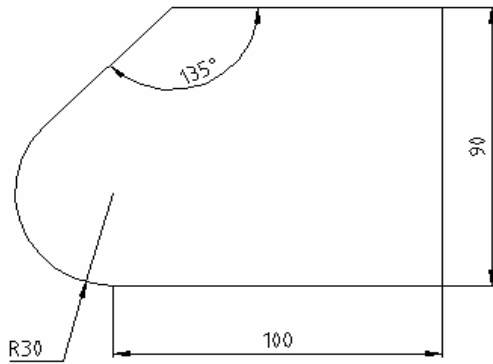


Figure 5.3a

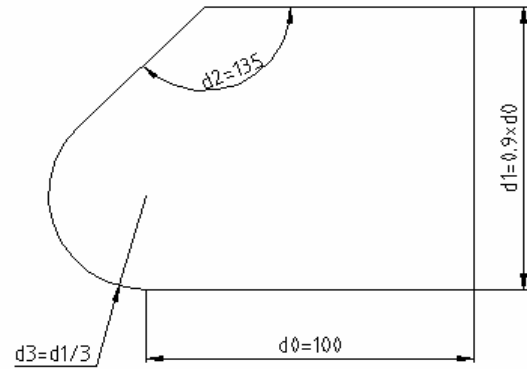


Figure 5.3b

In the second illustration, (Fig. 5.3b) the dimensions are expressed as equations. Equations are useful when the size of a geometric element is dependent upon the size of another element. In the particular case changing the $D0$ dimension will result in changing of $D1$ and $D3$ dimensions. The height of the sketch needs to be proportional to the length by remaining $9/10$ of the length. To maintain this relationship, the dimension for the vertical line is defined as an equation of $D1 = 0.9 * D0$ where $D1$ is the parameter name for the vertical line and $D0$ is the parameter name for one of the horizontal lines.

The designer can use complicated formulas in order to calculate and specify dependant dimensions.

When applying dimensional constraints, keep the following recommended practices in mind:

- ◆ Determine the dimensions that your sketch needs and how they should be expressed. Remember, before constraining your sketch you should fully study it and determine the geometric and dimensional constraints that it needs. When identifying the dimensions that are needed, consider how the sketch elements interrelate within the sketch or how they relate to other features. These relationships help to determine the way you should express the dimension (as a constant or an equation).

- ◆ Determine the order in which to apply the dimensions. When applying dimensions, it is recommended that you use a refinement technique. To minimize distortion of the sketch first define larger elements that have an overall bearing on the size of the sketch. Then, define the smaller elements to refine the size of the sketch.

- ◆ Delete or undo a dimension when the shape of the sketch becomes twisted. Since a sketch can move in areas where it is not fixed, this movement can force the sketch into the wrong shape. When this happens, delete the last dimension you assigned. Then either try another combination of dimensions or use a geometric constraint rather than a dimensional one. Depending on the sketch, a geometric constraint may not contort its shape.

5.4 ADDING THE THIRD DIMENSION

In order to add third dimension to the 2D sketch there must have at least one valid, in general closed fully constrained sketch. Some principle of modeling require several sketches to perform the operation. There are mainly four methods to construct a 3D feature:

Extrusion: This tool adds volume perpendicular to a 2D sketch (or sweeps the planar sketch along the vector perpendicular to the sketch plane. You can extrude in the both directions form the sketch plane or symmetrically. There are usually two methods to determine the height of extrusion:

- Specify the distance of extrusion (Figure 5.4);

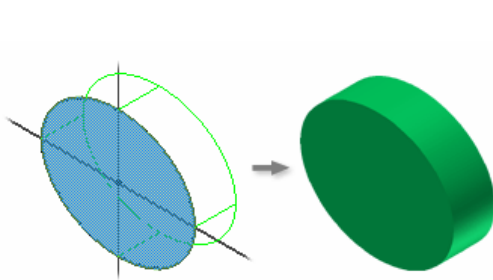


Figure 5.4

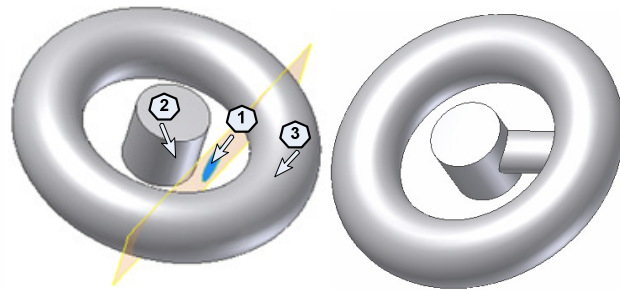


Figure 5.5

- Select a geometry for starting or/and ending the extrusion. Such geometry could be points, coordinate and work planes, vertexes, faces (Figure 5.5). Here position 1 represents a profile to be extruded).

