Computer Aided Engineering Design

Computer Aided Engineering Design

Anupam Saxena □ Birendra Sahay
Department of Mechanical Engineering
Indian Institute of Technology Kanpur, India

Springer

CONSISTING OF KNOWLEDGE
Anamaya

A C.I.P. catalogue record for the book is available from the Library of Congress

ISBN 1-4020-2555-6 (HB)

Copublished by Springer
233 Spring Street, New York 10013, USA
with Anamaya Publishers, New Delhi, India

Sold and distributed in North, Central and South America by Springer
233 Spring Street, New York, USA

In all the countries, except India, sold and distributed by Springer
P.O. Box 322, 3300 AH Dordrecht, The Netherlands

In India, sold and distributed by Anamaya Publishers
F-230, Lado Sarai, New Delhi-110 030, India

9 8 7 6 5 4 3 2 1

springeronline.com

Printed in India.

To my parents, all my teachers and for my son Suved when he grows up
Anupam Saxena

To my mother, Charushila Devi, an icon of patience
To my eldest brother, Dhanendra Sahay who never let me feel the absence of my father
To my brother, Dr. Barindra Sahay for initiating me into the realm of mathematics
To my teachers at McMaster University and University of Waterloo
To my wife, Kusum, children Urvashi, Menaka, Pawan and the little fairy
Radhika (granddaughter)
Birendra Sahay

Foreword

A new discipline is said to attain maturity when the subject matter takes the shape of a textbook. Several textbooks later, the discipline tends to acquire a firm place in the curriculum for teaching and learning. Computer Aided Engineering Design (CAED), barely three decades old, is interdisciplinary in nature whose boundaries are still expanding. However, it draws its core strength from several acknowledged and diverse areas such as computer graphics, differential geometry, Boolean algebra, computational geometry, topological spaces, numerical analysis, mechanics of solids, engineering design and a few others. CAED also needs to show its strong linkages with Computer Aided Manufacturing (CAM). As is true with any growing discipline, the literature is widespread in research journals, edited books, and conference proceedings. Various textbooks have appeared with different biases, like geometric modeling, computer graphics, and CAD/CAM over the last decade.

This book goes into mathematical foundations and the core subjects of CAED without allowing itself to be overshadowed by computer graphics. It is written in a logical and thorough manner for use mainly by senior and graduate level students as well as users and developers of CAD software. The book covers

(a) The fundamental concepts of geometric modeling so that a real understanding of designing synthetic surfaces and solid modeling can be achieved.
(b) A wide spectrum of CAED topics such as CAD of linkages and machine elements, finite element analysis, optimization.
(c) Application of these methods to real world problems.

In a new discipline, it is also a major contribution creating example problems and their solutions whereby these exercises can be worked out in a reasonable time by students and simultaneously encouraging them to tackle more challenging problems. Some well tried out projects are also listed which may enthuse both teachers and students to develop new projects. The writing style of the book is clear and thorough and as the student progresses through the text, a great satisfaction can be achieved by creating a software library of curve, surface, and solid modeling modules.

Dr. Anupam Saxena earned his MSME degree in 1997 at the University of Toledo, Ohio, USA. I am familiar with his work on a particularly challenging CAED problem for his thesis. He earned his Ph.D. degree from the University of Pennsylvania, USA and became a faculty member at IIT Kanpur in 2000. Dr. Sahay was Professor at IIT Kanpur where he performed research and teaching in design related fields for over the past 32 years after having earned his Ph.D. from the University of Waterloo, Canada. This textbook is a result of over ten years of teaching CAED by both authors.

The topics covered in detail in this book will, I am sure, be immensely helpful to teachers, students, practitioners and researchers.

Steven N. Kramer, PhD, PE
Professor of Mechanical and Industrial Engineering
The University of Toledo, Toledo, Ohio

Preface

The development of *computer aided engineering design* has gained momentum over the last three decades. Computer graphics, geometric modeling of curves, surfaces and solids, finite element method, optimization, computational fluid flow and heat transfer—all have now taken roots into the academic curricula as individual disciplines. Several professional softwares are now available for the design of surfaces and solids. These are very user-friendly and do not require a user to possess the intricate details of the mathematical basis that goes behind.

This book is an outcome of over a decade of teaching computer aided design to graduate and senior undergraduate students. It emphasizes the mathematical background behind geometric modeling, analysis and optimization tools incorporated within the existing software.

- Much of the material on CAD related topics is widely scattered in literature. This book is conceived with a view to arrange the source material in a logical and comprehensive sequence, to be used as a semester course text for CAD.
- The *focus* is on computer aided design. Treatment essential for geometric transformations, projective geometry, differential geometry of curves and surfaces have been dealt with in detail using examples. Only a background in elementary linear algebra, matrices and vector geometry is required to understand the material presented.
- The concepts of homogeneous transformations and affine spaces (barycentric coordinate system) have been explained with examples. This is essential to understand how a solid or surface model of an object can escape coordinate system dependence. This enables a distortion-free handling of a computer model under rigid-body transformations.
- A viewpoint that free-form solids may be regarded as composed of surface patches which instead are composed of curve segments is maintained in this book, like most other texts on CAD. Thus, geometric modeling of curve segments is discussed in detail. The basis of curve design is parametric, piecewise fitting of individual segments of low degree into a composite curve such that the desired continuity (position, slope and/or curvature) is maintained between adjacent segments. This reduces undue oscillations and provides freedom to a designer to alter the curve shape. A generic model of a curve segment is the weighted linear combination of user-specified data points where the weights are functions of a *normalized*, *non-negative* parameter. Further, barycentricity of weights* makes a curve segment independent of the coordinate system and provides an insight into the curve's shape. That is, the curve lies within

* Weights are all non-negative and for any value of the parameter, they sum to unity.

the convex hull of the data points specified. The associated variation diminishing property suggests that the curve's shape is no more complex than the polyline of the control points itself. In other words, a control polyline primitively approximates the shape of the curve. For Bézier segments, barycentricity is global in that altering any data point results in overall shape change of the segment. For B-spline curves, however, weights are locally barycentric allowing shape change only within some local region. Expressions for weights, that is, Bernstein polynomials for Bézier segments and B-spline basis functions for B-spline curves are derived and discussed in detail in this book and many examples are presented to illustrate curve design.

