SoftLink: A Matlab/Simulink Based Code for the Analysis, Synthesis, Optimization and Simulation of Mechanisms

Ahmad Smaili, Firas Zeineddine

Mechanical Engineering Department
American University of Beirut
Beirut, Lebanon

Abstract

This paper presents version-I of *SoftLink*, a software package based on Simulink and Matlab for the synthesis and analysis of linkage mechanisms. The purpose of the package is to provide teachers/students with a user-friendly, easy to use tool to facilitate teaching/learning of analysis and synthesis of mechanisms using the familiar programming platform provided by Matlab/Simulink which is used to solve a myriad of other engineering problems. SoftLink is organized and structured to facilitate future additions and enhancements. The current version of SoftLink provides for the analysis, synthesis, optimization, and simulation of 4R mechanisms. Precision point synthesis methods and optimimum synthesis techniques are coded to yield a mechanism for a specific task. The code is designed such that kinematic and dynamic analysis and synthesis of other linkage mechanisms, gear drives, and cam-follower mechanisms may be easily included in future versions. The basic structure of SoftLink is described in detail and two application examples on 4R synthesis are presented to demonstrate its usefulness.

Introduction

Mechanisms are mechanical devices that are used extensively in a myriad of applications which include home-tools, toys, automobiles, and machines. They are essential to the development and operation of almost any machine. Every mechanical engineering program includes a course designed to introduce students to various types of mechanisms. The wide range of topics to be covered and the limited time available pose a challenge to provide students with necessary modern tools to analyze and synthesize mechanisms. As such a simple code using a familiar programming environment would enhance the course offerings and allow for more coverage of mechanism design.

Linkage designer can put to use one of several powerful commercially available kinematic packages. Most notable of these packages are LINKAGES 2000[1] and SyMec[2]. These packages are designed to yield solutions for precision position synthesis problems of 4R mechanisms for motion, path, and function generation tasks. Computer aided mechanism packages like ADAMS[3], Pro/E[4], AutoCAD[5], and IDEAS[6] have also been used for mechanism design tasks.

SoftLink is being developed for educational use. The software will evolve to eventually include kinematic and dynamic analysis and control of linkage mechanisms, gear drives, cam-follower mechanisms, and other mechanisms. Currently it integrates kinematics synthesis of 4R mechanisms for precision position and optimum synthesis techniques. SoftLink provides synthesis for three- and four-precision position motion, function, and path synthesis using dyad equations in complex number forms. The four-position synthesis employs Burmester theory[7]. Optimum synthesis is required when more than four precision points are desired. The code implements Simulated Annealing (SA)[8,9,10] and Weighted Least-square techniques for optimization synthesis[10]. Once a mechanism is synthesized for a specified task, the code allows one to analyze the synthesized linkage to assess its kinematics performance. This is an important step because mathematical solutions do not guarantee that the synthesized mechanism can traverse the positions in the desired order or it does not suffer from branching defects. The simulation option in SoftLink allows the designer to check fine details in the kinematics before moving to the kinetics design and analysis.

Matlab/Simulink is the platform of choice for several reasons: (1) It is a code familiar to students and faculty as it is already available and used in many courses in a curriculum to solve a wide range of engineering problems including control, communications, DSP, optimization, etc.; (2) It has a vast library of functions dedicated to numerical computation; (3) It has an extensive graphics library; (4) It has Graphics User Interface capability; (5) It has the proper hooks for communication with external devices; (6) an extensive set of tool boxes covering a wide spectrum of engineering problems such as optimization and control; (7) the code can be compiled to create an executable file that could run independent of Matlab.

The ultimate goal of SoftLink is to develop an educational mechanical design code which can be used to perform kinematic and dynamic analysis and synthesis of linkages, gear drives, cam-follower systems, Geneva mechanism, screw-drives, and other mechanisms without the need to pay for expensive software. The impetus here is to facilitate teaching/learning of mechanical systems design using the already available Matalb/Simulink simulation, graphics, control, and optimization capabilities.