Revolution: Revolved features are created by sweeping one or more sketched profiles around an axis. The axis lies in the sketch plane and must not intersect the profile, but could be part of the contour. You can revolve in the both directions form the sketch plane or symmetrically. There are usually two methods to determine the angle of revolution:

- Specify the angle of revolution as a value (Figure 5.6);

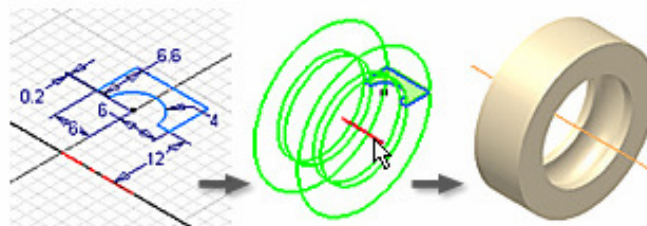


Figure 5.6

- Select a geometry for starting or/and ending the revolution. Such geometry could be points, coordinate and work planes, vertexes, faces (Figure 5.7).

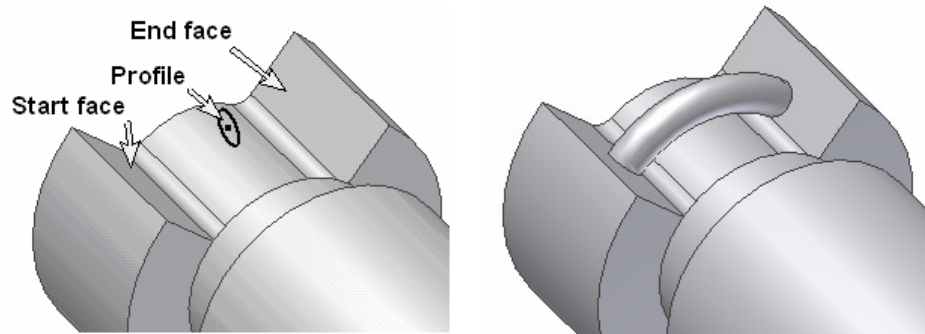


Figure 5.7

Sweeping: Sweep features or bodies are created by moving or sweeping one or more sketch profiles along a path. The path can be an open or closed loop, but must pierce the profile plane. In this case at least two valid sketches are required – one for the profile and another for the path (Figure 5.8).

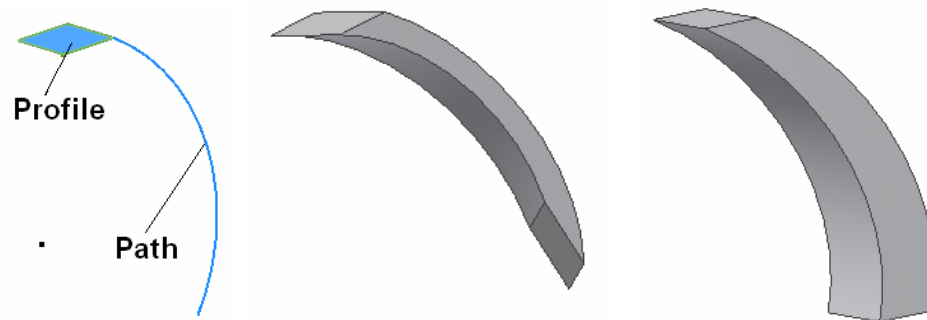


Figure 5.8

Lofting: Create lofted features or bodies by blending multiple profiles, called sections, and transitioning them into smooth shapes between the profiles or part faces. In addition to sketched profiles, you can also select object faces and points to include as loft sections. This tool is used to create complex, organic shapes such as those found in the automotive, marine, and consumer products industries (Figure 5.9).

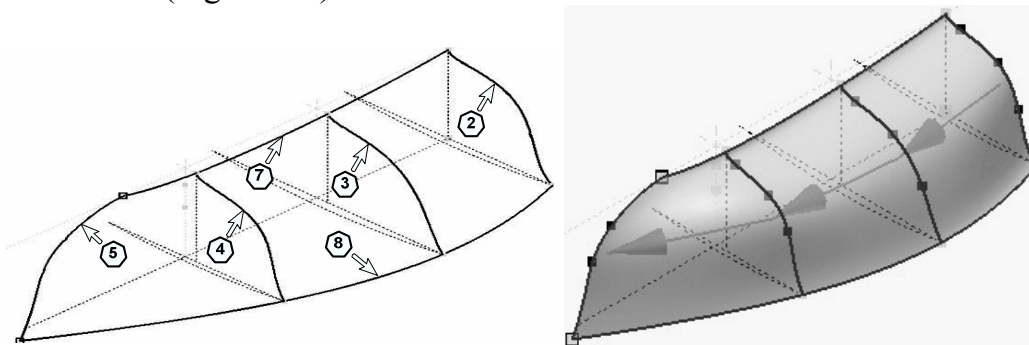


Figure 5.9

5.4 CONSTRUCTION FEATURES

In addition to the sketched, placed and standard features, the so-called Construction features (sometimes referred as Reference Geometry) are provided in order to specify sketch planes for different parts of a model.