- With the design of free-form curve segments accomplished, surface patches can be obtained in numerous ways. With two curves, one can sweep one over the other to get a *sweep surface patch*. One of the curves can be rectilinear in shape and represent an axis about which the second curve can be revolved to get a *patch of revolution*. One can join corresponding points on the two curves using straight lines to generate a *ruled surface*. Or, if cross boundary slope information is available, one can join the corresponding points using a cubic segment to get a *lofted patch*. More involved models of surface patches are the bilinear and bicubic Coon's patches wherein four boundary curves are involved. Eventually, a direct extension of Bézier and B-spline curves is their tensor product into respective free-form Bézier and B-spline surface patches. These surface patches inherit the properties from the respective curves. That is, the surface patch lies within the control polyhedron defined by the data points, and that the polyhedron loosely represents the patch shape. The aforementioned patches are derived and discussed in detail with examples in this book. Later, methods to model composite surfaces are discussed.
- The basis for solid modeling is the extension of Jordon's curve theorem which states that a closed, simply connected** (planar) curve divides a plane into two regions; its interior and its exterior. Likewise, a closed, simply connected and orientable surface divides a three-dimensional space into regions interior and exterior to the surface. With this established, a simple, closed and connected surface constituted of various surface patches *knit* or *glued* together at their respective common boundaries encloses a finite volume within itself. The union of this interior region with the surface boundary represents a free form solid. Any solid modeler should be generic and capable of modeling unambiguous solids such that any set operation (union, intersection or difference) performed on two valid solids should yield another valid solid. With this viewpoint, the concept of geometry is relaxed to study the topological attributes of valid solids. Such properties disregard *size* (lengths and angles) and study only the *connectivity* in a solid. With these properties as basis, the three solid modeling techniques, i.e., wireframe modeling, boundary representation method and constructive solid geometry are discussed in detail with examples. Advantages and drawbacks of each method are discussed and it is emphasized that professional solid modelers utilize all three representations depending on the application. For instance, wireframe modeling is usually employed for animation as quick rendering is not possible with the boundary representation scheme.
- Determination of intersection between various curves, surfaces and solids is routinely performed by the solid modelers for curve and surface trimming and blending. Intersection determination is primarily used in computing Boolean relations between two solids in constructive solid

** A closed curve with no self intersection.

geometry. Computational geometry that encompasses a set of algorithms to compute various relations like proximity, intersection, decomposition and relational search (e.g., point membership classification) between geometric entities is discussed in brief in this book. The working of these algorithms is described for polygonal entities with examples for easy understanding of the subject matter.

- Reverse engineering alludes to the process of creating CAD models from existing real life components or their prototypes. Applications are prolific; some being the generation of customized fit to human surfaces, designing prostheses, and reconstruction of archaeological collections and artifacts. For an engineering component whose original data is not available, a conceptual clay or wood model is employed. A point cloud data is acquired from an existing component or its prototype using available non-contact or tactile scanning methods. Surface patches are then locally modeled over a subset of the point cloud to interpolate or best approximate the data. Reverse engineering is an important emerging application in Computer Aided Design, and various methods for surface patch fitting, depending on the scanning procedure used, are briefed in this book.
- Having discussed in detail the geometric modeling aspects in free-form design, this book provides an introductory treatment to the finite element analysis (FEM) and optimization, the other two widely employed tools in computer aided design. Using these, one can analyze and alter a design form such that the latter becomes optimal in some sense of the user specified objective. The book discusses linear elastic finite element method using some basic elements like trusses, frames, triangular and four-node elements. Discussion on optimization is restricted to some numerical methods in determining single variable extrema and classical Karush-Kuhn-Tucker necessary conditions for multi-variable unconstrained and constrained problems. Sequential Linear and Quadratic Programming, and stochastic methods like genetic algorithms and simulated annealing are given a brief mention. The intent is to introduce a student to follow-up formal courses on finite element analysis and optimization in the curricula.

This book should be used by the educators as follows:

Students from a variety of majors, e.g., mechanical engineering, computer science and engineering, aeronautical and civil engineering and mathematics are likely to credit this course. Also, students may study CAD at primarily graduate and senior undergraduate levels. Geometric modeling of curves, surfaces and solids may be relevant to all while finite element analysis and optimization may be of interest of mechanical, aeronautical and civil engineering. Discretion of the instructor may be required to cover the combination of topics for a group of students. Considering a semester course of 40 contact hours, a broad breakup of topics is suggested as follows:

- 1^{st} hour: Introduction to computer aided design
- 3 hours: Transformations and projections
- 15 hours: Free-form curve design
- 9 hours: Surface patch modeling
- 6 hours: Solid modeling

The remaining 6 hours may be assigned as follows: for students belonging to mechanical, aeronautical and civil engineering, reverse engineering, finite element method and optimization may be introduced and for those in computer science and engineering and mathematics, computational geometry and optimization may be emphasized.

For a group of graduate students taking this course, differential geometry of curves and surfaces

(Chapters 3 and 6) may be dealt with in detail. Also, topological attributes of solids may be discussed. For only senior undergraduate students, differential geometry may be covered in brief emphasizing mainly Frenet-Serret relations, Gaussian and Mean curvatures and their importance in determining the nature of a surface. Chapters on computational geometry, reverse engineering, FEM and optimization may be omitted.

Assignments and projects form an important part of this course. Assignments may be tailored in a manner that students get a handle on manual calculations as well as code development for curve and surface design. A course project may run over a semester or can be in two parts each covering half the semester. Some example projects are mentioned in Appendix III.

Some examples presented in Chapter 1 on kinematic analysis and spring design pertain to students in mechanical engineering. For a generic class, an instructor may prefer to cover curve interpolation and fitting discussed in sections 3.1 and 3.2.

The practitioners, i.e., those developing professional software would require much deeper understanding of the design principles, mathematical foundations and computer graphics to render a robust Graphical User Interface to the software. This book would help them acquire adequate background knowledge in design principles and mathematical foundations. Those using the software may not require a deeper understanding of the mathematical principles. However, design aspects and essential properties of curve, surface and solid modeling would be needed to create the design and interpret the results.

Chapters 9 and 10 of this book on computations with geometry and modeling using point clouds has been contributed by Dr. G. Saravana Kumar, a former Ph.D. Student, Mechanical Engineering Department, IIT Kanpur. His enthusiasm as T.A. in the CAD course has also resulted in several good projects.

ANUPAM SAXENA
BIRENDRA SAHAY

Acknowledgements

The authors acknowledge the support of the Quality Improvement Program (QIP) at the Indian Institute of Technology, Kanpur for the preparation of this manuscript.

The design in Chapter 8 is attributed to three undergraduate students, Sandeep Urankar, Anurag Singh and Pranjal Jain who have accomplished their B. Tech project on *Robosloth* in 2002. G.S. Sharavanan Kumar, a Ph.D. student at IIT Kanpur has contributed Chapters 9 and 10. Other graduate students, particularly, Anupam Agarwal, Rajat Saxena and Manak Lal Jain have contributed in creating the figures and examples for some chapters. The authors also acknowledge the assignment contributions and assistance of their students in the CAD-2002(II) and 2003(I) classes, particularly, Abhishek Gupta, Anurag Tripathy, Rajkumar Singh, Gaurav Dubey, Shubham Saxena, Abhishek Luthra, Pritam Chakraborty, T.S. Sudhish Kumar and Prince Malik.