In this paper the basic structure of SoftLink is introduced and its menus and uses are explained. Two examples are provided to show the capabilities and possible limitations of the code. The first example is a four precision point synthesis problem solved using Burmester theory and SA. The second example utilizes SA to synthesize a mechanism for 8-position path generation task with user defined constraints.

Optimum Synthesis using Simulated Annealing

The theory behind precision and optimum synthesis techniques is well documented in the literature and will not be addressed here[7,8,9]. However, a brief description on features that have been devised and found to enhance the performance of SA is presented.

1. The objective function built in the code is developed to integrate Grashof criteria and to force the crank to rotate in one direction. This objective function may be easily modified by the user as needed.

2. The user can define the limits of any variable (angles, link dimensions, etc.) or choose to hold any set of variable to specific values or to be within a range of values while requesting the code to optimize for the other variables. A vector called *Modify* is added to allow the user

to select the parameters to optimize for while the rest are provided as initial guesses. An entry of 0 to the *Modify* vector indicates that the corresponding variable is fixed as the value provided in the input window and will not be varied in the optimization process. A nonzero value supplied to the *Modify* vector represents the upper limit of the corresponding variable whose lower limit is the value provided in the input window. While this feature adds computation time, it facilitates the use of the same objective function for many possibilities.

3. The code is designed to change the number of iterations as the number of variables change.

4. Geometric cooling schedule $T_{i+1} = c \times T_i$ is used. The default value for c is 0.9, but it may be changed by the user.

5. The probabilistic range of variables change as the temperature cools down from the very hot initial values. This is accomplished by allowing the range for parameters to be selected in regions that include low probabilities but mainly high probabilities; otherwise too much iteration may be performed without minimal acceptance rate. The range selection is made according to the following function

$$\Delta \mathbf{r} = \left[\frac{T_{\text{current}}}{T_{\text{hot}}} \right]^{0.25}$$

SoftLink Structure

SoftLink is designed to facilitate modifications and additions that will be implemented in future versions. This explains the clear separation between three different components of the code: the user interface, the solver modules, and the simulator. The user interface is the set of menus, buttons, popup windows, shortcut keys and other input/output functions that interact with the user. The solver modules are either precision point solvers or optimization solvers. The simulation option is made a separate entity to leave room for future enhancements to optimize graphics refresh rates, choice of link shapes, choices of colors and surrounding plot, and other features.

From operational standpoint, the software is divided into three levels. The first is Input/Output (IO), the second is the Data Transfer and Manipulation (DTM), and the third is the Solver level (Sol). Each level is briefly discussed below. Figure 1 shows a block diagram of the structure and operation of SoftLink dedicated to 4R linkage design. Figure 2 shows a 4R mechanism with the main parameters used in the code.

Input/Output (I/O)

The IO is very critical and very important for proper operation of SoftLink. The role of the IO level is twofold: (1) to wait for a command, and once a command is issued, the IO acquires relevant data, checks the validity of such data, and then provides that data to the DTM level, and (2) to properly and clearly display the output results.

It is relatively simple to wait and load data from the user, but a more involved part is checking the validity of the input. This is done by first checking the solver option. For each solver option a set of constraints and conditions are intelligently posed on the input. This allows the input modules to modify obvious errors or even accept different inputs for the

same set of options. In case of ambiguity or erroneous inputs, the input modules prompt the user and returns to the input window for reentry of data.