As each successive sketched feature must be placed on a planar surface, sometimes the model geometry requires creating planes (or workplanes), which placement and orientation are to suit particular design needs. Table 5.1 gives a typical example for a sketch plane which is parallel to a given planar face and offset from it a given distance.

The users are usually provided with the abilities to place sketch planes as follows:

- on XY , YZ or ZX of the World Coordinate System (the principle coordinate system for a current drawing);
- on XY plane of a particular user created coordinate system (User Coordinate System);
- on existing plane part of the model surface;
- on a existing work plane (reference plane) created by the user.



Figure 5.11

Workplanes or reference planes can be created using different techniques and they are integrated part of the model. Deleting the workplane will delete the geometry based on it. Figure 5.11 gives illustration for some cases. In particular reference planes could be created:

- parallel to a plane or face, offset by a specified distance;
- through an edge, axis, or sketch line at an angle to a face or plane;
- through three points;
- through a point parallel to a plane or face;
- through an edge, axis, or sketch line and a point;
- through a point and perpendicular to an edge, axis, or curve;
- on a non-planar face or angular surface.

Figure 5.5 shows a workplane created at a particular angle in order to add a cylindrical branch to the already created cylinder. The result from the operation is given on Fig. 5.12

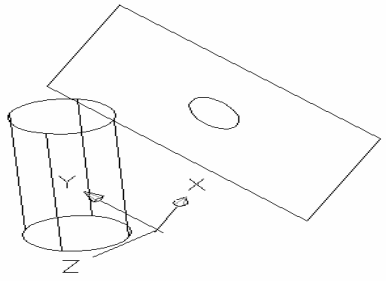


Figure 5.12

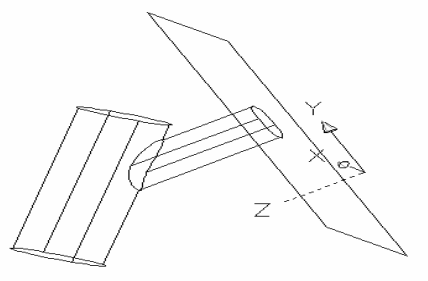


Figure 5.13

Questions:

1. What are the practical properties of the features?
2. Describe the types of features.
3. Draw the diagram illustrating the design process by features.
4. What are the states of the sketch in respect to its degrees of freedom?
5. You can build 3D features from:
 - a/ Overconstrained and underconstrained sketches
 - b/ Fully constrained and underconstrained sketches
 - c/ Fully constrained sketches
 - d/ Fully constrained and overconstrained sketches
6. Explain the horizontal and concentric constraints. Give graphical examples.
7. Explain the vertical and parallel constraints. Give graphical examples.
8. Explain the equal and tangent constraints. Give graphical examples.
9. Explain the perpendicular and collinear constraints. Give graphical examples.
10. Explain methods for determining the height of extrusion.
11. Sweep tool – give a definition.
12. Where the user can place a planar sketch?
13. List some possibilities to create a work plane.

6. ASSEMBLY MODELING

6.1 INTRODUCTION

In most engineering designs, the product of interest is a composition of parts, formed into assembly.

An assembly is a collection of parts, positions and orientation between which are determined by engineering principles. It is important to understand the nature and the structure of dependencies between parts in an assembly to be able to model the assembly properly. In order to determine, for example, whether a part can be moved and which other parts will move with it, the assembly model must include the spatial positions and hierarchical relationships among the parts, and the assembly or attachment relationships (also referred as mating conditions) between parts. The modeling representation of hierarchical relationships and mating conditions are what distinguishes between modeling individual parts and assemblies, and consequently between geometric modelers and assembly modelers.

Most of the existing modeling packages offered by today's CAD/CAM systems that are in use in practice can be classified as geometric modelers. Their data structures are designed to store and manipulate geometric data of individual parts only. Assembly modelers can be thought of as more advanced geometric modelers where the data structure is extended to allow representation and manipulation of hierarchical relationships and mating conditions. The geometric modeler acts as a preprocessor to the assembly modeler. Designers first create all the shape information (both geometry and topology) of the individual parts. They can also analyze the parts separately. Part analysis may include mass property calculations and finite element analysis. Once the parts design is complete, designers can utilize the assembly modeler to create the assembly and analyze it. Creating the assembly from its parts requires specifying the mating and spatial relationships between the parts. Assembly analysis may include interference checking, mass properties kinematical and dynamic analysis, and finite element analysis. The link between the geometric and assembly modelers is established such that designers need only to modify individual parts for design modification by using the geometric modeler and the assembly model is updated automatically.