Contents

Computer Aided Engineering Design

Chapter 1

Introduction

The development of mankind has depended on the ability to modify and shape the material that nature has made available, in ways to provide them their basic needs, and security and comfort required for their survival and advancement. They have devised tools for hunting, implements for agriculture, shelter for safeguard against the vagaries of nature, and wheels for transportation, an invention mankind has always been proud of. Much of the aforementioned *design* accomplishments have resulted even before mankind may have learnt to count. The then trial-and-error and/or empirical design procedures have been systematized to a great extent using the human understanding of the laws of physics (on force, motion and/or energy transfer) with concepts from mathematics. An idea to fulfill a need and then translating the idea into an implement forms the core of activities in design. *Design and manufacture is innate to the growth of human civilization.*

1.1 Engineering Design

Design is an activity that facilitates the realization of new products and processes through which technology satisfies the needs and aspirations of the society. Engineering design of a product may be conceived and evolved in four steps:

1. *Problem definition:* Extracting a coherent appreciation of *need* or *function* of an engineering part from a fuzzy mix of facts and myths that result from an initial ill-posed problem. The data collection can be done via *observation* and/or a *detailed survey*.
2. *Creative process:* Synthesizing *form*, a design solution to satisfy the need. Multiple solutions may result (and are sought) as the creative thought process is aided by the designers' vast experience and knowledge base. *Brainstorming* is usually done in groups to arrive at various forms which are then evaluated and selected into a set of a few workable solutions.
3. *Analytical process: Sizing* the components of the designed *forms*. Requisite functionality, strength and reliability analysis, feasible manufacturing, cost determination and environmental impact may be some design goals that could be improved optimally by altering the components' dimensions and/or material. This is an iterative process requiring design changes if the analysis shows inadequacy, or scope for further improvement of a particular design. Multiple solutions may be evaluated simultaneously or separately and the *best* design satisfying most or all functional needs may be chosen.
4. *Prototype development and testing:* Providing the ultimate check through physical evaluation under, say, an actual loading condition before the design goes for production. Design changes are

needed in the step above in case the prototype fails to satisfy a set of needs in step 1. This stage forms an interface between design and manufacture. Many groups encourage prototype failure as many times as possible to quickly arrive at a successful design.

1.2 Computer as an Aid to the Design Engineer

Machines have been designed and built even before the advent of computers. During World War-II, ships, submarines, aircrafts and missiles were manufactured on a vast scale. In the significant era (19th and 20th century) of industrial revolution, steam engines, water turbines, railways, cars and power-driven textile mills were developed. The method of representing three-dimensional solid objects was soon needed and was formalized through orthographic projections by a French mathematician Gaspard Monge (1746-1818). After the military kept it a secret for nearly half a century, the approach was made available to engineers, in general, towards the end of nineteenth century.

The inception of modern computers lies in the early work by Charles Babbage (1822), punched card system developed for the US census by Herman Hollerith (1890), differential analyzer at MIT (1930), work on programmable computers by Allan Turing (1936), program storage concept and re-programmable computers by John von Neumann (1946) and micro-programmed architecture by Maurice Wilkes (1951).

The hardware went through a revolution from electronic tubes, transistors (1953), semi-conductors (1953), integrated circuits (1958) to microprocessors (1971). The first 8-bit microcomputer was introduced in 1976 with the Intel 8048 chip and subsequently 16 and 32-bit ones were introduced in 1978 and 1984. Currently, 32 bit and 64 bit PCs are used. Tremendous developments have taken place in hardware, especially in the microprocessor technology, storage devices (20 to 80 GB range), memory input/output devices, compute speed (in GHz range) and enhanced power of PCs and workstations, enabling compactness and miniaturization. The display technology has also made significant advances from its bulky Cathode Ray Tube (CRT) to Plasma Panel and LCD flat screen forms.

Interactive Computer Graphics (ICG) was developed during the 1960s. Sutherland (1962) devised the Sketchpad system with which it was possible to create simple drawings on a CRT screen and make changes interactively. By mid 1960s, General Motors (GM), Lockheed Aircraft and Bell Laboratories had developed DAC-1, CADAM and GRAPHIC-1 display systems. By late 1960s, the term Computer Aided Design (CAD) was coined in literature. During 1970s, graphics standards were introduced with the development of GKS (Graphics Kernel System), PHIGS (Programmer's Hierarchical Interface for Graphics) and IGES (Initial Graphics Exchange Specification). This facilitated the graphics file and data exchange between various computers. CAD/CAM software development occurred at a fast rate during late 1970s (GMSolid, ROMULUS, PADL-2). By 1980s and 1990s, CAD/CAM had penetrated virtually every industry including Aerospace, Automotive, Construction, Consumer products, Textiles and others. Software has been developed over the past two decades for interactive drawing and drafting, analysis, visualization and animation. A few widely used products in Computer Aided Design and drafting are Pro-Engineer™, AutoCAD™, CATIA™, IDEAS™, and in analysis are NASTRAN™, ABAQUS™, ANSYS™ and ADAMS™. Many of these softwares have/are being planned to be upgraded for potential integration of design, analysis, optimization and manufacture.

1.2.1 Computer as a Participant in a Design Team

As it stands, a computer has been rendered a major share of the design process in a man-machine team. It behooves to understand the role of a human vis-à-vis a computer in this setting:

(a) *Conceptualization*, to date, is considered still within the domain of a human designer. Product design commences with the identification of its 'need' that may be based on consumer's/market's demand. An old product may also need design revision in view of new scientific and technological developments. An expert designer or a team goes through a creative and ingenious thought process (brainstorming), mostly qualitative, to synthesize the form of a product. A computer has not been rendered the capability, as yet, to capture non-numeric, qualitative 'thought' design, though it can help a human designer by making available relevant information from its stored database.
(b) *Search*, *learning and intelligence* is inherent more in a human designer who can be made aware of the new technological developments useful to synthesize new products. A computer, at this time, has little learning and 'qualitative thinking' capability and is not intelligent enough to synthesize a new form on its own. However, it can passively assist a designer by making available a large set of possibilities (stored previously) from a variety of disciplines, and narrow down the search domain for the designer.
(c) *Information storage and retrieval* can be performed very efficiently by a computer that has an excellent capability to store and handle data. Human memory can fade or fail to avail appropriate information fast enough, and at the right time from diverse sources. Further, a computer can automatically create a product database in final stages of the design.
(d) *Analytical power* in a computer is remarkable in that it can perform, say, the finite element analysis of a complex mechanical part or retrieve the input/output characteristics of a designed system very efficiently, provided mathematical models are embedded. Humans usually instruct the computers, via codes or software, the requisite mathematical models employed in *geometric modeling* (modeling of curves, surfaces and solids) and *analysis* (finite element method and optimization). Geometric modeling manifests the *form* of a product that a designer has in mind (qualitatively) while analysis works towards the systematic improvement of that *form*.
(e) *Design iteration* and improvement can be performed by a computer very efficiently once the designer has offloaded his/her conception of a product via geometric modeling. Finite element analysis (or other performance evaluation routine) and optimization can be performed simultaneously with the aim to modify the dimensions/shape of a product to meet the pre-specified design goals.
(f) *Prototyping* of the optimized design can be accomplished using the tools now available for Rapid Manufacturing. The geometric information of the final product can be passed on to a manufacturing set up that would analogically *print* a three dimensional product.

