Res Eng Des (1992) 4:75–87

Research in Engineering Design
Theory, Applications, and
Concurrent Engineering

Conceptual Design of Mechanisms Based on Computational Synthesis and Simulation of Kinematic Building Blocks

Sridhar Kota and Shean-Juinn Chiou
Department of Mechanical Engineering and Applied Mechanics, The University of Michigan, Ann Arbor, Ann Arbor, Michigan, USA

Abstract. Although many ingenious mechanisms have been designed, the fundamental task of conceptualizing these devices is, to a great extent, still an art. While sophisticated computational tools for dynamic analysis of mechanisms exist, hardly any computational methods exist for generalized synthesis. To develop a computational model for synthesis, a formal foundation for mechanisms design must be laid by rationalizing the process of mechanical synthesis. Rationalization in synthesis implies that complex mechanical motions can be described in terms of primitives or building blocks. In this paper, we present a matrix methodology that forms the basis for a computable approach to design synthesis. In this methodology, the continuous design space of a mechanisms domain is discretized into functional subspaces, and each subspace is represented uniquely by a conceptual building block. The matrix scheme serves as a formal means to (a) represent and reason with the building blocks at different levels of abstraction, (b) generate alternate conceptual design configurations, and (c) facilitate rapid simulation of design concepts by connecting a series of building blocks.

1 Introduction

1.1 Background

Myriad ingenious mechanisms, including toys, machine tools, automation equipment, cameras, copy machines, and automobiles, have been designed over the past 100 years [1, 20]; however, the fundamental task of conceptualizing how these devices perform the desired motions is, to a great extent, still an art. Once these mechanisms are built, their kinematics are not difficult to understand, but understanding how they were conceptualized is. One of the most challenging questions that faces anyone who attempts to automate the design process is: What makes these mechanisms ingenious, and how did these designs originate? Addressing these issues is like trying to pick the brains of the inventors who designed the mechanisms. Two approaches to this problem have been developed in the course of a long history. The first approach is the creation of *atlases of mechanisms* grouped according to function. These remain the primary source of ideas. The second approach involves an *abstract representation* of the structure of mechanisms similar in spirit to the symbolic representation of chemical compounds in the field of chemistry.

1.2 Indirect vs. Direct Design Synthesis

Indirect synthesis of mechanisms using either atlases or abstract representation of building blocks is necessary because a design cannot be synthesized directly. To perform direct synthesis and optimization for a given set of design specifications, an analytical solution should exist that provides the joints and the dimensions of all the elements of the system. The inherent motion characteristics of mechanisms are too difficult to express analytically, and the same motion can be obtained several different ways, ruling out any hope for direct synthesis methods. Therefore, a systematic classification of various solution principles through abstractions is necessary in order to conceptualize alternate working solutions to a given design task.

1.3 Early Research

Reuleaux was among the first to attempt a classification scheme and to study machinery design systematically [32]. He found that a machine consists of a limited number of parts occurring over and over, and he called these parts constructive elements. Some that he identified are screws and screwed joints, keys, rivets, bearings, pins, shafts, couplings, belts, cords and ropes, gears and gear trains, flywheels, levers, ratchet wheels, and springs. In addi-

Offprint requests: Design Laboratory, Department of Mechanical Engineering and Applied Mechanics, The University of Michigan, Ann Arbor, Ann Arbor, MI 48109-2125, USA tion to identifying the elements, Reuleaux also assigned them symbols in an attempt to mimic symbols in chemistry. His symbolism did not include all the variables necessary for analysis, so his attempts were not successful. The subject of abstract representation of the basic building blocks of machines fell into neglect for almost a hundred years. His symbolism work was not fully exploited for systematic synthesis of mechanisms even with the advent of computers. Other researchers, however, did carry on Reuleaux's attempts to identify building blocks using different names for them, including details, elements, and simple-parts. These classifications consisted of ad hoc labeling and did not provide a formal definition of why a part should be designated as a basic building block; therefore, they were not useful in systematic synthesis of complex machines.

1.4 Graph Theory

Beginning in the mid-1960s, the abstract representation of kinematic structure was investigated with the aid of graph theory first by Freudenstein and Maki [14, 15]. This procedure is based on the separation of the mechanism structure from its function. The mechanism structure can be enumerated in an essentially systematic, unbiased fashion. The method helps establish the total number of mechanisms in a given class given the number and type of joints. The ability to enumerate all possible kinematic topologies using graph theory lends itself to development of expert systems [5, 17, 22, 27, 30, 34, 35]. The kinematic chains that are enumerated using graph theory are based purely on topological considerations. The desirable motion characteristics of mechanisms are too complicated to be comprehended by evaluating the kinematic chains. By assigning different dimensions to individual links, entirely different motion characteristics can be obtained by mechanisms derived from a single kinematic chain.

Typically, a designer wishes to specify the desired behavior in terms of motion characteristics such as: degrees-of-freedom, sequence of output motions, nature of output motions (translational, rotational, etc.), whether the input–output relationship is linear or nonlinear, and continuous or intermittent, unidirectional or bidirectional, reciprocation or oscillation, and so on. One of the limitations of graph theory is that the enumeration procedure is based primarily on structural and topological considerations. Graph theory accounts mainly for the degrees of freedom and structural constraints such as whether the input link is connected to frame or not. Such considerations alone do not reflect the desired behavior. Therefore, mechanisms must be characterized from several different viewpoints, such as a mechanisms's function and its operational constraints. The goal of this work, however, is to develop an automated synthesis procedure with desired behavior as the starting point.

In spite of its limitations, graph theory is still a useful and a practical technique for systematic enumeration of alternate kinematic chains, especially if an initial mechanism configuration is known a priori. It is particularly useful in patient recognition. Numerous practical applications of graph theory have been developed [6, 13, 16, 35]. A detailed account of various techniques for creative design and type synthesis of mechanisms is given in [10].

Recently, researchers in computer science have investigated qualitative theories for analyzing the behavior of mechanisms and have qualitatively analyzed the kinematic behavior of mechanisms from the shape and the initial positions of its parts [11, 21]. Forbus proposed symbolic place vocabularies for qualitative spatial reasoning to capture the geometric interactions between physical objects [12]. His research goal was to develop methods for qualitative descriptions of motion sufficient to understand a mechanism. His analysis is restricted to two-dimensions. The majority of applications of qualitative reasoning in kinematics have been in the analysis rather than the synthesis of mechanical motions.

The methodology presented in this paper is an alternate approach to type synthesis of mechanisms. It compliments the graph theory approach by addressing a higher-level synthesis task. Mechanism solutions generated by the matrix methodology could serve as a starting point for graph theory based enumeration of alternate mechanisms.