In the context of the foregoing, some actions are necessary for assembly modeling:

- Modeling of individual parts (Creating subassemblies);
- Specifying the hierarchical relationships between parts in the assembly;
- Turning the part or subassembly into a component definition;
- Instancing component definitions into the assembly;

- Constraining assembly components - specifying the mating conditions between parts or specifying the locations and orientations of the parts in their assembled positions;
- Editing and modifying the assembly;
- Testing and analyzing the assembly;
- Documenting the assembly.

6.2 PARTS MODELING AND REPRESENTATION

This is the first step in creating an assembly model. Individual parts can be created using a geometric modeler with the proper representation scheme. Solid modeling, specifically boundary representation, is the appropriate scheme because the mating conditions are related to the faces, edges, and vertices of the assembled parts. In addition to the shape information, a part database can store assembly attributes such as the parts material type and properties, mass and inertial properties, frictional properties of the faces, and others. Some of these attributes such as the mass and inertial properties can be evaluated automatically by the geometric modeler after the part's database is created.

6.3 HIERARCHICAL RELATIONSHIPS

The hierarchy is of great importance to the assembly sequence. Before one can create an assembly model one must define parts and subassemblies as component definitions. There are some reasons for this:

- After a part model is completed, it might be needed by other users for different assemblies. For example, the same washer could be used by different users for wheel assembly or for a handle. Using a reference copy leaves the actual part model intact and available for number of users;
- If the user needs to use a part model multiple times in your assembly, for example the same bolt or nut, then he needs not to create number of models, which takes less space;
- When a component definition is changed, all references to that part update automatically. Changing the bolt definition changes the bolt wherever it has been used, eliminating the work needed to change every instance of the bolt manually.

By creating the component definition, you package all of the part's information - its parameters, geometric data, specifications, and name into a block that can be inserted into an assembly. The component definition thus creates a link between the individual part and the assembly drawing. With the modern assembly modelers however, instanting the part into the assembly creates component definition automatically.

When a part or subassembly is defined as a component definition, it can then be instanced into an assembly or subassembly. An instance is a single occurrence of a component definition into a physical assembly. It is a copy of the

actual part model. It is important to consider carefully the order in which the component instances are brought into the assembly. This is very important, and it is general rule that one instances his component definitions in the same order in which one wishes to assemble them.

The earlier the user instances the part into the drawing, the higher rank it has in the assembly. The user can usually attach parts (or mate parts) from the lower rank to the higher rank.

The first instance into the assembly is called the **grounded component**. Because the grounded component is the first component in the assembly, it has not degree of freedom - is immovable. In fact it could be viewed as a magnet, which attracts the other components towards him. For example, if you wish to constrain instances 1 (the grounded component) and component 3 in the assembly, instance 3 will move to instance 1 and take appropriate position according to the mating conditions specified.

When creating an assembly one may wish to constrain two components before constraining them to the grounded component. Constraining components without a link to the grounded component creates **floating sets**.

The most natural way to represent the hierarchical relationships between the various parts of an assembly is an assembly tree as shown bellow. An assembly is divided into several subassemblies at different levels (shown in the figure as the tree depths), and each subassembly at depth $(n - 1)$ is composed of various parts. The leaves of the tree represent individual parts (a subassembly leaf can be decomposed into its individual pasts), its nodes represent subassemblies, and its root represents the assembly itself. The assembly is located at the top of the tree at depth 0 or at the highest hierarchy n of the assembly sequence.