Computers help in manifesting the qualitative conception of a design form a human has of a product. Further, they prove useful in iterative improvement of the design, and its eventual realization. Computers are integrated with humans in design and manufacture, and provide the scope for automation (or least human interaction) wherever needed (mainly in analysis and optimization). Computer Aided Process Planning (CAPP), scheduling (CAS), tool design (CATD), material requirement planning (MRP), tool path generation for CNC machining, flexible manufacturing system (FMS), robotic systems for assembly and manufacture, quality inspection, and many other manufacturing activities also require computers.

1.3 Computer Graphics

Computer Graphics, which is a discipline within Computer Science and Engineering, provides an important mode of interaction between a designer and computer. Sutherland developed an early form of a computer graphic system in 1963. Rogers and Adams explain computer graphics as the *use* *of computers to define*, *store*, *manipulate*, *interrogate and present pictorial output*. Computer graphics involves the creation of two and three dimensional models, shading and rendering to bring in realism to the objects, natural scene generation (sea-shores, sand dunes or hills and mountains), animation, flight simulation for training pilots, navigation using graphic images, walk through buildings, cities and highways, and creating virtual reality. War gaming, computer games, entertainment industry and advertising has immensely benefited from the developments in computer graphics. It also forms an important ingredient in Computer-Aided Manufacturing (CAM) wherein graphical data of the object is converted into machining data to operate a CNC machine for production of a component. The algorithms of computer graphics lay behind the backdrop all through the process of virtual design, analysis and manufacture of a product. Two primary constituents of computer graphics are the *hardware* and the *software*.

1.3.1 Graphics Systems and Hardware

Hardware comprises the *input*, and *display* or *output devices.* Numerous types of graphics systems are in use; those that model one-to-many interaction and others that allow one-to-one interface at a given time. *Mainframe-based systems* use a large mainframe computer on which the software, which is usually a huge code requiring large space for storage, is installed. The system is networked to many designer stations on time-sharing basis with display unit and input devices for each designer. With this setting, intricate assemblies of engineering components, say an aircraft, requiring many human designers can be handled. *Minicomputer* or *Workstation* based systems are smaller in scale than the Mainframe systems with a limited number (one or more) of display and input devices. Both systems employ one-to-many interface wherein more than one designer can interact with a computer. On the contrary, *Microcomputer* (PC) based systems allow only one-to-one interaction at a time. Between the Mainframe, Workstation and PC based systems, the Workstation based system offers advantages of distributed computing and networking potential with lower cost compared with the mainframes.

1.3.2 Input Devices

Keyboard and *mouse* are the primary input devices. In a more involved environment, digitizers, joysticks and tablets are also used. Trackballs and input dials are used to produce complex models. Data gloves, image scanners, touch screens and light pens are some other input devices. A keyboard is used for submitting alphanumeric input, three-dimensional coordinates, and other non-graphic data in ‘text’ form. A mouse is a small hand held pointing device used to control the position of the cursor on the screen. Below the mouse is a ball. When the mouse is moved on a surface, the amount and direction of movement of the cursor is proportional to that of the mouse. In optical mouse, an optical sensor moving on a special mouse pad having orthogonal grids detects the movements. There are push buttons on top of the mouse beneath the fingers for signaling the execution of an operation, for selecting an object created on the screen within a rectangular area, for making a selection from the pulled down menu, for dragging an object from one part of the screen to other, or for creating drawings and dimensioning. It is an important device used to expedite the drawing operations. A special z-mouse for CAD, animation and virtual reality includes three buttons, a thumb-wheel and a track-ball on top. It gives six degrees of freedom for spatial positioning in x-y-z directions. The z-mouse is used for rotating the object around a desired axis, moving and navigating the viewing position (observer’s eye) and the object through a three-dimensional scene.

Trackballs, *space-balls* and *joysticks* are other devices used to create two and three-dimensional drawings with ease. Trackball is a 2-D positioning device whereas space-ball is used for the same in 3-D. A joystick has a vertical lever sticking out of a base box and is used to navigate the screen cursor.

Digitizers are used to create drawings by clicking input coordinates while holding the device over a given 2-D paper drawing. Maps and boundaries in a survey map, for example, can be digitized to create a computer map. *Touch panels* and *light pens* are input devices interacting directly with the computer screen. With touch panels, one can select an area on the screen and observe the details pertaining to that area. They use infrared light emitting diodes (LEDs) along vertical and horizontal edges of the screen, and go into action due to an interruption of the beam when a finger is held closer to the screen. Pencil shaped *light pens* are used to select screen position by detecting the light from the screen. They are sensitive to the short burst of light emitted from the phosphor coating as the electron beam hits the screen. *Scanners* are used to digitize and input a two-dimensional photographic data or text for computer storage or processing. The gradations of the boundaries, gray scale or color of the picture is stored as data arrays which can be used to edit, modify, crop, rotate or scale to enhance and make suitable changes in the image by software designed using geometric transformations and image processing techniques.

FaroArm®, a 3-D coordinate measuring device, is a multi-degree of freedom precision robotic arm attached to a computer. At the tip of the end-effector is attached a fine roller-tipped sensor. The tip can be contacted at several points on a curved surface to generate a point data cloud. A 3-D surface can then be fitted through the data cloud to generate the desired surface. A non-contact 3-D digitizer, Advanced Topometric Sensor (ATOS) uses optical measuring techniques. It is material independent and can scan in three-dimensions any arbitrary object such as moulds, dies, and sculptures. It is a high detailed resolution and precision machine. It uses adhesive retro targets stuck on the desired surface. Digital reflex cameras then record the positions of these retro targets from different views. The images consisting of the coordinates of targets are transferred from the digital camera to the computer. The image coordinates are then converted to the object coordinates by calculating the intersection of the rays from different camera positions. Finally, the required object surface is generated. Techniques for scanning objects in three-dimensions are very useful in reverse engineering, rapid prototyping of existing objects with complex surfaces such as sculptures and other such applications.

1.3.3 Display and Output Devices

Three types of display devices are in use: Cathode ray tube (*CRT*), Plasma Panel Display (PPD) and Liquid Crystal Display (LCD). CRT is a popular display device in use for its low cost and high-resolution color display capabilities. It is a glass tube with a front rectangular panel (screen) and a cylindrical rear tube. A cathode ray gun, when electrically heated, gives out a stream of electrons, which are then focused on the screen by means of positively charged electron-focusing lenses. The position of the focused point is controlled by orthogonal (horizontally and vertically deflecting) set of amplifiers arranged in parallel to the path of the electron beam. A popular method of CRT display is the Raster Scan. In raster scan, the entire screen is divided into a matrix of picture cells called *pixels*. The distance between pixel centers is about 0.25 mm. The total number of pixel sets is usually referred to as *resolution*. Commonly used CRTs are those with resolution of 640×480 (VGA), 1024×768 (XGA) and 1280×1024 (SXGA). With higher resolution, the picture quality is much sharper. As the focused electron beam strikes a pixel, the latter emits light, i.e. the pixel is 'on' and it becomes bright for a small duration of time. The electron beam is made to scan the entire screen line-by-line from top to bottom (525 horizontal lines in American system and 625 lines in European system) at 63.5 microseconds per scan line. The beam keeps on retracing the path. The *refresh rate* is 60Hz, implying that the screen is completely scanned in $1/60^{th}$ of a second (for European system, it is $1/50^{th}$ of a second). In a black and white display, if the pixel intensity is '0', the pixel appears black, and when '1', the pixel is bright. As the electron beam scans through the entire screen, it switches off those pixels which are supposed to be black thus creating a pattern on the screen. For the electron beam to know precisely which pixels are to be kept 'off' during scans, a *frame buffer* is used that is a hardware programmable memory. At least one memory bit ('0' or '1') is needed for each pixel, and there are as many bits allocated in the memory as the number of pixels on display. The entire memory required for displaying all the pixels is called a *bit plane* of the frame buffer.