1.5 Rational Classification and Matrix Methodology

The work described in this paper provides a system of indirect design synthesis. It provides a rational classification scheme for the constructive elements based upon their kinematic nature. In an earlier paper, we presented a qualitative classification scheme and computerized catalogs that we developed for a subset of mechanisms [23–25]. This classification scheme is in contrast to the earlier ad hoc methods of labeling that cannot be used for synthesis. Representation methods presented in this paper are based on functions and operating constraints—notions that are natural to human designers. The work described in this paper uses computational symbolic matrices for synthesis and simulation of a wide variety of mechanisms based on the work of Denavit and Hartenberg [7]. The matrix methodology, based on rationally classified design building blocks, is not only intuitive but also computable. It also covers a wide range of mechanisms, such as cams and followers, gear trains, linkages, ratchets, and screw mechanisms. Our method allows for computational synthesis of design.

2 Computational Synthesis

2.1 *Discretization of Design Space: Basis for Computational Synthesis*

By identifying abstract building blocks, we will chunk the subspaces inside the continuous design space. Each building block represents, in an abstract way, a subspace of the entire design space. Once the most promising subspace is identified for a given design problem, we can use mathematical models or design rules (modification rules) to vary the design parameters within the subspace. We have successfully implemented this methodology in a Dwell-Expert program for a subset of mechanisms [23–25]. Hoeltzel and Chieng [18] have developed a systematic approach to pattern classification of motion curves and automated the design process using neural networks. The notion of using building blocks to describe the domain and using modification rules to generate variations of the building blocks is like piecing together the design subspaces that were originally discretized. Such discretization and recombination is an essential part of the conceptual design automation process.

2.2 *Discretization in the Design Process*

To accomplish computational synthesis in the design process, the requirements of a particular design must be discretized into functions and constraints. The goal of the design is to fulfill first the functional requirements and then satisfy the constraints. So, design alternatives based on function are filtered through the constraints to select a final design.

Not only must the requirements of the design be discretized but so must the design knowledge. Design knowledge is discretized into functional and physical building blocks [26]. Functional building blocks describe only primary functions without consideration of constraints, physical descriptions, or manufacturing processes. Physical building blocks describe a set of physical artifacts that fulfill a specific primary function and include all the information omitted for functional building blocks. Functional building blocks can be decomposed into alternatives and components. Physical building blocks are decomposed by alternative alone. This allows us to map abstract functions to alternate structures or physical artifacts.

The concept of mapping functions to structures in conceptual design is shared by many researchers [2, 8, 18, 19, 29, 31, 33]. Computational models for configuration design are investigated in [26, 28]. All of these researchers have found that the nature and the granularity of the building blocks are important issues in any automated design system. If the granularity of the building blocks is too fine, combinational explosion occurs. If the granularity is too coarse, novel synthesis of building blocks will be hindered. Therefore, the challenge in defining building blocks is to determine a proper granularity that precludes combinatorial explosion and permits synthesis of new configurations.

Function-to-structure maps are based on physical working principles. During the function decomposition process, it may be possible to discover more than one way to decompose a given function into subfunctions or components. These, then, are alternative function-level solutions. Each should lead to a different design configuration. This functional design methodology has been developed and implemented for hydraulic systems [26]. With an understanding of the nature of the building blocks of motion, this methodology can be extended to the general mechanisms domain.

2.3 *Cam Synthesis as an Example of Discretization and Recombination*

Consider a classical example of mechanical cam-design process where discretization takes place implicitly. The geometry of a cam profile dictates, to a great extent, the function of a cam. The function of a cam is described in terms of the displacement of the mating follower. The follower displacement is a continuous function. However, the desired functional relationship, including first-, second-, and higher-order derivatives, can be approximated by a piecewise combination of standard library functions. These functions can be, for example, cycloidal, harmonic, trapezoidal, or polynomial, using only a finite set of standard mathematical functions and follower-types. One can systematically synthesize the entire cam profile by piecing together the individual profile-segments. Therefore, conceptually, one can create numerous output motions by novel combinations of finite building blocks.

3 Computational Synthesis of Motion

Traditionally, kinematic synthesis tasks are classified as: motion generation, path generation, and function generation. In practice, motion generation usually is restricted to a finite number of positions at which the designer desires to control the orientation of output members. In path generation, a desired path, or discrete points along a desired path, are considered. The synthesis task involves configuring mechanism solutions in which the endpoint traces the desired path. Function generation involves synthesis of mechanisms that satisfy the prescribed input–output motion relationships. Computer-based design tools for these synthesis tasks, such as LINCAGES [9], perform dimensional synthesis but not type synthesis. These tools dimension the individual links in a presupposed mechanism topology; they do not address the issue of determining the most suitable type of mechanism for given design task. In the traditional classification of the phases of a design process, type synthesis can be viewed as conceptual design and dimensional synthesis as parameter design. The matrix scheme presented in this paper addresses conceptual design of mechanisms for function-generation tasks.

3.1 Function Generation: Problem Description

- *Given*: A description of input and output motion and a set of constraints, such as kinematic and dynamic operational constraints, cost constraints, and reliability.
- *Find*: One or more mechanism configurations that fulfill the prescribed functional requirements within the constraints. The configuration should include descriptions of one or more mechanism types and a feasible range of parameters that control the performance characteristics.

The function-generation task involves synthesis of a mechanical device to satisfy a desired motion transformation from the input to the output. For instance, the input may be a uniform rotational motion, and the output may be an oscillatory motion through a certain range with a specified dwell period during a portion of the cycle. The input and the output are generally considered to have a fixed axis. Mappings from functional specifications to mechanism types are many-to-many. In a simple function-generation task in which the input is rotational motion and the output is translational motion, the possible candidate mechanism types are slider crank, rack-and-pinion, screw, rope and pulley, or cam-follower systems. The choice of a particular type depends on the constraints imposed by the design specifications. For example, the choice may depend

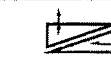
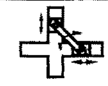
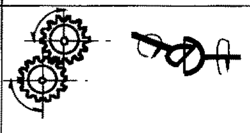
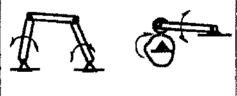
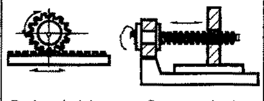
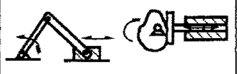
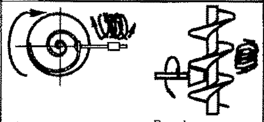
Functional Building Blocks	*Physical Building Blocks*	
	Linear I/O relationship	Non-Linear I/O relationship
Translation <--> Translation	 Simple wedge	 Double slider
Rotation <--> Rotation	 Gear-pair Universal Joint	 Four-bar linkage Cam- Oscillating follower
Rotation <--> Translation	 Rack-and-pinion Screw mechanism	 Slider-crank Cam - translating follower
Rotation <--> Helicazl	 Spiral gear Bevel-screw gear	

Fig. 1. A taxonomy of functional and physical building blocks.

on whether the output and the input must be interchangeable or whether the output must be bidirectional. Regardless of the constraints, the fundamental design requirement of the artifact is often purely kinematic. Therefore, it is important to classify all function-generating systems according to input–output motion relationships (i.e., rotational motion, translational motion, and helical motion). This leads to the hierarchical structure shown in Fig. 1. A more detailed account of functional and physical building blocks must be developed.