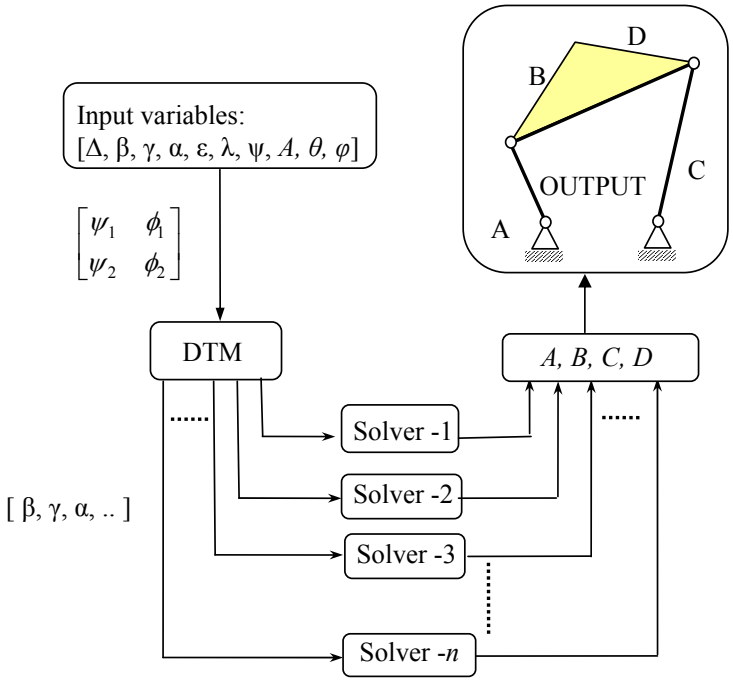

Figure 1: SoftLink structure

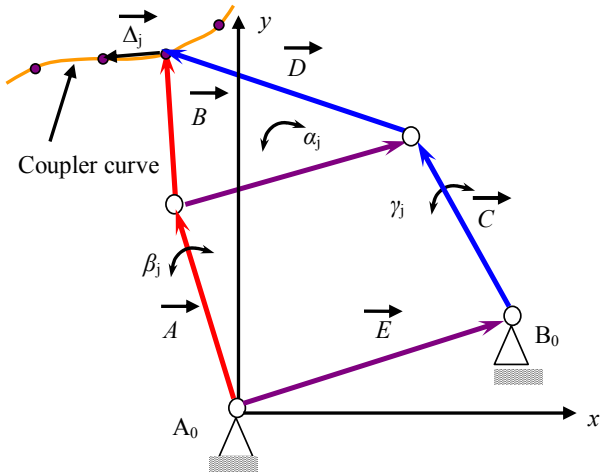

Figure 2 A 4R linkage showing some parameters referred to in the text

The output is much more organized than the input in terms of format. No matter what problem is being solved there is a set of output variables that are common to all solvers. Outputs that are task specific, like Burmester curves, are dealt with separately so as not to effect other operations that might follow. Further details and figures of IO are shown in the examples to follow, where IO is a part of a broader picture.

Data Transfer and Manipulation (DTM)

Even though this stage is mundane in some cases, it could be crucial and necessary in other cases for smooth operation of the software. Sometimes the input provided by the user is directly passed to the solver. But in many cases the solver requires data to be re-partitioned into matrices or vectors differently. Some solvers require different number of inputs. The DTM fills the missing inputs. The DTM mission could be briefly summarized in two tasks: (1) reformatting input data and (2) adding missing data or options that are not transparent to the user. DTM is not a separate module in the code, but rather bits and pieces of code that are distributed inside functions and subroutines. A good way to distinguish DTM codes is to consider the solver as a function that could be used independent of SoftLink, which is true for all solvers, and then considering the input from the user which is different from the input of the function.

Solvers (Sol)

Solvers are functions where pertinent algorithms are coded. They are direct realizations of solutions available for design of mechanisms. Solvers are either MATLAB functions or SIMULINK modules. Whenever a new feature is to be added to SoftLink, the first step is to design its solver. The solver is then tested and used as a function without the user interface shell. If the results are verified a DTM code is customized for the solver and a new input form associated with it is constructed. If the solver is tested again and the results did not match desired results, the problem is more easily determined and limited to the pertinent IO and DTM parts. One common user mistake is inconsistency of angle formats (deg or radians).

This coding scheme makes the solvers robust and independent, which helps in building a toolbox for 4R linkage design with minor additional effort.

Operating SoftLink

To start the software simply type 'SoftLink' at the MATLAB prompt. Figure 3 shows the default design that is loaded initially. Currently there are three user interface menus in SoftLink: File, Solver and Quit. There are also shortcut keys - S to simulate mechanism, D to input a new design, and H for Help. More will be added in future versions.