One bit plane would create only a 'black' and 'white' image, but for a realistic picture, one would need *gray levels* or shades between black and white as well. To control the intensity (or shade) of a pixel one has to use a number of bit planes in a frame buffer. For example, if one uses 3 bit planes in single frame buffer, one can create 8 (or 2^3) combinations of intensity levels (or shades) for the same pixel- 000 (black)-001-010- 011-100-101-110-111(white). The intermediate values will control the intensity of the electron beam falling on the pixel. To have an idea about the amount of memory required for a black and white display with 256 × 256 (or 2^{16}) pixels, every bit plane will require a memory of 2^{16} = 65,536 bits. If there are 3 bit planes to control the gray levels, the memory required will be 1,96,608 bits! Since memory is a digital device and the raster action is analog, one needs digital-to-analog converters (DAC). A DAC takes the signal from the frame buffer and produces an equivalent analog signal to operate the electron gun in the CRT.

For *color display*, all colors are generated by a proper combination of 3 basic colors, viz. red, green, and blue. If we assign '0' and '1' to each color in the order given, we can generate 8 colors: black (000), red (100), green (010), blue (001), yellow (110), cyan (011), magenta (101) and white (111). The frame buffer requires a minimum of 3 bit planes—one for each RGB color; this can generate 8 different colors. If more colors are desired, one needs to increase the number of bit planes for each color. For example, if each of the RGB colors has 8 bit planes (a total of 24 bit planes in the frame buffer with three 8-bit DAC), the total number of colors available for picture display would be 2^{24} = 1,67,77,216! To further enhance the color capabilities, each 8-bit DAC is connected to a color look up memory table. Various methods are employed to decrease the access and display time and enhance the picture sharpness.

CRT displays are popular and less costly, but very bulky and suitable only for desktop PCs. Flat Panel Displays (FPD) are gaining popularity with laptop computers and other portable computers and devices. FPD belongs to one of the following two classes: (a) active FPD devices, which are primarily light emitting devices. Examples of active FPD are flat CRT, plasma gas discharge, electroluminescent and vacuum fluorescent displays. (b) Passive FPD devices are based on light modulating technologies. Liquid Crystal (LC) and Light Emitting Diodes (LED) are some examples.

Plotters and *printers* constitute the output devices. Line printers are the oldest succeeded by 9-pin and 24-pin *dot matrix plotters* and printers. *Ink jet plotters*, *laser plotters* and *thermal plotters* are used for small and medium sized plots. For large plots, *pen and ink plotters* of the flat bed, drum and pinch roller types are used.

1.4 Graphics Standards and Software

Till around 1973, software for producing graphics was mostly device dependent. Graphics software written for one type of hardware system was not portable to another type, or it became useless if the hardware was obsolete. Graphics standards were set to solve portability issues to render the application software device independent. Several standards have been developed; most popular among them are GKS (Graphics Kernel System), PHIGS (Programmer's Hierarchical Interactive Graphics System), DXF (Drawing Exchange Format), and IGES (Initial Graphics Exchange Specification).

For designing mechanical components and systems, one requires 3-D graphics capabilities for which GKS 3-D, PHIGS and DXF are suitable. For 3-D graphics and animation, PHIGS is used.

It provides high interactivity, hierarchical data structuring, real time graphic data modification, and support for geometric transformations. These standards provide the core of graphics including basic graphic primitives such as line, circle, arc, poly-lines, poly-markers, line-type and line-width, text, fill area for hatching and shading, locators for locating coordinates, valuators for real values for dimensioning, choice options and strings. Around such standard primitives, almost all standard software for CAD is written. They also include the device drivers for standard plotters and display devices.

Another comprehensive standard is IGES to enable the exchange of model databases among CAD/CAM systems. IGES contains more geometric entities such as, curves, surfaces, solid primitives, and Boolean (for Constructive Solid Geometry) operations. Wire-frame, surface modeling and solid modeling software can all be developed around IGES. It can transmit the property data associated with the drawings which helps in preparing, say, the bill of materials. Though these standards appear veiled or at the *back end*, they play a crucial role in creation of the application software.

1.5 Designer-Computer Interaction

A CAD/CAM software is designed to be primarily interactive, instructive and user-friendly wherein a designer can instruct a computer to perform a sequence of tasks ranging from designing to manufacture of an engineering component. The front end of a software is a graphical user interface or GUI while the back end comprises computation and database management routines. The front end is termed so as a user can visually observe the design operations being performed. However, computation and data storage routines are not very apparent to a designer, which is why they may be termed collectively as the back end of the software. In most CAD software, the GUI is divided into two parts (or windows) that appear on the display device or screen (Figure 1.1): (i) the visual manifestation or the