The functional design building blocks for the kinematic synthesis of mechanical systems corresponds to each of the leaf-nodes of the decomposition hierarchy. Figure 1 shows some of the physical building blocks. Corresponding to each functional building block, a physical building block has been identified. For example, corresponding to the linear rotation <-> translation functional building block, a rack-and-pinion primitive mechanism has been identified. The form of the rack-and-pinion as shown in Fig. 1 should be treated conceptually. Each primitive mechanism has concept variants, physical variations of the same solution principle (see Fig. 2). Although the devices shown in Fig. 2 have either multiple racks or multiple pinions or are used in conjunction with other primitive elements, the underlying design concept remains the same in all the devices. The notion of concept variants permits the grouping of numerous designs into a single design concept and thereby helps discretize the design space into building blocks.

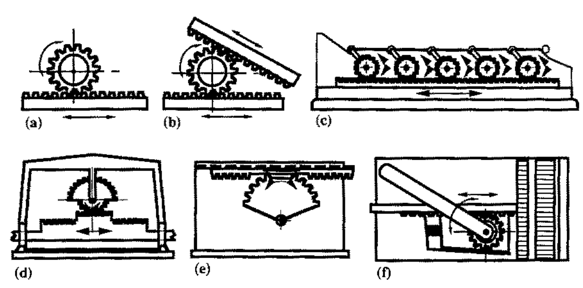


Fig. 2. Variants of the rack-and-pinion concept. (a) standard rack-and-pinion, (b) multiple racks, (c) multiple pinion-based transfer mechanism, (d) step-wise motion due to modified interface, (e) dwell motion due to modified interface, and (f) a clamping device—rack-and-pinion building block in conjunction with a spring and a wedge.

3.2 Concept Variants

The building blocks derived from the function decomposition hierarchy represent standard physical forms. Some of the variations of the standard form of rack-and-pinion are also shown in Fig. 2. It is important to note that each of the building blocks identified represents a "solution-concept" rather than any particular physical form. That is, "rack-and-pinion" is a solution-concept for transformation between rotation about a fixed axis and translation. The concept itself does not preclude, say using multiple racks or multiple pinions. Many other variations may exist. This leads to the notions of standard form and modification operators. The modification operators transform the standard form into a suitable variation of the conceptual building block. The standard form and its variations are all essentially different manifestations of the same working principle. The modification operators when applied to the standard form change the behavior of the system, retaining, however, the underlying concept. Examples of some modification operators are given here:

1. Duplicate the interacting elements of the building block. (Fig. 2b and c)
2. Change the form of the interface. The tooth geometry is modified to obtain step-wise motion (Fig. 2d) and dwell motion (Fig. 2e).
3. Add one or more primitive elements (wedge, spring, lever, etc.). Figure 2f shows a clamping device based on this modification.
4. Apply a kinematic inversion. For instance, this rule modifies a standard gear-pair into a planetary gear system.

A generic set of modification operators should be defined for each building block to enable the computer to create appropriate variations of the solution principle—that is, the concept represented by a building block. The notion of using a limited set of building blocks to describe the entire domain of mechanisms does not abandon flexibility. In fact, the notion of modification operators complements the notion of a limited set of building blocks and provides the flexibility to explore physical variations that are typical in the continuous design space of the mechanical world.

4 Qualitative Matrix Representation of the Building Blocks

Conceptual design is based on an appropriate means of *abstracting functionality,* whether it be the simple functionality of an individual mechanical widget or the complex functionality of an electromechanical system. In our approach to design synthesis, the functionality of a primitive mechanism is expressed as a concatenation of matrices, including a motion transformation matrix (MTM) and a sequence of constraint matrices. For example, a *rack-and-pinion* building block is expressed as:

$$\underset{\text{Output}}{\begin{Bmatrix} R_{xo} \\ R_{yo} \\ R_{zo} \\ T_{xo} \\ T_{yo} \\ T_{zo} \end{Bmatrix}} = \left\{ \underset{\substack{\text{Function matrix} \\ \text{(MTM)}}}{\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}} \right.$$

$$\text{Basic constraint matrices}$$

$$+ \begin{bmatrix} 0 & 0 & 0 & 0 & \leftrightarrow & \leftrightarrow \\ 0 & 0 & 0 & \leftrightarrow & 0 & \leftrightarrow \\ 0 & 0 & 0 & \leftrightarrow & \leftrightarrow & 0 \\ 0 & \leftrightarrow & \leftrightarrow & 0 & 0 & 0 \\ \leftrightarrow & 0 & \leftrightarrow & 0 & 0 & 0 \\ \leftrightarrow & \leftrightarrow & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\left. + \begin{bmatrix} 0 & 0 & 0 & 0 & + & + \\ 0 & 0 & 0 & + & 0 & + \\ 0 & 0 & 0 & + & + & 0 \\ 0 & + & + & 0 & 0 & 0 \\ + & 0 & + & 0 & 0 & 0 \\ + & + & 0 & 0 & 0 & 0 \end{bmatrix} \right\} \underset{\text{Input}}{\begin{Bmatrix} R_{xi} \\ R_{yi} \\ R_{zi} \\ T_{xi} \\ T_{yi} \\ T_{zi} \end{Bmatrix}}$$

The first 6 × 6 matrix is the MTM which is constructed in order to abstract the couplings between the output rotations and translations (the R_o's and T_o's) and the respective input rotations and translations (the R_i's and T_i's) for x, y, and z coordinates. The ones in the matrix represent all the motion transformations that are possible with this class of devices. The transformation T_y <-> R_z is just one of them. The other two 6 × 6 matrices are constraint matrices. The first of these indicates the reversibility or irreversibility of the allowed couplings (denoted

○ : Continuous motion	C : Constant transmission ratio
◌ : Intermittent or dwell motion	V : Variable transmission
<----> : Input & output are interchangeable	n : Cycle ratio between the input & output
+ : Output is in the same direction as input	± : Reciprocating / Oscillating

3

Symbolic Algebra :

○ * ○ = ○	C * C = C	<----> * <----> = <---->
○ * ◌ = ◌	C * V = V	<----> * <--/-- = <----
◌ * ◌ = ◌	V * V = V	(ratchet)

4

Fig. 3. Operational constraints.