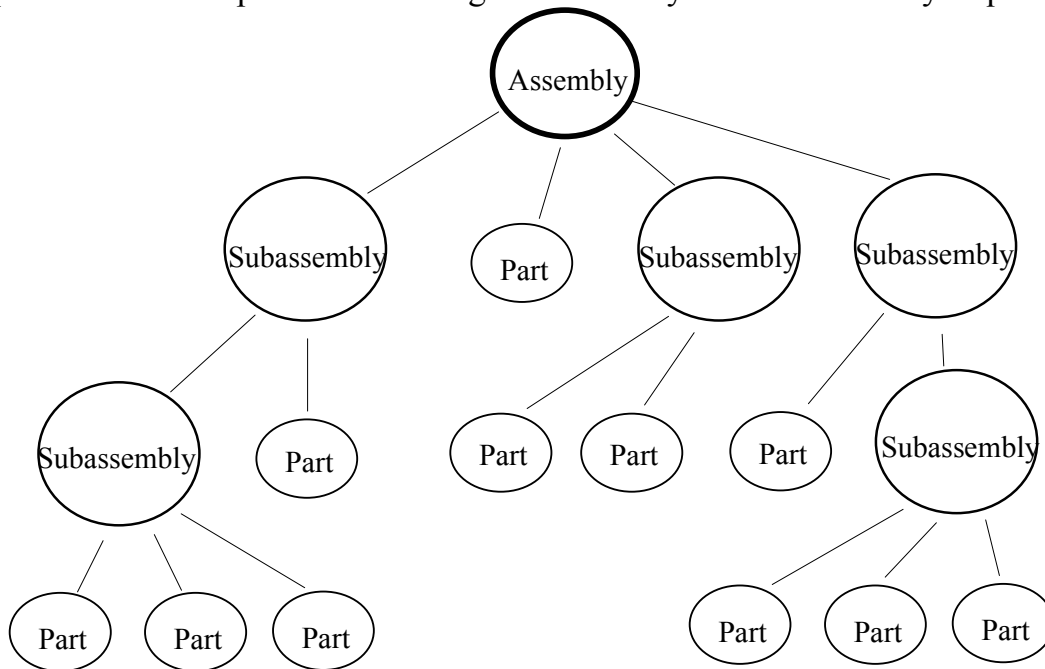


Figure 6.1

The assembly tree is not unique as it is possible to generate other valid assembly sequences.

6.4 MATING CONDITIONS. ASSEMBLY CONSTRAINTS

Individual parts of an assembly are usually created separately using a CAD/CAM system and then merged (assembled) together to form assembly. Parts may have to be scaled up or down before merging to fit properly into the assembly. Each part has its own database with its own PCS (part coordinate system). Typically, the user selects one of the parts as a base part (host by first instances into the assembly model) and merges the other parts into it. The PCS of the host becomes the global coordinate system, that is, the WCS (world coordinate system) of the assembly and the PCS of each other part becomes a local coordinate system for this part. The final correct position of each part in the assembly is obtained by locating and orienting its corresponding PCS property with respect to the global coordinate system of the assembly.

There are two alternatives of representing an assembly depending on how the locations and orientations of its various parts are provided by the user. The simplest alternative is to specify the location and orientation of each part in the assembly, together with the representation of the part itself, by providing a 4×4 homogeneous transformation matrix. This matrix transforms the coordinates of the geometric entities of the part from its local coordinate system (PCS) to the global coordinate system of the assembly. One convenient way for the user is to provide the transformation matrix interactively by specifying the location of the local coordinate system of a part relative to the assembly global coordinate system and by forcing the orientation of the local system to coincide with the orientation of the proper ICS (intermediate coordinate system). This ICS is defined relative to the assembly global coordinate system and sets the orientation of the part after mating into the assembly.

This alternative of merging parts into their assemblies is however tedious by still used in some existing CAD/CAM systems.

The other alternative to represent an assembly is based on specifying the spatial relationships between its individual parts as mating conditions. Mating feature information can be provided interactively with ease because mating features are simple graphics entities such as faces and centerlines or edges.

Mating relationships give possibility to position precisely the components with respect to each other in an assembly. They let the user define how the components move and rotate with respect to other components. By adding mating relationships successively, one can move the components into the desired positions. Mating creates geometric relationships, such as coincident, perpendicular, tangent, and so on. Each mating relationship is valid for specific combinations of geometry.

Table 6.1 gives an example of possible relationships between some geometry features.

Table 6.1

	Cylinder	Sphere	Plane	Line	Point
Cylinder	Angle Concentric Distance Parallel Perpendicular Tangent	Concentric Tangent	Distance Tangent	Angle Coincident Concentric Distance Parallel Perpendicular Tangent	Coincident Concentric Distance
Sphere	Concentric Tangent	Concentric Distance Tangent	Distance Tangent	Concentric Distance Tangent	Coincident Concentric Distance
Plane	Distance Tangent	Distance Tangent	Angle Coincident Distance Parallel Perpendicular	Coincident Distance Parallel Perpendicular	Coincident Distance
Line	Angle Coincident Concentric Distance Parallel Perpendicular Tangent	Concentric Distance Tangent	Coincident Distance Parallel Perpendicular	Angle Coincident Distance Parallel Perpendicular	Coincident Distance
Point	Coincident Concentric Distance	Coincident Concentric Distance	Coincident Distance	Coincident Distance	Coincident Distance

For example, mating condition can consist of planar faces butting up against one another ('against' condition) or requiring centerlines of individual parts to be collinear 'fits' condition). Therefore, by simply selecting graphics entities the assembly data can be provided interactively. Providing the assembly data as mating features seems more natural than defining and supplying transformations as required by the first alternative. By assigning the mating conditions, the transformation matrices that merge parts into their assembly can be automatically computed and stored for each part. In addition, using mating conditions instead of providing transformation matrices determines whether or not the parts in an assembly can be assembled. If some parts cannot physically be assembled due to specifying inconsistent mating conditions, the transformation matrices satisfying all mating conditions do not exist. The computation algorithm for transformation matrices will diverge.