The File menu allows the user to save the mechanism task parameters and the link dimensions for later use. This is done by using the *Save* command which saves a file in a '*.mech'* format. *Open* loads a saved file and *Save as* makes a new file with a different name. Figure 4 shows the File menu.

The Solver menu allows the user to choose the method used to synthesize the problem. The Solver menu is shown in Figure 5. The 3-precision-point synthesis requests the user to provide the coupler point displacements, Δ_j, the crank angular displacements, β_j, and the follower angular displacements, γ_j, and finally the angular rotations of the coupler link, α_j as the mechanism moves from the first position to the *j*-th position. For three-position synthesis, these increments are formed as 1×2 Matlab vectors. The same variables are required for 4-precision-point and Least Squares Optimum syntheses except that the number of increments is 3 for 4-precision-point and more than 3 for optimum synthesis. See Figure 6 for and example of the input window. The synthesis task results in the first position of the mechanism.

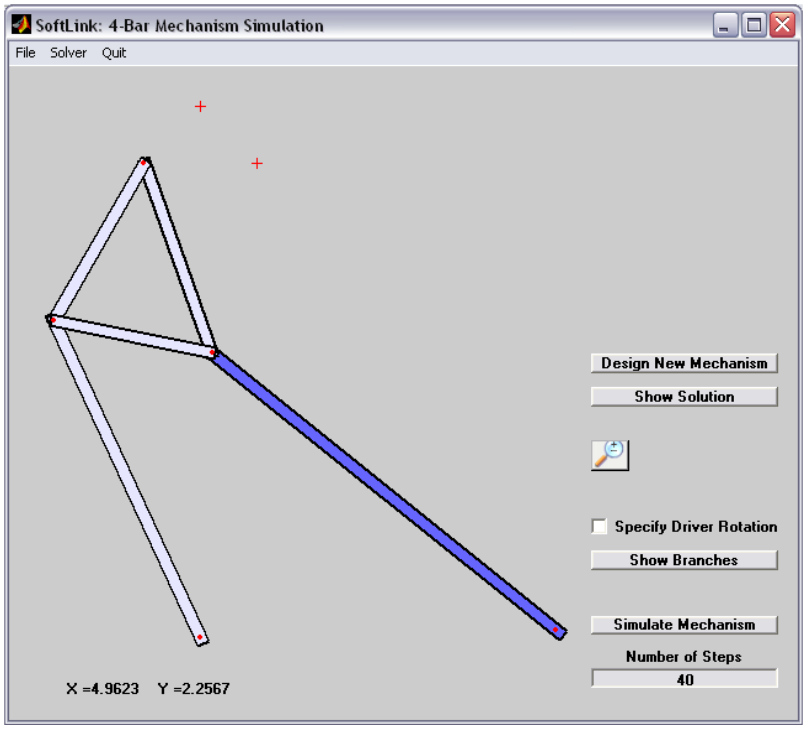

Figure 3: Initial screen

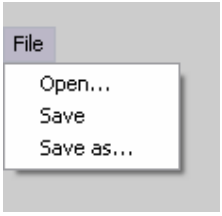

Figure 4: File menu

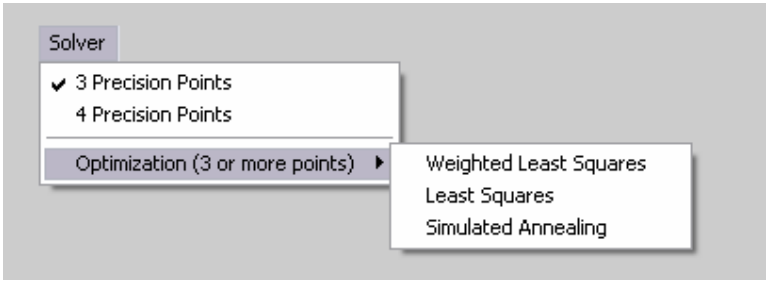

Figure 5: Input window

Figure 6: Typical input window (vectors shown correspond to 3-point synthesis)

For Weighted Least Squares optimum synthesis only an additional weights vector is required. Figure 7 shows a typical weight window. The use of Simulated Annealing (SA) is explained in the example below.