Fig. 4. Operational constraints in matrix form.

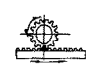
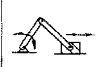
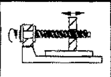
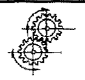
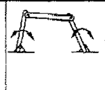
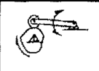
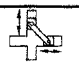
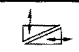
FBB	MTM	Physical Building Blocks & Their Constraint Matrices		
R↔T	$R_X \leftrightarrow T_Y$ $\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}$	 $\begin{pmatrix} \bigcirc&0&0&0&0\\ 0&+&0&0&0\\ 0&0&V&0&0\\ 0&0&0&\leftrightarrow&0\\ 0&0&0&0&n \end{pmatrix}$	 $\begin{pmatrix} \bigcirc&0&0&0&0\\ 0&\pm&0&0&0\\ 0&0&V&0&0\\ 0&0&0&\leftrightarrow&0\\ 0&0&0&0&1 \end{pmatrix}$	 $\begin{pmatrix} \bigcirc&0&0&0&0\\ 0&\not\pm&0&0&0\\ 0&0&V&0&0\\ 0&0&0&\leftarrow&0\\ 0&0&0&0&n \end{pmatrix}$
R↔R	$R_Z \leftrightarrow R_Z$ $\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&1&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}$	 $\begin{pmatrix} \bigcirc&0&0&0&0\\ 0&-&0&0&0\\ 0&0&C&0&0\\ 0&0&0&\leftrightarrow&0\\ 0&0&0&0&n \end{pmatrix}$	 $\begin{pmatrix} \bigcirc&0&0&0&0\\ 0&\not\pm&0&0&0\\ 0&0&V&0&0\\ 0&0&0&\leftrightarrow&0\\ 0&0&0&0&1 \end{pmatrix}$	 $\begin{pmatrix} \bigcirc&0&0&0&0\\ 0&\pm&0&0&0\\ 0&0&V&0&0\\ 0&0&0&\leftarrow&0\\ 0&0&0&0&n \end{pmatrix}$
T↔T	$T_X \leftrightarrow T_Y$ $\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&1&0&0\\ 0&0&0&0&0&0 \end{bmatrix}$	 $\begin{pmatrix} \bigcirc&0&0&0&0\\ 0&-&0&0&0\\ 0&0&V&0&0\\ 0&0&0&\leftrightarrow&0\\ 0&0&0&0&1 \end{pmatrix}$	 $\begin{pmatrix} \bigcirc&0&0&0&0\\ 0&-&0&0&0\\ 0&0&C&0&0\\ 0&0&0&\leftarrow&0\\ 0&0&0&0&1 \end{pmatrix}$	

Fig. 5. A taxonomy of motion building blocks and their matrix representations. FBB, functional building block; MTM, motion transformation matrix.

by the symbol <->), which in this simple case are all reversible. The other indicates whether the responses to unidirectional inputs result in unidirectional (or in the case of rotation, unichiral) or bidirectional (bichiral) outputs, as indicated by the presence of singly or doubly signed matrix elements corresponding to the appropriate couplings. As indicated by the 1s in the MTM, a generalized rack-and-pinion can be used to convert a translational motion along any direction within the x–y plane into rotational motion about the z-axis. The operational constraints are expressed symbolically in Fig. 3. The operational constraints are also expressed in matrix form (Fig. 4). Concatenation of the constraint matrices is based on symbolic manipulation of individual constraints according to the rules of symbolic algebra.

In a computational design system for conceptual design, the MTMs and constraint matrices provide a rationalization of the possible functions of mechanisms. The conceptual design process is initiated with the statement of the design task that describes the input signal and desired output motion. The prescribed task is then decomposed into a series of subfunction motion transformation matrices. Each mode of decomposition leads to a unique design configuration. The decomposed subfunction matrices are then matched against a library of matrices (MTMs and constraints) representing existing devices in the knowledge base. Thus, the computable process of systematic decomposition and matrix-matching constitute the reasoning processes involved in conceptual design. At the lower levels of abstraction, discussed in Section 5, the symbolic elements of candidate matrices are replaced by the actual design parameters before performing qualitative simulation of the designed artifact. The top-level matrix representation of building blocks consisting of a string of function and constraint matrices is unique, but the function matrices of building blocks need not be unique. However, in conjunction with constraint matrices, the overall representation will be unique for each building block. Figure 5 shows a taxonomy of motion building blocks and their matrix representations. As indicated in Fig. 5, although multiple physical building blocks, rack-and-pinion, slider-crank, and screw mechanism in row 1, share a common functional building block (FBB), individual constraint matrices are different. Therefore, a functional matrix, in conjunction with a particular constraint matrix, provides a unique representation for each physical building block. Note that although the function matrix of the slider-crank and the rack-and-pinion are the same, their constraint matrices are not. Thus, the initial set of candidate building blocks that satisfies the basic motion requirements is filtered by imposing the basic motion constraints.

4.1 Different Levels of Representation

The matrix representation method allows for reasoning with the abstract function of the primitive mechanisms without cluttering the reasoning process with unnecessary details. By investigating the nonzero terms in the motion transformation matrices, a computer program can select possible candidate mechanisms that can fulfill the basic kinematic function and constraints. The matrix representation allows for inclusion of additional matrices to represent the dynamic and cost constraints. The next lower-level of abstraction contains design parameters for input–output relationships. These matrices are called parameter matrices and are used in qualitative simulation, which is explained in Section 5.

4.2 Matrix Operations for Task Decomposition

A given design task is first formulated as a set of system specification matrices. This set consists of desired motion transformation matrix and operational constraint matrix. The first step in our computational synthesis procedure is to match desired MTM against building block MTMs. If a suitable match is not found, the desired MTM is then decomposed into a series of subfunction matrices. The decomposition process can be automated by performing matrix manipulations. Based on our preliminary research, we have identified three methods of decomposition using three different types of matrix manipulation techniques: column shifting, row shifting, and a decomposition matrix.

The goal of these matrix manipulation techniques is to transform or relocate ones in the desired MTM in such a way that resulting MTMs (or subfunction MTMs) match one or more building block MTMs. Matrix operators perform these transformations.

1. *Column shifting*: To shift column i to column j in the original MTM, postmultiply the original MTM by another matrix that has a nonzero element only in its ith row and jth column as shown below.

$$\underset{H_{yz} <\text{--}> R_z}{\overset{\text{Original MTM}}{\begin{bmatrix} 0&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&1&0&0&0&0 \end{bmatrix}}} * \underset{H_{yz} <\text{--}> R_y}{\overset{\text{Basic operator}}{\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&1&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}}}$$

$$= \underset{R_y <\text{--}> R_z}{\overset{\text{New MTM}}{\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&1&0&0&0\\ 0&0&1&0&0&0\\ 0&0&0&0&0&0\\ 0&0&1&0&0&0\\ 0&0&1&0&0&0 \end{bmatrix}}}$$

Physically, this matrix decomposition implies that to transform a rotational motion about the Z-axis (R_z) into a helical motion in the x–y plane (H_{yz}), one could transform the rotational motion about the z-axis first into a rotational motion about the y-axis (R_y) and then transform R_y into H_{yz}.