Mating conditions or mating relationships are also called **spatial constraints** or **3D constraints**. Assembly constraints also determine how the component parts in the assembly fit together one at a time, in the order they have been instantiated into the assembly. The system usually reads the instances in numerical order in the graphics display or top-to-bottom in the assembly tree. The process works as follows:

— The first component instanced into the assembly or subassembly is the grounded component, which does not move. Consider this the first instance in the graphics display or the top-most instance in the assembly tree. The grounded component acts as a constraint ‘magnet’ and pulls all other instances to it.

— All constraints between the grounded component and the second component are solved and the second component is positioned or ‘pulled’ to the grounded component. The second component then becomes part of the grounded component.

— All constraints between the grounded component and the third component are solved and the third component is positioned to the grounded component.

— The third component becomes part of the grounded component and the process continues through all the components, in order, until everything is in position.

There are two important rules to remember when constraining components:

— An unconstrained component can be constrained to the grounded component provided the unconstrained component is located later in the assembly tree than the selected grounded component;

— A constraint can be added between a floating set and grounded component provided all the components in the set are located later in the assembly tree than the selected component of the set. The selected component of the set must also be later than the selected grounded component.

Here are some examples for typical mating conditions. These constraints are defined by the direction of the normal and the object used for the constraint.

Mate. Mate is the most commonly used in different CAD systems constraint because it positions all object types. A mate constraint causes a point, line (with direction), or plane (with direction) on one component to be coincident with a point, line (with opposite direction), or plane (with opposite direction) on a second component. Objects that can be selected include:

— two planes coplanar with their normals aligned in opposite directions (facing each other also called ‘against’ condition – see Fig. 6.2);

— an axis planar with a plane;

— two axes that share the same direction and slope (collinear) ;

— a point that lies on an axis;

— two coincident points ;

— a sphere, cylinder, or cone tangent to a plane or to other spheres, cylinders, and cones (see Fig. 6.3).

How the ‘against’ or ‘mate’ condition works is illustrated in Fig.6.3. the two parts have their own MCSs (Model Coordinate Systems) X_1, Y_1, Z_1 and X_2, Y_2, Z_2 respectively. The designated (shown dark in the figure) faces are the faces to be mated. Each face is specified by its unit normal vector and any one point on the face with respect to the part MCS. For example, the planar face of

the first part is specified by a unit with respect to the X_1, Y_1, Z_1 coordinate system. Similarly, the cylindrical face of the second part is specified by a unit vector with respect to the X_2, Y_2, Z_2 coordinate system. The 'against' condition is satisfied by forcing the two unit vectors to be opposite to each other, and the two faces touch each other. This is the same situation shown on Fig. 6.2 mating two planar faces.

