TECHNICAL PAPER

MS88-711

Optimization and Synthesis for Mechanism Design

authors

PATRICK RYAN TURNER
Engineering Manager of Mechanical Computer Aided Engineering
Schlumberger Technologies
Ann Arbor, Michigan

MICHAEL E. BODNER
Software Specialist
Schlumberger Technologies

abstract

This paper presents a method and a computer program which performs kinematic mechanism synthesis using mathematical optimization techniques. An overview of the system is discussed, as well as its architecture, the underlying kinematic and optimization theory, the graphics and user interface, and limitations and difficulties in applying the method. Also discussed is how a program like this could be used by a design engineer/analyst in industry, and how it would be integrated into a full CAE/CAD/CAM environment.

conference

AUTOFACT '88
October 30–November 2, 1988
Chicago, Illinois

index terms

Design
Software
Optimization
Mathematical Models
Synthesis

SME®

Society of Manufacturing Engineers • One SME Drive • P.O. Box 930
Dearborn, Michigan 48121 • Phone (313) 271-1500

Optimization and Synthesis for Mechanism Design

PATRICK RYAN TURNER, MICHAEL E. BODNER
Schlumberger Technologies

This paper presents a method and a computer program which performs kinematic mechanism synthesis using mathematical optimization techniques. An overview of the system Is discussed, as well as its architecture, the underlying kinematic and optimization theory, the graphics and user interface, and limitations and difficulties in applying the method. Also discussed is how a program like this could be used by a design engineer/analyst in industry, and how it would be integrated into a full CAE/CAD/CAM environment.

1 INTRODUCTION

The Mechanism Synthesis Package (MSP) is a software system used to assist in the design of the mechanical motion of planar and spatial mechanisms composed of rigid-body parts. To define how such a system fits into the manufacturing/design community it is useful to examine how a design engineer or mechanical analyst thinks when designing a mechanism.

When a design engineer wants to use a linkage he typically starts by looking around to see what has been done before in similar applications. This is desirable because the design engineer really doesn't want to get too involved in linkage theory if this can be avoided. Or, even more likely, the engineer may have only a limited understanding of linkage theory. In most cases, a starting mechanism is selected based on functional attributes that are close to those desired. Then, the design is refined to conform to the needs of the new application. The process of solving a specific design problem from an existing design will be defined here as "production design".

However, these starting linkages came from somewhere! Therefore, you will find (in large companies, research departments, or smaller but more specialized companies) people who are experts in the application of linkage theory to certain industry disciplines. When trying to solve a new class of design problem in this industry specific setting, the engineer would like to quickly generate many possible design solutions. This will be defined here as "conceptual design".

Whether using “production design” or “conceptual design” methods, the engineer will often use industry specific jargon to describe design tactics and functions. Therefore, care must be taken so no assumed terminology is unnecessarily included in the mechanism synthesis system interface.

With the advent of general dynamic analysis systems (e.g. ADAMS[1]) even the non-specialized engineer can successfully design some linkages from scratch. This is because general dynamic analysis does not necessarily involve much knowledge of linkage theory. General dynamic analysis promotes “design through understanding” which contrasts with methods where designs are done through tradition and experience. Design through understanding advocates using software analysis tools to build a body of knowledge about the problem and seek solutions (possibly analytical) via the resultant understanding. In light of the above comments, MSP was designed with assumptions made about linkage synthesis systems because of the way linkages are typically designed in industry.

The two situations described above are categorized as “production design” and “conceptual design”. Each offers an opportunity for an engineer to use a synthesis tool. The initial MSP system focuses on “production design” because of the applicability in more industry settings.

The rest of the paper will cover system integration, a system description, a scenario for use of the system, system architecture, a description of the technical problems, and an example.

2 DISCUSSION

Linkage synthesis systems have been commercially available for two decades but have not received wide use in industry. Most systems currently available are limited to 2D four bar linkages because they are based on a closed form approach to synthesis using “Burmester theory” rather than “optimal synthesis”. The systems using the Burmester approach require knowledge of linkage theory and can be hard to apply in a practical industry setting. Conversely, because these systems use closed form solutions they execute very fast. This speed advantage does not seem to outweigh the difficulty in their use.

2.1 Mechanism Synthesis and CAD/CAE Integration

Today engineers are greatly aided by software prototyping of their design concepts. It is possible to build software models of the designs; analyze them on the computer; modify those designs on the computer; and remodel and re-analyze until a design meets its specifications. Communication between design and analysis disciplines can be as simple or complex as needed. In the final analysis industry will judge what is efficient and productive.

When CAE techniques were introduced, they revolutionized the design cycle by providing turnaround times that were order of magnitude faster than previous manual methods. Today we take for granted that a mechanism can he designed in a solids modeler; analyzed for motion in a mechanisms analysis package; and its parts analyzed for stress, deflections and natural frequencies in finite element modeling and analysis packages.