2. *Row shifting*: To shift row i to row j in the original MTM, premultiply the original MTM with another matrix that has its only nonzero element occurring in jth row and ith column as shown below.

$$\underset{T_y <\text{--}> H_y}{\overset{\text{Basic operator}}{\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&1&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}}} * \underset{T_y <\text{--}> R_x}{\overset{\text{Original MTM}}{\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&1&0&0&1&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}}}$$

$$= \underset{R_z <\text{--}> H_y}{\overset{\text{New MTM}}{\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&1&0&0&1&0\\ 0&0&0&0&0&0 \end{bmatrix}}}$$

3. *Decomposition matrix*: A given matrix can be decomposed by starting with any of its non-zero elements. If a non-zero element (i, j) is selected as a starting point, the process of decomposition takes the following steps:

   Form a new matrix A with all of its elements zero except the original nonzero elements in column j.

   Form a decomposition matrix B with B_{ji} as its only non-zero element

   Form a matrix C with all of its elements zero except the original non-zero elements in row i.

   The original matrix is [A]*[B]*[C]. Matrices A and C can be recursively decomposed, if necessary, using any of the three matrix manipulation techniques described above.

An example of use of decomposition matrix for decomposing an MTM is shown below with element (4,1) as the starting point

$$\underset{H_{xy} <\text{--}> H_{xz}}{\overset{\text{Original matrix}}{\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 1&0&1&0&0&0\\ 1&0&1&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}}} = \underset{H_{xy} <\text{--}> R_x}{\overset{\text{Matrix A}}{\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 1&0&0&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}}}$$

Table 1.

Prescribed motion characteristics	Input	Output
Type of motion	Rotation	Translation (reciprocating)
Orientation	X-axis	X-axis
Continuity	Continuous	Continuous
Direction	+	±
Linearity	Not specified	Not specified
Reversibility	←	
Periodicity	Not specified	Not specified

$$\underset{R_x <\text{--}> T_x}{\overset{\text{Matrix B}}{\begin{bmatrix} 0&0&0&1&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}}} * \underset{T_x <\text{--}> H_{xz}}{\overset{\text{Matrix C}}{\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 1&0&1&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}}}$$

Usually there are several ways in which a matrix can be decomposed. Depending on the mode of decomposition, alternate solutions are generated. The design example given in the next section illustrates this point.

5 Design Example

The task is to design a mechanism that converts a uniform rotational motion about x-axis into a reciprocating motion along the same axis. The output and the input should not be interchangeable (see Table 1)

Step 1: Form the system specification matrices

(a) Form or derive the desired MTM from the specifications of the type of motion and orientation.

$$[\text{Output matrix}] \cdot [\text{Input matrix}]^T = [\text{desired MTM}]$$

$$\begin{bmatrix} 0\\0\\0\\1\\0\\0 \end{bmatrix} * [1\ 0\ 0\ 0\ 0\ 0] = \begin{bmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}$$

(b) Form the System Constraint Matrix, given that the operational constraints are formulated in a matrix form (Fig. 6).

Step 2: Match the desired MTM against the MTMs of the building blocks. In this case, no match is found.

$$\begin{bmatrix} \mathrm{O}&0&0&0&0\\ 0&\pm&0&0&0\\ 0&0&\mathrm{C/V}&0&0\\ 0&0&0&<\text{----}&0\\ 0&0&0&0&\mathrm{n} \end{bmatrix}$$

Fig. 6. System constraint matrix.

Step 3: Decompose the desired MTM into a series of subfunction matrices:

$$\overset{\text{Original matrix}}{\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}} = \overset{\text{Subfunction 1}}{\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}} * \overset{\text{Subfunction 2}}{\begin{bmatrix} 0&0&0&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}}$$

Step 4: Check if the new series of MTMs match any existing building block MTMs.

$$\overset{\text{Subfunction 1}}{\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}} \Rightarrow \overset{\text{Candidate building blocks}}{\begin{array}{l} \text{Slider-crank}\\ \text{Cam-follower}\\ \text{Rack-pinion} \end{array}}$$

$$\overset{\text{Subfunction 2}}{\begin{bmatrix} 0&0&0&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}} \Rightarrow \overset{\text{Candidate building blocks}}{\begin{array}{l} \text{Worm-gear}\\ \text{Bevel-gear} \end{array}}$$

Step 5: Check the constraint matrices.

(a) Write the constraint matrices for each of candidate building blocks (Fig. 7).

(b) Concatenate building block constraint matrices in the sequence in which the original MTM was decomposed. Since three building blocks matched the first subfunction MTM and another two building blocks matched the second subfunction, we have a total of six (3 × 2) combinations leading to six different mechanism configurations. Constraint matrices for each of the

Slider-Crank :

$$\begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & \pm & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & \longleftrightarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix}$$

Rack-Pinion :

$$\begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & + & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & \longleftrightarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix}$$

Cam-Follower :

$$\begin{bmatrix} \% & 0 & 0 & 0 & 0 \\ 0 & \pm & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & \longleftarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix}$$

Worm-Gear :

$$\begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & \not\pm & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & \longleftarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix}$$

Bevel-Gear :

$$\begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & - & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & \longleftrightarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix}$$

Fig. 7. Constraint matrices.

[Configuration 1 = [Slider-Crank] * [Worm-Gear]

$$\begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & \pm & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & \longleftarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix} = \begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & \pm & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & \longleftrightarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix} * \begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & \not\pm & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & \longleftarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix}$$

Configuration 2 = [Slider-Crank] * [Bevel-Gear]

$$\begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & \pm & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & \longleftrightarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix} = \begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & \pm & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & \longleftrightarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix} * \begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & - & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & \longleftrightarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix}$$

Configuration 3 = [Rack-Pinion] * [Worm-Gear]

$$\begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & \not\pm & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & \longleftarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix} = \begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & + & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & \longleftrightarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix} * \begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & \not\pm & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & \longleftarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix}$$

Fig. 8. Constraint matrices—configurations 1–3.