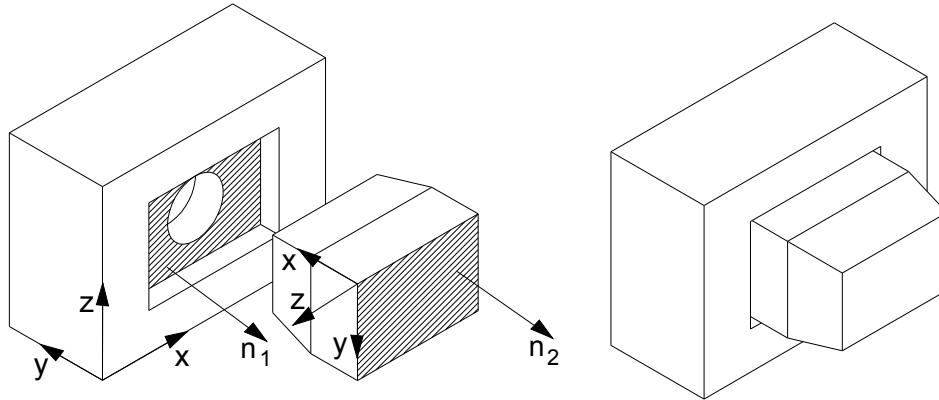


Figure 6.2

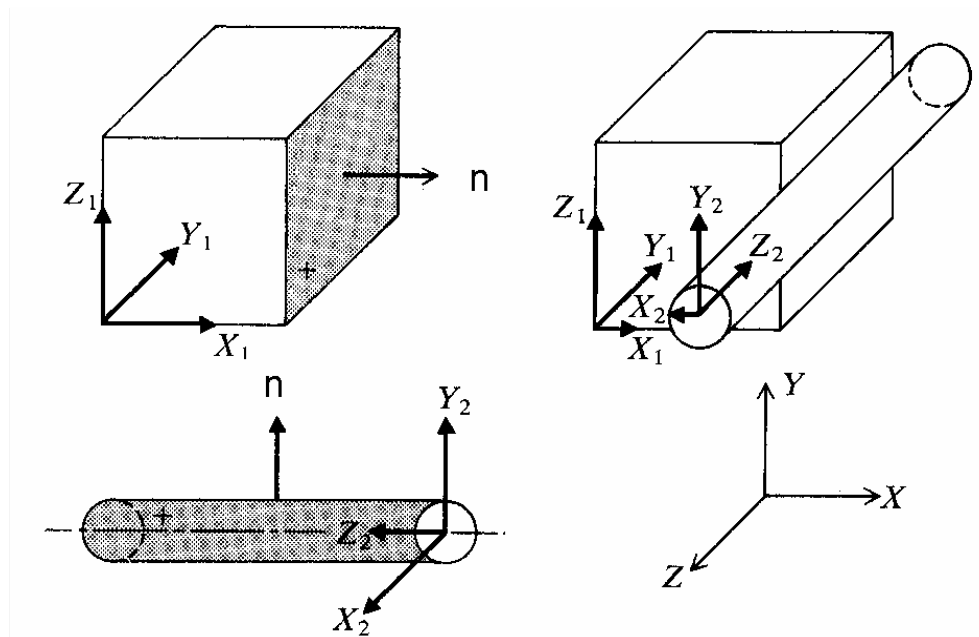


Figure 6.3

Insert. The insert constraint (or 'fits' condition) aligns two circles, including their centre axes and planes (for example, a bolt into a hole). The 'fits' condition is achieved by forcing the shaft and hole axes to be collinear (Fig. 6.4). Each axis is specified by two points.

The 'tight fits' condition is introduced to prevent the rotational movement that may accompany the 'fits' condition. The 'tight fits' condition describes fits between parts where the force to rotate one part relative to another is too great to

be called a rotational degree of freedom. One can think of ‘fits’ as a clearance fit and of ‘tight fits’ as an interference fit.

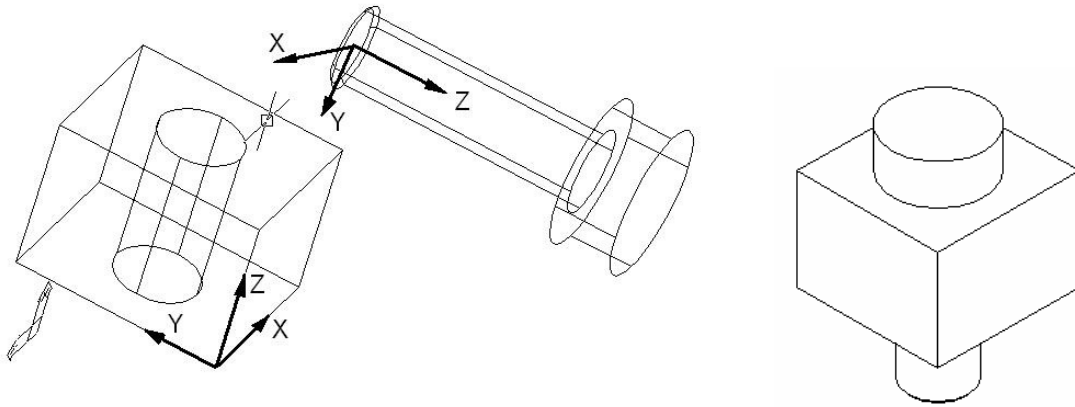


Figure 6.4

Flush. Flush (or ‘coplanar’ condition) creates coplanar positioning between two components. A flush constraint causes a plane (with direction) on one component to be coplanar with a plane (with the same direction) on a second component. The ‘coplanar’ condition is the complement (opposite) of the ‘against’ condition, and is satisfied by forcing the two normals to be in the same direction.

An **Angle** condition might be used to specify the exact angle between the normals of two planes, two vectors or a combination of a plane and a vector. The angle is measured according to the positive directions of normals (normally pointed outside the body), or positive directions of vectors.