[1] (Automatic Dynamic Analysis of Mechanical *Systems)* is a 3D mechanisms analysis package produced by MDI (Mechanical Dynamics Inc., Ann Arbor, MI.)

However, state-of-the-art today is not merely having solids, mechanisms and finite element analysis capabilities. State-of-the-art is comprehensive integration, and MSP is integrated into a system providing geometric integration between solids modeling, mechanisms analysis and finite element modeling; mass property integration between solids and mechanisms; force and inertial loading integration between mechanism analysis results and finite element modeling, and assembly integration between mechanisms and solids permitting visualization and interference checking.

This is all done using a fundamentally sound approach of a single data base. While this integration is state-of-the-art in computer-aided engineering today, the next revolution of the design cycle is already in the making. This system has computerized design, modeling, and analysis, and communication *between* those functions. ‘What remains is to further harness the computer for the engineer. We must work on the *modify* portion of the design cycle, using the computer to generate and test alternatives until design criteria are met. This is the revolution in the making, linkage synthesis and optimization.

2.2 Description of the Mechanism Synthesis System

MSP provides the ability to perform kinematic “dimensional synthesis” of mechanisms. This means that the system will change the dimensions of the mechanism to satisfy some design objective. The system will not change the topology of the mechanism (a six bar Watt type mechanism will remain a six bar Watt type, etc.). The “dimensional synthesis” system for mechanisms has three major parts:

- A graphic display and user interface allows the user to describe the design problem. This is done by first reading the starting mechanism from a file. Next, the user defines the “path objective” through which a user-selected “tracer point” (on the mechanism) must pass. The user then defines “design variables” by selecting links that have a variable length, joints of the mechanism that can move in relation to each other, etc. These design variables are modified by the optimizer until the objective is achieved as closely as possible.

- An optimization package is used to determine how the design variables must change in order for the tracer point to come closer to the “objective”, i.e. the desired path defined in the first step. The optimization calculation requires an analysis to determine gradients so the design variables can be changed in the best way. These changes are made to the mechanism (design variables) and the analysis is repeated to determine if the objective has been met.

- A proprietary kinematic analysis package is provided as an integral part of the system. This analysis program is used to solve for the positions of the mechanism parts as the path is traced. This is necessary to provide feedback to the user and to generate information for the optimization.

The initial mechanism itself may be defined using any means available to create an input file in the ADAMS format. This can range from a text editor to commercially available graphic mechanism modelers. Because a user of MSP will likely be the type of engineer that would use some form of dynamic analysis, they would be very inclined to provide a starting mechanism, and typically would have existing designs that they would like to read in. In addition, the system could also read in the mechanism from an input file from any appropriate mechanism analysis code, like a DADS[2] file, if needed in the future.

The user specifies design variables and objectives through a FIT[3] (Flexible Interface Tool) graphic based interface. The interface represents the mechanism on a graphics screen and the user has full 3D view control. The user interacts with the system by “picking” the mechanism display graphics, making selections from menus, etc. FIT provides the strong advantages of programmability and customizing capabilities for the user. The user may alter the interface to conform to standard terminology for a specific industry, or, change the interface using the provided command language to make the interface more specific to a given design situation or user preference. The command language allows the user to write extensive macros or procedures to automate a specific command sequence or specify parametric operations.

Optimization is performed using a set of commercially available general purpose optimization algorithms and specially developed proprietary optimization algorithms.

2.3 Tactics for Use of the Mechanism Synthesis Package

The user starts by reading in an ADAMS file containing the mechanism data. Once the mechanism is read in the user is allowed to make some choices that will affect the speed of the system. In particular, if the mechanism is indeed planar, the user can choose to run the problem as a 2D problem. This triggers the system to convert the mechanism into a 2D representation and subsequently use a more efficient 2D kinematic solver during the synthesis. This has many implications.

The more simple the synthesis system is (by using 2D solvers when possible) the more robustly (2D iterative solvers and close form solutions converge better than 3D iterative solvers) and quickly it can manipulate the mechanism. Therefore, as mentioned earlier, the user can do conceptual design in 2D, which is the natural tendency, and take advantage of performance improvements allowed by the nature of the 2D design problem. Also, when inventing a new design, the designer would tend to visualize in 2D because paths are easier to visualize in three view projection.

However, if the user has a 3D mechanism to start with, the flexibility of MSP will allow this type of design to be manipulated as well. The added complexity of the 3D mechanism would mean the user would have to work in the realm of production design, implying that a more numerically complex 3D kinematic solver would be used (refer to later sections for the technical details). This does not add any complexity to the use of the synthesis system, but, does limit the magnitude of modifications that the system can apply to the starting design. This implies the user should start with a design that generates an initial path that is relatively close to the desired path.