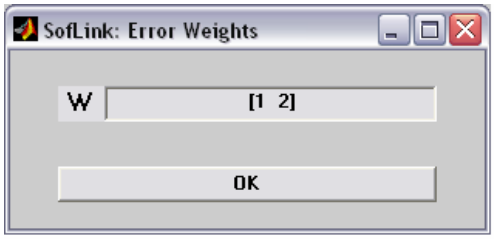

Figure 7: Weight window

Application Examples

In this section, SoftLink will be demonstrated by way of two examples. One applies four-precision position synthesis using Burmester theory and SA and the other is an 8-position synthesis using SA. The task for the first example is to synthesize a motion generator 4R mechanism to traverse a straight line trajectory through four prescribed positions using Burmester theory and SA. The task parameters are given in Table 1.

Table 1 Design positions and solution for example 1

Position	Δ_j	α_j	Solution
1	-	-	A = 0.9507 + 2.9438i
2	1 + 0 i	17	B = 0.2387 - 2.0863i
3	2 + 0 i	33	C = -3.3277 + 3.0285i
4	3 + 0 i	52	D = -1.8595 - 1.3819i

Figure 8 shows the two sets of Burmester pair curves generated for the 4 position synthesis problem. The user can choose the set of solution points from either pair. The color of associated Burmester points changes to help the user choose the corresponding points. Once chosen, the code will assemble the mechanism and the user can simulate it to assess its

performance and check for possible branching and order defects. The user may go back and select different Burmester points until a satisfactory linkage is found. In this example, the Burmester point pairs chosen corresponds to $\beta'_2 = -8°$ for the left dyad and to $\gamma_2 = -9°$ for the right dyad. The final solution for this example is shown in Figure 9.