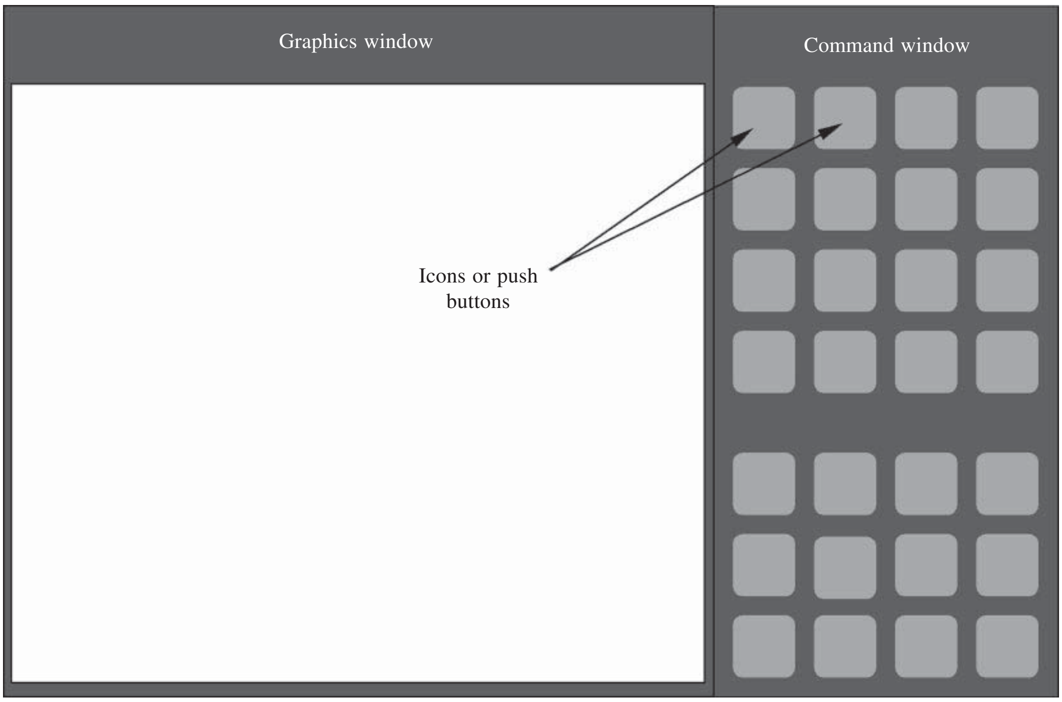

Figure 1.1 Generic appearance of the Front end of a CAD software

Graphics Window and (ii) the Command window. The Graphics window provides the visual feedback to the user detailing desired information about an object being designed. One can manipulate the position (through translation/rotation) of an object relative to another or a fixed coordinate system and visualize the changes in the Graphics window. In essence, all design operations involving transformations, curve design, design of surfaces and solids, assembly operations pertaining to relative positioning of two or more components, drafting operations that provide the engineering drawings, analysis operations that yield results pertaining to displacements and stresses, optimization operations that involve sequential alterations in design, and many others can be visualized through the Graphics window.

The design instructions are given through a user-friendly Command Window that is subdivided into several *push buttons* or *icons*. To accommodate numerous applications in CAD and to allow a guided user interface, the icons appear in groups. For instance, icons pertaining to the design of curves would be grouped in the Command window. Push buttons pertaining to curve trimming, extension, intersection and other such actions would be combined. Icons used in surface and solid design would appear in two different groups. Options under transformations, analysis, optimization and manufacture would also be clustered respectively. A user may make a design choice by clicking on an icon using the mouse. There may be many ways to design a curve, for instance. To accommodate many such possibilities, a CAD GUI employs the *pull down menus* (Figure 1.2). That is, when an icon on curve segment design is clicked on, a menu would drop down prompting the user to choose between, say, the Ferguson, Bézier or B-spline options. Similarly, for a surface patch design, a pull down menu may have choices ranging between the analytical patches, tensor product surfaces, Coon's patches, rectangular or triangular patches, ruled or lofted patches and many others. For solid modeling, a user may have to choose between Euler operations or Boolean sequences. After a design operation is chosen using a push button and from a respective pull down menu, the user would be prompted to enter further choices through *pop up* menus. For instance, if a user chooses to sketch a line, a pop up window may appear expecting the user to feed in the start point, length and orientation of the line. Note that for a two dimensional case, a much easier option to draw a curve segment may be to select a number of points on the screen through a sequence of mouse clicks.

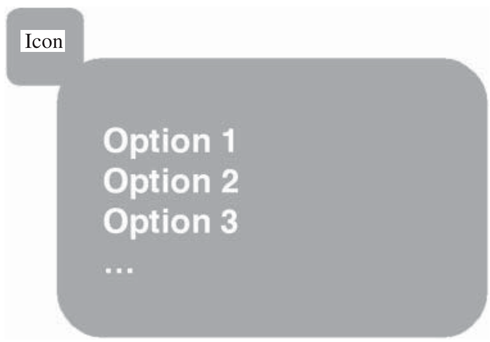

Figure 1.2 A pull down menu that appears when clicking on an icon in the command window

1.6 Motivation and Scope

Developing the front end GUI of a CAD software is an arduous and challenging task. However, it is the back end wherein the core of Computer Aided Design rests. This book discusses the design concepts based on which various modules or objects of the back end in a CAD software are written. The concepts emerge as an amalgamation of *geometry*, *mathematics* and *engineering* that renders the software the capability of *free-form* or generic design of a product, its analysis, obtaining its optimized form, if desired, and eventually its manufacture. Engineering components can be of various forms (sizes and shapes) in three-dimensions. A Solid can be thought of as composed of a simple *closed connected surface* that encloses a finite volume. The closed surface may be conceived as an interweaved arrangement of constituent surface patches, which in turn, can be individually considered as composed of a group of curves. It then behooves to discuss the generic design of curves, surfaces and solids in that order. Even before, it may be essential to understand how three-dimensional objects or geometrical entities are represented on a two-dimensional display screen, and how such entities can be positioned with respect to each other for assembly purposes or construction operations.

Engineers have converged to numerous standard ways of perceiving a three-dimensional component by way of *engineering drawings* depicted on a two-dimensional plane (conventionally blue prints, but for CAD's purpose, a display screen). The following chapter comprises a broad discussion on transformations and projections. Rotation and translation of a point (or a rigid body) with respect to the origin are discussed in two-dimensions. Both transformations are expressed in matrix notation using the homogenous coordinates. The advantage is that like rotation, translation can also be executed as a matrix multiplication operation without requiring any addition or subtraction of matrices or vectors. Performing a sequence of transformations then involves multiplying the respective transformation matrices in the same order. Rotation is next generalized about any point on the plane. The reflection transformation is discussed in two-dimensions. A property of translation, rotation and reflection matrices is that they are *orthogonal* which ensures the preservation of lengths and angles. In other words, the three transformations do not cause any deformation in a rigid body for which reason they are termed *rigid-body transformations*. Those that do affect deformations, i.e., scaling and shear, are discussed next. The aforementioned transformations are extended to use with three-dimensional solids using four-dimensional homogenous coordinates. It may be realized that these transformations help in the Computer Aided Assembly of rigid-body components. For drafting or engineering drawing applications, the geometry of perspective and parallel projections is detailed. A reader would note that the matrix forms of transformations and projections are similar. In addition to conventionally employed first (or third) angle orthographic and isometric projections to pictorially represent engineering components, perspective viewing, oblique viewing and axonometric viewing are also discussed in Chapter 2.

Chapters 3 to 5 are exclusively devoted to the design of curves. Chapter 3 commences by differentiating between curve fitting/interpolation and curve design, the latter is more generic and can be adapted to achieve the former. Among the explicit, implicit and parametric equations to describe curves, the third choice is suited best to accommodate vertical tangents, to ease the computation for intersections (for trimming purposes, for instance), and to represent curve segments by restricting the parameter range in [0, 1]. Unnecessary oscillations in curves from the design viewpoint are undesired for which reason a curve is sought to be a composite one with constituent curve segments of low degree (usually cubic) arranged end to end. The position, slope and curvature continuity at junction points of a composite curve can be addressed via the differential geometry of curves covered in this chapter. Two of the three widely used curve segment models are discussed in Chapter 4. The first is Ferguson cubic segment that requires two end points and two respective slopes to be specified by the user. For a set of data points and respective slopes, a composite Ferguson curve of degree three can be constructed. Its shape can be altered by relocating any one (or more) data point(s) and/or slopes (by changing their magnitudes and/or directions). A Ferguson curve would have the slope continuity through out, however, if one desires curvature continuity, using differential geometry, one can determine that any three consecutive slopes are related. Thus, for a given set of data points and slope information at the start and end points, intermediate slopes can be determined using the constraint equations resulting from curvature continuity. The advantage is two-fold: first, a designer need not specify all slopes which is a higher order information usually difficult for a designer to submit as input. Second, the result is a smooth, curvature continuous cubic Ferguson curve.