Configuration 4 = [Rack-pinion] * [Bevel-Gear]

$$\begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & - & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & \longleftrightarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix} = \begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & + & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & \longleftrightarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix} * \begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & - & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & \longleftrightarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix}$$

Configuration 5 = [Cam-Follower] * [Worm-Gear]

$$\begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & \pm & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & \longleftarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix} = \begin{bmatrix} \% & 0 & 0 & 0 & 0 \\ 0 & \pm & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & \longleftarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix} * \begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & \not\pm & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & \longleftarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix}$$

Configuration 6 = [Cam-Follower] * [Bevel-Gear]

$$\begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & \pm & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & \longleftarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix} = \begin{bmatrix} \% & 0 & 0 & 0 & 0 \\ 0 & \pm & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & \longleftarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix} * \begin{bmatrix} \circlearrowright & 0 & 0 & 0 & 0 \\ 0 & - & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & \longleftrightarrow & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix}$$

Fig. 9. Constraint matrices—configurations 4–6.

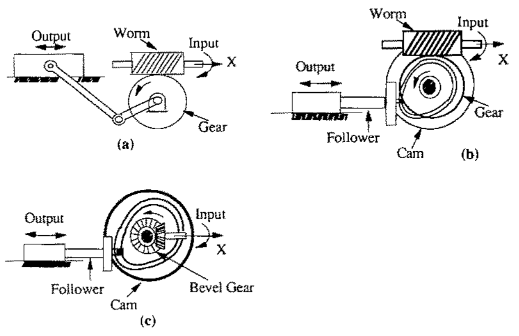


Fig. 10. Three alternate mechanism configurations generated by matrix decomposition. (a) slider-crank and worm-gear; (b) cam-follower and worm-gear; and (c) cam-follower and bevel-gear.

configurations are generated by using symbolic algebra (Figs. 8 and 9).

(c) Identify potential solutions—that is, the combinations of building block constraint matrices that match the system constraint matrix. Of the six combinations, only three configurations (1, 5, and 6) satisfy the requirements after comparing all six constraint matrices with the specified system constraint matrix. These configurations form three alternate solutions to the given task and are shown in Fig. 10.

6 Qualitative Simulation Using Matrix Representation

While the process of synthesizing building blocks is accomplished by selecting an appropriate set of subfunction matrices, we are still left without a way to visualize how the designed system functions. Therefore, the next task is to enable simulation of the conceptual design using a simplified model. Although large-scale computer programs such as ADAMS [3], are available and could be used, these are too complex for our purposes, since a simple and approximate analysis is all that is required during initial phases of design. Besides, all the pertinent information to perform a thorough dynamic analysis is not usually available at the initial stages. Therefore, a qualitative simulation method, which simply combines the building blocks that are selected during initial phases of design to show the input–output motion relationships, is needed to help evaluate conceptual designs. It is qualitative in the sense that it hides the details of intermediate joints and links and simply illustrates how the output member moves in relation to the motion of the input. The output of the first building block will be automatically treated as the input to the second building block and so on. Note that the sequence in which the building blocks physically combine follows exactly the same sequence in which the original desired motion function is decomposed into subfunctions.

The qualitative simulation can be performed by combining lower-level parametric matrices. The parametric matrix representation for the relative motion between rack-and-pinion is given by the following matrix:

$$\begin{Bmatrix} \Delta R_{xo} \\ \Delta R_{yo} \\ \Delta R_{zo} \\ \Delta T_{xo} \\ \Delta T_{yo} \\ \Delta T_{zo} \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{\cos\varphi}{r} & \frac{\sin\varphi}{r} \\ 0 & 0 & 0 & \frac{\sin\varphi}{r} & 0 & \frac{\cos\varphi}{r} \\ 0 & 0 & 0 & \frac{\cos\varphi}{r} & \frac{\sin\varphi}{r} & 0 \\ 0 & r\sin\varphi & r\cos\varphi & 0 & 0 & 0 \\ r\cos\varphi & 0 & r\sin\varphi & 0 & 0 & 0 \\ r\sin\varphi & r\cos\varphi & 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \Delta R_{xi} \\ \Delta R_{yi} \\ \Delta R_{zi} \\ \Delta T_{xi} \\ \Delta T_{yi} \\ \Delta T_{zi} \end{Bmatrix}$$

This matrix representation gives the output motion (e.g., rack ΔT_{yo}) given an incremental input (e.g., ΔR_{zi} to the pinion). To combine the rack-and-pinion building block to the next building block (e.g., slider-crank), we need to represent each of the building blocks in relation to a Global Reference Frame (GRF). The parametric matrix transformation to GRF is derived below (see Fig. 11)

For a pinion input and a rack output in the X–Y plane, the vector T_o is given by

$$\vec{T}_o = \vec{T}_i + \vec{n} + \vec{S} \qquad (1)$$

All the vectors in Equation (1) are defined respect to the global coordinate frame X–Y.

$$\vec{n} = \begin{Bmatrix} 0 \\ -r \end{Bmatrix} \text{ and } \vec{S} = \begin{Bmatrix} rR_{zi} \\ 0 \end{Bmatrix}$$

The vectors above are defined with respect to the local x–y coordinate.

$$\begin{Bmatrix} T_{xo} \\ T_{yo} \end{Bmatrix} = \begin{Bmatrix} T_{xi} \\ T_{yi} \end{Bmatrix} + \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\eta & \cos\varphi \end{bmatrix} \begin{Bmatrix} rR_{zi} \\ -r \end{Bmatrix} \qquad (2a)$$

$$= \begin{Bmatrix} T_{xi} + (r\cos\varphi)R_{zi} + r\sin\varphi \\ T_{yi} + (r\sin\varphi)R_{zi} - r\cos\varphi \end{Bmatrix} \qquad (2b)$$

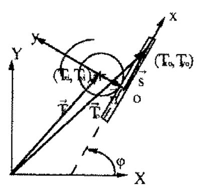


Fig. 11. A vector representation of a rack-and-pinion configuration with respect to the global reference frame.

This formulation, which is given only for the x–y plane with pinion-input, is generalized for all cases (rack-input-and-pinion-output, pinion-input-rack-output, x–y plane, y–z plane, or z–x plane) and expressed as a single parametric matrix in GRF as:

$$\begin{Bmatrix} R_{xo} \\ R_{yo} \\ R_{zo} \\ 1 \\ T_{xo} \\ T_{yo} \\ T_{zo} \\ 1 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & r\sin\varphi & r\cos\varphi & 0 & 0 \\ 1 & 0 & 0 & r\cos\varphi & 0 & r\sin\varphi & 0 & 0 & r\sin\varphi(R_{zi}/R_{yzi}) - r\cos\varphi(R_{yi}/R_{yzi}) \\ 1 & 0 & r\sin\varphi & 0 & r\cos\varphi & 0 & 0 & 0 & r\sin\varphi(R_{xi}/R_{xzi}) - r\cos\varphi(R_{zi}/R_{xzi}) \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r\sin\varphi(R_{yi}/R_{xyi}) - r\cos\varphi(R_{xi}/R_{xyi}) \\ & & & & & & & & 1 \end{bmatrix} \begin{Bmatrix} R_{xi} \\ R_{yi} \\ R_{zi} \\ 1 \\ T_{xi} \\ T_{yi} \\ T_{zi} \\ 1 \end{Bmatrix}$$

where

$$R_{yzi} = R_{yi} + R_{zi}$$
$$R_{xzi} = R_{xi} + R_{zi}$$
$$R_{xyi} = R_{xi} + R_{yi}$$