6.5 DEGREES OF FREEDOM

It is known from the mechanics that the free body in space is capable of moving in six relatively independent directions – three translational movements along three mutually perpendicular axes and three rotational movements about the same axes. Thus the free body in space has 6 degree of freedom. Constraining the element with 3D constraints is also connected with eliminating rigid body degrees of freedom (DOF). Each time a constraint between two components is added, one (or more) DOF is eliminated. A fully constrained component cannot move in any direction.

Each condition eliminates different number DOF. For example the ‘against’ (or mate) conditions with two planes eliminates one translational and two rotational DOFs (see Fig. 6.2), which is the same with the ‘flush’ or ‘align’ conditions. The ‘fits’ or ‘insert’ condition eliminates two rotational and two

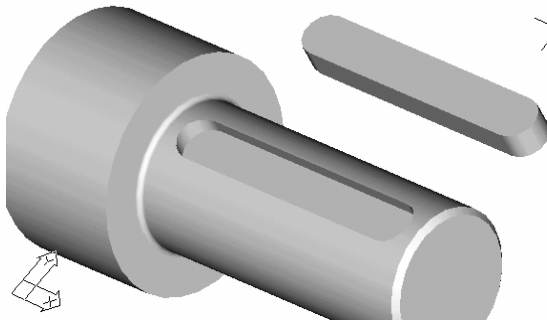


Figure 6.5

translational DOFs, while ‘tight fits’ removes all six DOF.

The following example constraints a parallel key into the shaft groove with elimination of all degrees of freedom. The initial is shown on Fig. 6.5. The process will be done in three stages:

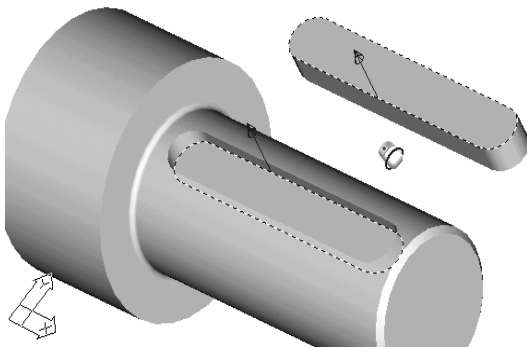


Figure 6.6

The first step is to use the ‘against’ or (‘mate’) condition with face – face mating in order to make the faces of the groove and the key co-planar (laying on the same plane). The corresponding faces and their normal vectors are shown on Fig 6.6. The ‘mate’ condition will cause the normal vectors to point in opposite directions after the mating, so the key will be rotated 180 degrees about the longitudinal axis. This operation takes 3 DOF – two rotational and one translational.

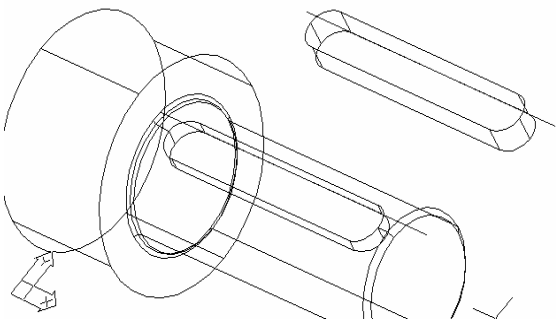


Figure 6.7

On **the second stage** we will use the same ‘mate’ constraint with line-to-line mating in order to align the edges of the key and the groove. After the mating, the corresponding edges (as shown on Fig. 6.7) will become collinear as shown on Fig. 6.9. This constraint takes two additional DOF – one rotational and one translational.

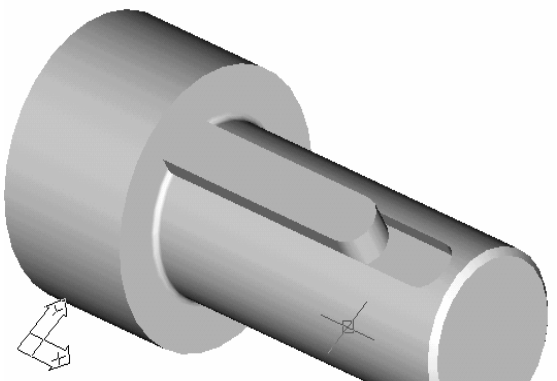


Figure 6.8

The **last step** will eliminate the last remaining degree of freedom by making the centerlines of the semi-cylindrical part of the key and the groove coincident

Questions:

1. List the actions necessary for assembly modeling:
2. What are the benefits of turning parts into component definition?
3. What is grounded component?
4. Give definition for the mate constraint. Give graphical example.
5. Give definition for the insert constraint. Give graphical example.
6. How many are the DOF taken with line to line mate condition?
7. How many are the DOF taken with line to plane mate condition?
- 8.