Higher order information, like specifying the slopes, can be avoided with Bézier curve segments that are modeled using only data points (also called control points). Bézier segments may be regarded as the geometric extension of the construction of a parabola using the three tangent theorem. The resultant algebraic equation is the weighted linear combination of data points wherein the weights are Bernstein polynomials which, in turn, are functions of the parameter. In parameter range [0, 1], Bernstein polynomials have the property of being non-negative, and that they sum to unity for any value of the parameter. These features render some interesting convex hull and variation diminishing properties to Bézier segments. The shape of the latter can be altered by relocating any data point. However, the effect is global in that the shape of the entire curve is changed. Modeling of continuous Bézier curves is also described using cubic segments. The slope and curvature continuity of composite Bézier curves at junction points restrict the placement of some data points. A designer is constrained to relocate two data points in the neighborhood of the junction point along a straight line for slope continuity. For curvature continuity, four points in the neighborhood of the junction point inclusive, need to be coplanar.

Splines, which are in a manner generalized Bézier curves, are discussed extensively in Chapter 5. The term *spline* is inspired from the draughtman's approach to pass a thin metal or wooden strip through a given set of constrained points called *ducks*. In addition to data points required to construct a spline, a set of parameter values called the *knot vector* is required. Thus, wherein primarily the number of data points determine the degree of Bézier segments, for splines, it is the number of *knots* in the knot vector. Chapter 5 discusses the modeling of polynomial splines which are then normalized to obtain *basis-splines* or *B-splines*. B-splines are basis functions similar to Bernstein polynomials in case of Bézier segments. All B-spline basis functions are non-negative, and only some among those required for curve definition, sum to unity. This renders strong convex hull property to B-Splines which provides the local shape control to a B-spline curve. Newton's divided-difference and the related Cox-de Boor recursive method to compute B-spline basis functions are described in the chapter. Generation of knot vector from given relative placement of data points, and approximation and interpolation with B-spline curves are also discussed.

Chapters 6 and 7 cover surfaces in detail. Like with curves, parametric representation of surfaces is preferred. Also, surfaces are sought as composites of patches of lower degree. There are methods to join together and to *knit* or *weave* such patches at their common boundaries to ensure tangent plane and/or curvature continuity. Chapter 6, thus details the differential geometry of surfaces. Quadric or analytical surface patches are not adequate enough to help design a free-form composite surface. Based on the principles of curve design in Chapters 4 and 5, some basic methods to design a surface patch are described in Chapter 6. These include methods to realize developable and ruled surface patches, parallel surface patches, and patches resulting from revolution and sweep. The shape of such patches can be controlled by relocating the data points and/or slopes used for the ingredient curves. Chapter 7 entails methods of surface patch design that are direct extension of the techniques described in Chapters 4 and 5. Herein, patches are treated under two groups, the tensor product patches and boundary interpolation patches. In the former, Ferguson, Bézier and B-Spline patches are covered while in the latter, bilinear and bi-cubic Coon's patches are discussed. Methods to achieve composite Ferguson, Bézier and Coons patches are also mentioned.

Discussion on curve and surface design lays the foundation for solid or volumetric modeling. Though the treatment is purely geometric when discussing curves and surfaces, it takes more than geometry alone to interpret solids. Any representation scheme for computer modeling of solids is expected to (i) be versatile and capable of modeling a generic solid, (ii) generate valid and unambiguous solids, (iii) have closure such that permitted transformations and set operations on solids always yield valid solids, and (iv) be compact and efficient in matters of information storage and retrieval. Chapter 8 commences with an understanding of solids. The Jordon's theorem establishes that a *closed connected surface* divides the Euclidean space into two subspaces, the space enclosed within the closed surface, which is the interior of a solid, and the space exterior to it. A brief discussion on topology then follows describing homeomorphism, closed-up surfaces, topological classification and invariants of surfaces. The intent is to describe solids topologically and highlight how two geometrically different solids can be topologically similar to use identical modeling methods with different geometry information. In this chapter, three solid modeling techniques, namely, *wireframe modeling*, *boundary representation method* and *Constructive Solid Geometry* are discussed. Wireframe modeling is one of the oldest ways that employs only vertex and edge information for representation of solids. The connectivity or topology is described using two tables, a vertex table that enumerates the vertices and records their coordinates, and an edge table wherein for every numbered edge, the two connecting vertices are noted. The edges can either be straight lines or curves in which case the edge table gets modified accordingly. Though the data structure is simple, wireframe models do not include the facet information and thus are ambiguous.

The boundary representation (B-rep) method is an extension of wireframe modeling in that the former includes the details of involved surface patches. A popular scheme employed is the Baumgart's *winged edge* data structure for representation of solids. Though developed for polyhedrons, the Baumgart's method is applicable to homeomorphic solids. That is, the primary B-rep data structure of a tetrahedron would be the same as that of a sphere over which a tetrahedron with curved edges is drawn. The difference would be that for a sphere, the edges and faces would be recorded as entities with finite curvature. The associated Euler-Poincaré formula is discussed next which is a topological result that ensures the validity of a wide range of polyhedral solids. Based on the Euler-Poincaré formula are the Euler operators for construction of polyhedral solids. Two groups of Euler operators are put to use, the *MAKE* and *KILL* groups for adding and deleting respectively. Euler operators are written as *Mxyz* or *Kxyz* for the Make and Kill groups respectively where x, y and z represent a vertex, edge, face, loop, shell or genus. Using Euler operators, every topologically valid polyhedron can be constructed from an initial polyhedron by a finite sequence of operations.

Constructive Solid Geometry (CSG) is another way for modeling solids wherein *primitives* like block, cone, cylinder, sphere, triangular prism, torus and many others can be combined using Boolean set operations like union, intersection and difference. Solids participating in CSG need not be bounded by analytical surfaces. A closed composite surface created using generic surface patches discussed in Chapters 6 and 7 can also be used to define a CSG primitive. Boolean, regularized Boolean operations and the associated construction trees are discussed in detail in Chapter 8. Other method like the Analytical Solid Modeling which is an extension of the tensor product method for surfaces to three-dimensional parametric space is also mentioned. Chapter 8 ends highlighting the importance of the *parametric modeling* for engineering components. One may require machine elements like bolts of different nominal diameters for various applications wherein parametric design helps. Also, using analysis (Chapter 11) and/or optimization (Chapter 12), one may hope to determine the optimal parameter values of an engineering component for a given application. Chapter 9 highlights some concepts from computational geometry discussing intersection problems and Boolean operations on two-dimensional polygons to consolidate the concepts in constructive solid geometry. Chapter 10 discusses different techniques to model surfaces from a set of given point cloud data, usually encountered in reverse engineering.