6.1 *Simulation Example*

To illustrate how individual parameter matrices can be combined to perform qualitative simulation of a mechanical assembly, consider a design task in which the input is a rotational motion having a constant angular velocity (driving shaft) and the desired output has a slow cutting stroke (shaping machine) and a quick return stroke. The motor's speed must be reduced by using a gear train. The design function can be decomposed into

1. Speed reduction (R–R), uniform output rotation.
2. Rapid return motion (R–R), nonuniform output rotation.
3. Reciprocating motion (R–T), nonuniform output translation.

In this example, the three design building blocks are generated by applying modification operators to primitive mechanisms (gear-pair, inverted slider-crank, and a slider-crank). An inverted slider-crank is used to obtain the desired ratio of forward to return stroke time. The details of design concept generation are not given, as the intention here is to illustrate the application of parameter matrices to simulate the assembly. The individual building blocks are shown in Fig. 12a–c. Figure 12d shows the physical assembly of the system. Figure 12e shows the qualitative simulation of the assembly obtained by concatenating individual parametric matrices. It is qualitative in the sense that it ignores the details of the individual elements that make up each building block, and it also simply combines the input and output functions of the various building blocks. The result is an approximate analysis showing the general behavior of the designed artifact. By providing incremental inputs to the first parametric matrix, its output is computed and automatically serves as an input to the second building block. The concatenation of individual parametric matrices is equivalent to the physical assembly of the system. The parametric matrix representation of each building block is given below.

Gear pair 1 and 2:

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & (R_1 + R_2)C\psi \\ 0 & 1 & 0 & (R_1 + R_2)S\psi \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\dfrac{R_1}{R_2} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Gear pair 3 and 4:

$$T_2 = \begin{bmatrix} 1 & 0 & 0 & (R_3 + R_4)C\psi \\ 0 & 1 & 0 & (R_3 + R_4)S\psi \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

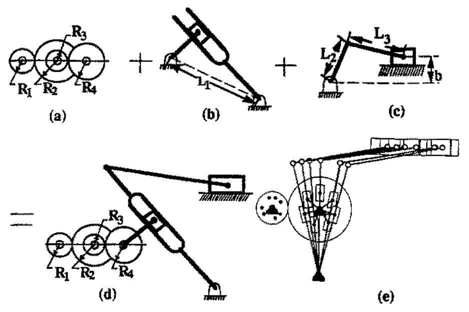


Fig. 12. Building blocks and their assembly. (a) spur gear train derived from the gear-pair building block; (b) inverted slider-crank derived from the slider-crank building block; (c) an offset slider-crank; and (d) assembly and (e) qualitative simulation of the assembly generated by concatenation of parametric matrices of individual building blocks.

$$R_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\dfrac{R_3}{R_4} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Inverted slider crank:

$$T_3 = \begin{bmatrix} 1 & 0 & 0 & L_1C\psi \\ 0 & 1 & 0 & L_1S\psi \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \tan\left(\dfrac{L_2S\theta_{1Z} - L_1S\psi}{L_2C\theta_{1Z} - L_1C\psi}\right) \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Slider crank:

$$T_4 = \begin{bmatrix} 1 & 0 & 0 & (L_2C\theta_{1Z} + \sqrt{L_3^2 - (b - L_2S\theta_{1Z})^2})C\psi - bS \\ 0 & 1 & 0 & (L_2C\theta_{1Z} + \sqrt{L_3^2 - (b - L_2S\theta_{1Z})^2})S\psi - bC \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$R = [0]_{4\times4}$; (zero transformation from rotation to rotation)

The output motion matrix can be described as

$$[T_4][T_3][T_2][T_1] = [T_{\text{out}}]_4$$

where

$$[T_{\text{out}}]_4 = \begin{bmatrix} 1 & 0 & 0 & S_1 \\ 1 & 0 & 0 & S_2 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$S_1 = \{[L_2C\theta_{4z} + \sqrt{L_3^2 - (b - L_2S\theta_{4z})^2}]C\psi_4 - bS\psi_4\} + L_1C\psi_3 + (R_3 + R_4)C\psi_2 + (R_1 + R_2)$$

$$S_2 = \{[L_2C\theta_{4z} + \sqrt{L_3^2 - (b - L_2S\theta_{4z})^2}]S\psi_4 - bC\psi_4\} + L_1S\psi_3 + (R_3 + R_4)S\psi_2 + (R_1 + R_2)$$

and

$$[R_4][R_3][R_2][R_1] = [R_{\text{out}}]_4$$

$$[R_{\text{out}}]_4 = [0]_{4\times 4}$$

7 Conclusions

The methodology based on matrix representation provides a computable theory for type synthesis of mechanisms. The computational synthesis process involves manipulation of matrix elements (0s and 1s). The matrix method is used as a vehicle to represent design building blocks at different levels of abstraction. At the highest level of abstraction, each building block is represented by a string of matrices (function matrix and operational constraint matrices). A function matrix represents the nature of motions that can be accomplished with the mechanism building block that it represents. Operational constraint matrices reveal inherent limitations of the building block interms of its capability to provide reversibility, input–output interchangeability, etc. In principle, additional constraint matrices, beyond operational constraints, can be appended to enable a comprehensive representation of each building block. The matrix representation scheme allows computational synthesis of building blocks by providing a formal means to decompose a given task into an ordered sequence of subtasks (subfunctions) and to match each subfunction matrix with one or more building block function matrices. The representation scheme also permits us to generate alternate solutions depending on the mode of decomposition. Simple rules of matrix algebra provide mathematical operators for task decomposition.

At lower levels of abstraction, symbols in the function matrices are replaced by actual design parameters. By concatenating these matrices, we can perform qualitative simulation of the synthesized mechanism. This allows the designer to verify the input–output relationships between each of the constituent building blocks as well as the overall motion. The simulation is qualitative in the sense that it hides all the details of individual elements of each building block and shows only the input–output relationships. The qualitative simulation capability enables the designer to perform quick simulations of various design concepts.

The key concept is that we can synthesize complex and novel devices by reasoning with functions and operating constraints of motion building blocks. A complete set of building blocks are being identified by analyzing hundreds of ingenious mechanisms from the literature [1, 4]. A library of motion building blocks and their matrix representations is being developed.