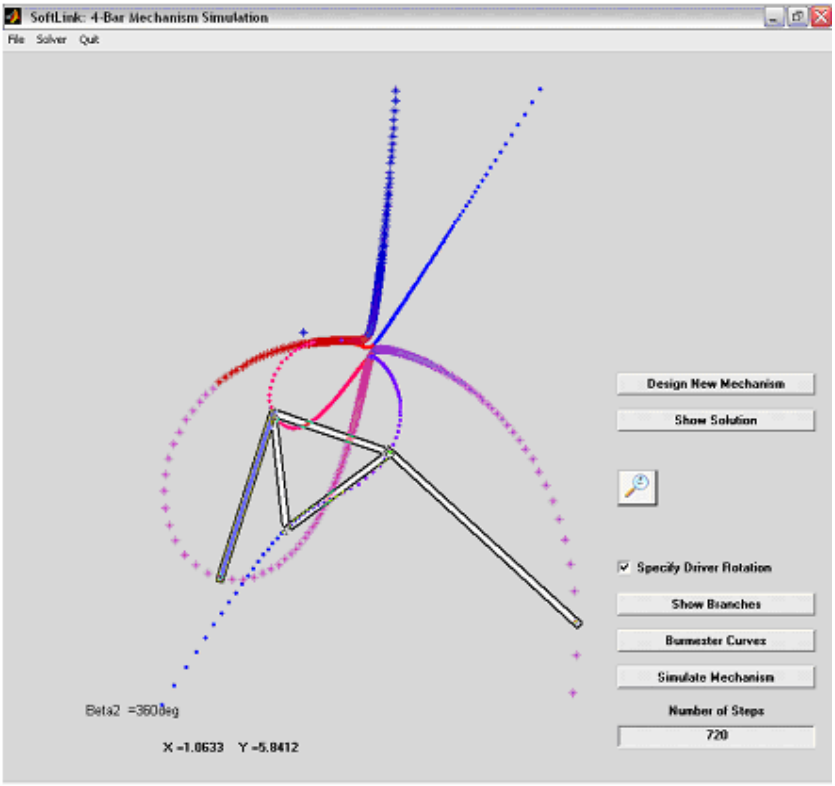

Figure 8: Burmester curves and the mechanism chosen

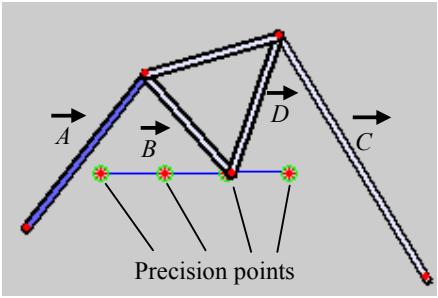

Figure 9: The synthesized mechanism

The initial input window for SA is similar to that shown in Figure 10. The vector size is one less than the number of points desired. The same window is prompted with any optimization option except that the parameters to be optimized are set to the minimum allowable values.

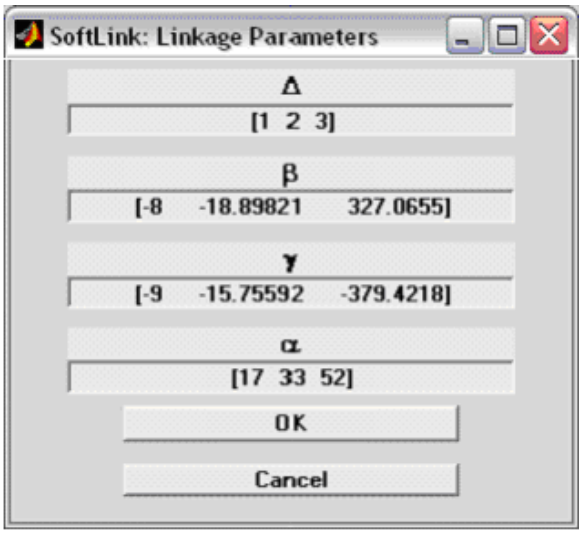

Figure 10: Input box for SA

Once the task variables are provided, the program requests the user to provide the *Modify* vector which includes variables' range to be included in the search. A value of zero means that the variable will not be optimized; else the desired range provided in a separate window. Since the link dimensions do not appear for other optimization approaches, special windows request the minimum value and the range of link dimensions. Those windows are not shown here to save space. The SA synthesized mechanism is so close to that obtained from Burmester solution (Figure 9) and will not be shown here.

The task for the second example is to use SA to synthesize a path generator for eight prescribed positions given in Table 2 and shown in Figure 11. The closed coupler curve is the one generated by the synthesized mechanism shown at the first desired position.

Table 2 Parameters and resulting solution for example 2

Position	Δ	α_j	β_j	γ_j
1	-	-	-	-
2	1 + 0 i	-8.9988	21.8695	-10.6481
3	2 + 0 i	-19.1190	34.9146	-27.8532
4	3 + 0 i	-27.8974	47.2920	-35.7177
5	4 + 0 i	-35.5514	61.3130	-38.8225
6	5 + 0 i	-42.0683	78.6143	-37.0967
7	6 + 0 i	-48.3691	96.7430	-34.1182
8	3 + 3 i	-5.0482	256.9581	32.0151
Solution: **A** = -1.4382 - 0.6277i, **B** = -2.7928 + 3.7471i **C** = 2.7709 - 0.2738i, **D** = -4.9925 + 6.6286i				