That analysis and optimization both play a key role in Computer Aided Design, Chapters 11 and 12 are allocated accordingly. Most engineering components are complex in shape for classical stress analysis methods to be employed. An alternative numerical approach called the Finite Element Method (FEM) is in wide use in industries and elsewhere, and is usually integrated with the CAD software. FEM is a broad field and is a result of an intensive three decade research in various areas involving stress analysis, fluid mechanics and heat transfer. The intent in Chapter 11 is to only familiarize a reader with concepts in FEM related to stress analysis. The Finite Element Method is introduced using springs and later discussed using truss, beam and frame, and triangular and four-node elements. Minimization of total potential is mainly employed when formulating the stiffness matrices for the aforementioned elements.

Chapter 12 discusses various classical and stochastic methods in optimization. Among classical methods, first, *zero-order* (function-based) and *first-order* (gradient-based) methods for objectives with single (design) variable are discussed. These include (a) the bracketing techniques wherein the search is limited to a pre-specified interval and (b) the open methods. Classical multi-variable optimization without and with constraints is discussed next. The method of Lagrange multipliers is detailed, and Karush-Kuhn Tucker necessary conditions for optimality are noted. The Simplex method and Sequential Linear Programming are briefed followed by Sequential Quadratic Programming. Among the stochastic approaches, genetic algorithm and simulated annealing are briefed.

1.7 Computer Aided Mechanism and Machine Element Design

Using existing software, solid models or engineering drawings of numerous components can be prepared. In addition, a computer can also help design machine elements like springs, bearings, shafts and fasteners. It can also help automate the design of mechanisms, for instance. A few familiar examples are presented below in this context, and many more can be similarly implemented.[1,2]

Example 1.1 A Four-Bar Mechanism

Design of mechanisms has been largely graphical or analytical. The vector loop method is a convenient tool in computer solution of planar mechanism problems such as determination of point path, velocity and acceleration. Consider a four-bar mechanism shown in Figure 1.3. OA is the crank (link-2), other links being AB (link-3) and BK (link-4). O and K are fixed to the ground forming the link-1. All joints are pin joints. Assume that the link lengths are known and that the x-axis is along OK and y-axis is perpendicular to OK. All angles are measured positive counterclockwise (CCW) with respect to the x-axis. Regard the vector $\mathbf{r}_1$ attached to the fixed link 1. Similarly, $\mathbf{r}_2$ is attached to the crank link-2 and rotates with it. Vectors $\mathbf{r}_3$ and $\mathbf{r}_4$ are similarly attached to links 3 and 4. These vectors have magnitudes equal to the link lengths to which they are attached and have directions along the instantaneous positions of the links OA, AB, and BK. Let the angle (CCW) as measure of the vector direction for $\mathbf{r}_i$ be θ_i, $i = 1, \ldots 4$. $\theta_1 = 0$ since link OK is fixed and is along the x-axis. Using vector method

$$\vec{OA} + \vec{AB} + \vec{BK} - \vec{OK} = \vec{0}$$

$$\overset{vI}{\mathbf{r}_2} + \overset{v?}{\mathbf{r}_3} + \overset{?v}{\mathbf{r}_4} - \overset{vv}{\mathbf{r}_1} = 0 \tag{1.1}$$

where vI (magnitude and direction) on $\mathbf{r}_2$ indicates that both the magnitude and direction (input) are known, $v?$ on $\mathbf{r}_3$ shows that while the magnitude is known, the direction is yet unknown (and depends upon the present position of $\mathbf{r}_2$), vv indicates given (known) magnitude and direction, and $?v$ shows

[1]Nikravesh, P.E. (1988) Computer Aided Analysis of Mechanical Systems, Prentice-Hall, N.J.
[2]Hall, Jr., A.S. (1986) Notes on Mechanism Analysis, Waveland Press, Illinois.

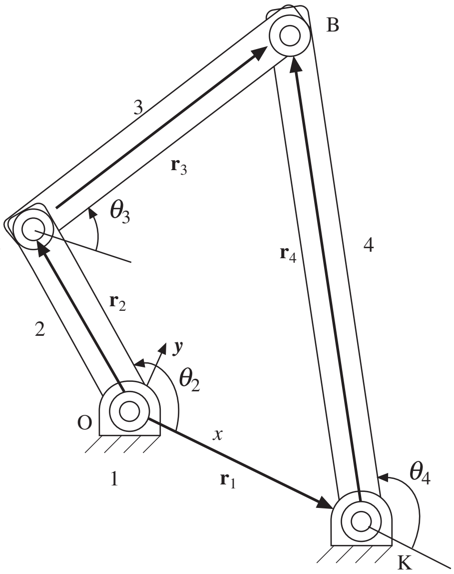

Figure 1.3 Schematic of a four-bar mechanism

unknown magnitude and known direction. The components of the vectors along x and y axes can be expressed as:

$$X : r_2 \cos\theta_2 + r_3 \cos\theta_3 + r_4 \cos\theta_4 - r_1 \cos\theta_1 = 0$$

$$Y : r_2 \sin\theta_2 + r_3 \sin\theta_3 + r_4 \sin\theta_4 - r_1 \sin\theta_1 = 0 \tag{1.2}$$

Here, θ_2 is the known crank angle and $\omega_2 = \dot{\theta}_2$, $\alpha_2 = \ddot{\theta}_2$ are also given. Since $\theta_1 = 0$, Eq. (1.2) is reduced to

$$X : r_2 \cos\theta_2 + r_3 \cos\theta_3 + r_4 \cos\theta_4 - r_1 = 0$$

$$Y : r_2 \sin\theta_2 + r_3 \sin\theta_3 + r_4 \sin\theta_4 = 0 \tag{1.3}$$

Evaluating Link Positions

Eq. (1.3) is nonlinear if they are to be solved for θ_3 and θ_4 for given steps of θ_2. Newton's method converts the problem into an iterative algorithm suitable for computer implementation. Let the estimated values be (θ_3', θ_4'). If the guess is not correct, Eqs. (1.3) will be different from zero, in general. Let the errors be given by:

$$X : r_2 \cos\theta_2 + r_3 \cos\theta_3' + r_4 \cos\theta_4' - r_1 = \varepsilon_1$$

$$Y : r_2 \sin\theta_2 + r_3 \sin\theta_3' + r_4 \sin\theta_4' = \varepsilon_2 \tag{1.4}$$

For small changes $(\Delta\theta_3, \Delta\theta_4)$ in the change in error $(\Delta\varepsilon_1, \Delta\varepsilon_2)$ is given by the Taylor's series expansion up to the first order. That is

$$\Delta\varepsilon_1 = \frac{\partial\varepsilon_1}{\partial\theta_3'}\Delta\theta_3 + \frac{\partial\varepsilon_1}{\partial\theta_4'}\Delta\theta_4 = -r_3 \sin\theta_3'\,\Delta\theta_3 - r_4 \sin\theta_4'\,\Delta\theta_4$$