Acknowledgments. The author wishes to thank the National Science Foundation, Design Engineering Division and the Dynamic Systems and Control Program, for sponsoring this research (grant DDM-9103008).

References

1. Artobolevski, I.I., *Mechanisms in Modern Engineering Design,* v. 1–3, MIR Publishers, Moscow, 1986
2. Brown, D.C. and Chandrasekaran, B., "An Approach to Expert Systems for Mechanical Design," *Proceedings of the IEEE Trends and Applications Conference,* 1983, pp. 173–180
3. Chase, M.A., "Using DRAM and ADAMS Programs to Simulate Machinery, Vehicles," *Agricultural Engineering,* November 1978
4. Chironis, N., *Mechanisms, Linkages and Mechanical Controls,* McGraw-Hill, 1978
5. Datseris, P. and Palm, W., "Principles on the Development of Mechanical Hands Which Can Manipulate Objects by Means of Active Control," ASME Paper 84-DET-37, ASME, 1984
6. Datseris, P. and Lee, H.T., "Development of All Non-isomorphic Graphs for Mechanism Design with Emphasis on Robot Gripper Design," *Proceedings of the Tenth OSU Applied Mechanisms Conference,* New Orleans, December 1987
7. Denavit, J. and Hartenberg, R.S., "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," *Transactions of the ASME,* 1955, pp. 215–221
8. Dixon, J.R., "Artificial Intelligence and Design: A Mechanical Engineering View," *Proceedings of the AAAI Fifth National Conference on Artificial Intelligence,* v. 2, American Association of Artificial Intelligence, 1986, pp. 872–877
9. Erdman, A.G. and Riley, D.R., "Computer Aided Linkage Design Using LINCAGES Package," Paper 81-DET-121, ASME, 1981
10. Erdman, A.G., "Forty Years of Modern Kinematics," *Proceedings of the Conference on The First Forty Years of Modern Kinematics—A Tribute to Ferdinand Freudenstein,* John Wiley & Son, 1991, Ch. 3
11. Faltings, B., "A Theory of Qualitative Kinematics in Mechanisms," Technical Report UIUCDCS-R-86-1274, University of Illinois at Urbana-Champaign, 1987
12. Forbus, K., Nielsen, P. and Faltings, B., "Qualitative Kinematics: A Framework," *Proceedings of IJCAI-87,* Morgan Kaufman Publishers, Milan, 1987
13. Freudenstein, F. and Dobrjansky, L. "On a Theory for the Type Synthesis of Mechanisms," *Proceedings of the Eleventh International Conference of Applied Mechanics,* 1964, pp. 420–428

14. Freudenstein, F. and Maki, E.R., "The Creation of Mechanisms According to Kinematic Structure and Function," Research Publication GMR-3073, General Motors Research Labs., 1980
15. Freudenstein, F. and Maki, E.R., "Development of an optimum variable-stroke internal-combustion engine mechanism from the viewpoint of kinematic structure," *Journal of Mechanisms, Transmissions and Automation in Design,* v. 105, 1983, pp. 259–266
16. Hain, K., *Applied Kinematics,* McGraw-Hill, 1961
17. Hoeltzel, D.A., Chieng, W.-H., and Zissimides J., "Knowledge Representation and Planning Control in an Expert System for the Creative Design of Mechanisms," *AI EDAM,* v. 1, n. 2, 1987, pp. 119–137
18. Hoeltzel, D.A. and Chieng, W.-H., "Knowledge-Based Approaches for Creative Synthesis of Mechanisms," *Computer-Aided Design,* v. 22, n. 1, 1990, pp. 420–428
19. Hubka, V. and Andreasen, M.M., *WDK 10, Proceedings of the International Conference on Engineering Design,* Heurista, Copenhagen, 1983
20. Jones, F.D., Horton, H.L. and Newell, J.A. (eds.), *Ingenious Mechanisms for Designers and Inventors,* New York: Industrial Press, 1986
21. Joskowicz, L., "Simplification and Abstraction of Kinematic Behaviors," *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* Detroit, 1989, pp. 1337–1342
22. Kannapan, S.M. and Marshek, K.M., "An Algebraic and Predicate Logic Approach to Representation and Reasoning in Machine Design," *Mechanisms and Machine Theory,* v. 25, n. 3, 1990, pp. 335–353
23. Kota, S., Erdman, A.G. and Riley, D.R., "Development of knowledge-base for designing linkage-type dwell mechanisms: Part 1-Theory," ASME Transactions 86-DET-47, ASME, 1986
24. Kota, S., Erdman, A.G. and Riley, D.R., "Development of knowledge-base for designing linkage-type dwell mechanisms: Part 2—Application," ASME Transactions 86-DET-48, ASME, 1986
25. Kota, S., Erdman, A.G., Riley, D.R., Esterline, A. and Slagle, J.R., "A Network-based Expert System for Intelligent Design of Mechanisms," *AI EDAM,* v. 2, n. 1, 1988, pp. 17–32
26. Kota, S. and Lee, C.-L., "A Functional Framework for Hydraulic Systems Design Using Abstraction/Decomposition Hierarchies," *ASME International Computers in Engineering Conference,* American Society of Mechanical Engineers, Boston, August 1990
27. Kramer, S. and Premkumar, P., "A Generalized Expert System Shell for Implementing Mechanical Design Applications: Review, Introduction and Fundamental Concepts," *Proceedings of the 1988 ASME Design Automation Conference,* American Society of Mechanical Engineers, 1988
28. Mittal, S. and Frayman, F., "Towards a Generic Model of Configuration Tasks," *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* v. 2, Detroit, August 1989, pp. 1395–1402
29. Mostow, J., "Toward Better Models of the Design Process," *AI Magazine,* Spring 1985, pp. 44–57
30. Olson, D.G., Erdman, A.G. and Riley, D.R., "A Systematic Procedure for Type Synthesis of Mechanisms with Literature Review," *Mechanisms and Machine Theory,* v. 20, 1985, pp. 285–295
31. Pahl, G. and Beitz, W., *Engineering Design,* The Design Council, Springer-Verlag, London, 1984
32. Kennedy, A.B.W., ed., *The Kinematics of Machinery,* Dover Publications (reprint), New York, 1854
33. Simon, H.A., *The Sciences of the Artificial,* MIT Press, Cambridge, MA, 1969
34. Soni, A.H., Dado, M. and Weng, Y., "An Automated Procedure for Intelligent Mechanism Selection and Dimensional Synthesis," ASME Transactions 86-DET-14, ASME, 1986
35. Thompson, T.R., Riley, D.R. and Erdman, A.G., "An Expert System Approach to Type Synthesis of Mechanisms," *Proceedings of the ASME Computers in Engineering Conference,* American Society of Mechanical Engineers, Boston, MA, November 1985, pp. 71–76