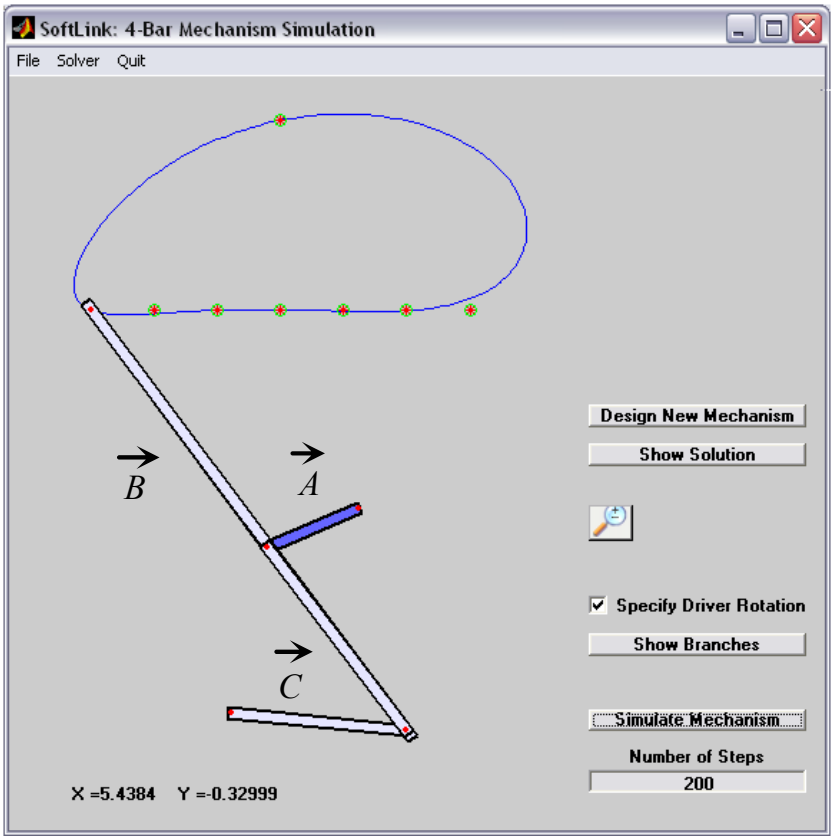

Figure 11: The 8-prescribed positions and the Synthesized Mechanism with the actual coupler curve it generates

Summary

This paper presented version-I of SoftLink, an educational code based on Matlab/Simulink for the analysis and synthesis of mechanical systems. The current version provides for the synthesis of 4R linkages using Burmester precision point synthesis and optimum synthesis using simulated annealing and weighted least square techniques. Future versions will include other optimization techniques such as genetic algorithms (GA), Tabu search, neural networks, and possibly hybrid techniques. The structure of the code is explained and examples are provided. The code is designed such that future additions are easily implemented. The ultimate goal of SoftLink is to develop an educational package for the analysis, synthesis and control of mechanical systems including linkages, cam-followers, gear drives, screw drives, tendon drives, and Geneva wheels.

Bibliography

1. N. Yu, A. Erdman, and B. Byers, "LINCAGES 2000: Latest Development and Case Study," *Proc. of ASME DETC 2002*, Paper # DETC2002/MECH-34375.

2. J. B. Cook and D. G. Olson, "The Design of a 10-bar Linkage for Four Functions Using SyMech," *Proc. of ASME DETC 2002*, Paper # DETC2002/MECH-34369.

3. Mechanical Dynamics, http://www.adams.com/.

4. Parametric Technology Corporation, http://www.ptc.com/.

5. AutoDesk Inc., http://www.autodesk.com/.

6. Structural Dynamics Research Corporation, http://www.sdrc.com/.

7. G. Sandor and A. Erdman, *Design of Mechanism: Vol. II,* Prentice Hall, 1984.

8. D. T. Pham and D. Karaboga, *Intelligent Optimization Techniques*, Springer, 2000.

9. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipe*, *the Art of Scientific Computing*, Cambridge University Press, 1989.

10. Ulah, I. and Kota S., "Globally-optimal Synthesis of Mechanisms for Path Generation Using Simulated Annealing and Powell's Method," *Proc. of ASME DETC 1996*, Paper # 96-DETC/MECH-1225.

Biography

AHMAD SMAILI had served on the Mechanical Engineering Faculty at Mississippi State University (1987-1991) and Tennessee Technological University (1991-1999). Currently, he is an Associate Professor of Mechanical Engineering at the American University of Beirut. His areas of expertise are mechatronics, vibration control, and product design.

FIRAS ZEINEDDINE is currently a graduate student in the Mechanical Engineering Department at The American University of Beirut. He had co-developed ModelAngelo, a rapid prototyping system for polystyrene. His areas of expertise are computer vision and CAD/CAM.