MSP provides the engineer with the ability to use a portion of the trace curve which can avoid problems such as small transmission angles and locking configurations. The use of a portion of the trace curve will cause improvements in system performance.

[2] DADS (Dynamic Analysis and Design System) is a product of Cadsi (Computer Aided Design Software Inc., Oakdale, IA.)
[3] FIT (Flexible Interface Tool) is a product of Schlumberger Technologies, CAD/CAM Division, Ann Arbor, Michigan.

Motion generation is sometimes needed along with path generation. Motion generation requires that not only does the mechanism have to trace through a path, but, the orientation of the tracer point on the mechanism is important as it traces the path. The system will allow this type of design objective.

In defining the tracer path the user will want to think in terms that are convenient to the specific design problem. The user may want to define the path in terms of global coordinates, coordinates that define the part carrying the tracer point (called the “Local Part Reference Frame”), or by offsetting the original position of the tracer point. And, the path must be generated by a single degree of freedom (dof) mechanism where that single dof is prescribed by a motion generation parameter. The user will lock down dof’s that are not under consideration in the current design situation for starting designs that are multi-dof.

The use of familiar terms is important and must be given consideration. In addition to customization provided by the Flexible Interface Tool (FIT) these considerations are present in the default interface. When defining objectives, design variables, etc., the user should be able to relate to the mechanism in terms that are convenient. This is done by attempting to keep the interface from having linkage theory specific jargon. Also, the user can use terms that are specific to the starting design that has been input to the system. Specifically, the user will pick graphic representations of the starting design to define objectives, design variables, and constraints needed by the synthesis system. This will promote user-friendliness and ease of use.

In conceptual design the quick generation of many ideas is important to efficiently move to a subset of candidate solutions. However, in production design the decisions are limited because the “starting design” defines the candidate solution. So, the engineer is more willing to wait for the solution to a very specific design question related to this “starting design”. Therefore, the system can crunch away at a complex problem and return with an optimized answer to a specific question In the past the user would simplify the design into terms that could be handled easily, rather than working with the complex problem. Like converting a complex 3D problem into a series of simpler 2D problems. This is common in all engineering analysis situations.

Working in 3D with MSP may require more user interaction. For example, in a complex design the synthesis system may stop short of meeting the objective. This is true for two reasons:
an exact answer may not exist for the given mechanism, or, a local optimum may be found by the optimizer. In these cases the user frequently is willing to use the points describing the *desired* path to influence the path that the system will calculate. This is done knowing that the complexity of the solution is not allowing convergence to the exact points. This example is intended to stress the point that the Mechanism Synthesis Program is a tool to help guide the engineer down the path to an optimal solution.

2.4 Architecture of the Mechanism Synthesis System

2.4.1 Synthesis Problem Specification

Design Objective Specification

The design objectives which MSP provides for are path generation and motion generation.[4] These tasks have traditionally been treated separately. With the approach that MSP takes however these objectives can be combined in the same problem.

The user first creates a *tracer point,* which in a linkage is often referred to as a coupler point. This is the point on the mechanism that is to trace a particular path. The user specifies the part the tracer point is on, the tracer point's coordinates in the part's local reference frame, and, if pertinent, the tracer point's orientation.

Next, any number of *target points,* or precision points, are created by entering their global coordinates (and orientation if desired). These are the set of points that define the *desired* path. that the tracer point is to traverse. Target points can be specified to be for path generation only (i.e. the tracer point is simply to pass through the target point), or for motion generation (the tracer point is to pass through the target point *and* its orientation must be aligned with the target point orientation.)

A nuance of using numerical methods is that design objectives will not be reached "exactly". However, since the optimization iterations can continue until a specified tolerance is reached, it is only necessary to specify a small enough tolerance for "practical" purposes (e.g.. a piece can only be manufactured to a tolerance of, say, .001").

Satisfying the objective function for mechanism synthesis requires minimizing the sum of individual terms of two types, traditionally referred to as path generation and motion generation. Each target point contributes a term to the objective function.

Path Generation:

In path generation we want a tracer point (coupler point) to pass through a set of target points (precision points). That is, we want the distance between each target point and the trace curve to be zero at some time during the motion of the mechanism. The approach used by MSP is to calculate the objective function as the square root of the sum of the squares of the distances from the target points to the corresponding closest points on the tracer curve.[5]

$$F_p = \sqrt{\sum_i k_i \cdot (d_i^2)}$$

where,

F_p = path generation objective function term to be minimized

K_i = importance weighting factor (user defined)

d_i distance from target pt i to the closest point on the tracer curve

[4] Motion will be available soon. Function generation may also be incorporated into the system in the future.

[5] An alternate approach would be to consider the objective function as the *maximum* of these distances for all target points.

Motion Generation:

In motion generation we want a tracer point to pass through a target point *and* we want the part that the tracer point is on to be oriented in a prescribed manner when it passes through the target point. Since tracer and target points can each have their own local coordinate axes, the orientation objective can be met by requiring that the tracer point's x-axis be parallel to the target point's x-axis and that the tracer point's y-axis be parallel to the target point's y-axis. This will then ensure that the local z-axes are also aligned. Two vectors will be parallel when their dot product equals one (i.e., when the angle between them is zero). Again, the square root of the sum of the squares can be used.

$$F_m = \sqrt{\sum_i k_i \cdot [(1 - |\vec{A}_i \odot \vec{B}_i|)^2 + (1 - |\vec{C}_i \odot \vec{D}_i|)^2]}$$

where,

F_m = motion generation objective function term to be minimized

k_i = importance weighting factor (user defined)

$\vec{A}_i$ = tracer pt local x-axis vector (at the path position closest to target pt i)

$\vec{B}_i$ = target pt i local x-axis vector

$\vec{C}_i$ = tracer pt local y-axis vector (at the path position closest to target pt i)

$\vec{D}_i$ = target pt i local y-axis vector

Alternate methods could be used for calculating the objective function term for motion generation. The axes' *cross'* product could be minimized, or the difference in the tracer and target point *Euler angles* could be minimized.

Other design objectives may also be included in MSP in the future. *Function generation* could be done by discretizing the function and treating each point as a target point. Depending on the accuracy desired this could produce a large number of target points and a corresponding degradation in system performance.

Prescribed timings for the target points could be incorporated quite easily. This would allow the user to specify that the tracer point pass through each target point at particular times. Terms would be added to the objective function by calculating the difference between the desired time and actual time that the tracer passes closest to the target point.

Design Variable (DV) Specification

The third step (after tracer and target point definition) in a mechanism synthesis problem is to specify what dimensions of the model will be allowed to vary. These dimensions will correspond with the optimization *design variable's,* or DV's. MSP will systematically change the DV values in order to find the best *design.* This results in a set of values for all DV's, which satisfies the objective.

MSP allows the user to graphically model five different types of design variables. Each of these is ultimately correlated to the position of a joint endpoint marker. When the optimization changes the value of a DV, MSP translates (or rotates) the marker by the corresponding change. A change in the position of a marker will affect how the mechanism is assembled and, thus, will change the shape and path of the tracer curve.

The types of DV's that can be modeled are presented in the following table.

Variable Type	Description
variable length link	A part that has two and only two joints can be specified to be a *variable length link,* or simply a *link.* For each link MSP creates one DV whose value is the distance between the two joints. As the optimization changes the value of the DV, MSP automatically translates one of the joint endpoint markers on the part along the line connecting the two joints.
floating ground support	A joint which is connected to ground can be specified to be a *floating ground support.* MSP creates one DV for each co-ordinate (e.g. the global X-coordinate) which is allowed to vary. The DV value is the corresponding global coordinate of the marker. As the optimization changes the value of the DV, MSP automatically translates the joint endpoint marker (on the ground part) along the float direction.
floating joint	*Floating joints* are used on parts that have more than two joints. The user is required to specify which part's marker will float, and in what direction it will be allowed to float in the local part reference frame.
floating tracer point	The location of a tracer point on its part is normally fixed. However, the user can specify the tracer point to be a *floating tracer point,* in which case its local x, y, and/or z coordinates can be made variable. An optimization DV will be created for each coordinate allowed to float.
floating target point	Target points, which are normally stationary, may be specified to be *floating target points.* A floating target point may float in any of the global X, Y, or Z directions. An optimization DV will be created for each coordinate the user allows to float. Floating target points would be used in problems which do not place a restriction on all of the coordinates of a target point.

2.4.2 Optimization Problem Formulation

Formulating the optimization problem consists of converting the user's specification of the design objective and design variables into a form suitable for applying mathematical programming techniques. This formulation is done entirely by the software "behind the scenes".

The optimization DV array is built up in the following fashion. For each variable length link modeled by the user MSP creates an optimization DV with an initial value equal to the length of the link. For floating supports MSP creates an optimization DV for each direction (X, Y, or Z) that the support has been specified to float in. The value of the DV is set equal to the corresponding global coordinate of the joints marker on the ground part. For example, assume we have a support (a joint connected to ground) at global coordinates (10,8,0), and the user has specified that this support can float in the X and the Y directions. During formulation one DV is created with an initial value of 10., and a second DV with an initial value of 8. Floating joints, floating tracer points, and floating target points are treated similar to floating supports. The DV values for floating tracer points, however, would be in the *local* part coordinate system.

2.4.3 Constraint Specification

MSP, as currently implemented, sets up an *unconstrained* optimization problem. Unconstrained problems are easier and faster to solve than constrained problems. In the future MSP may be extended to consider various user modeled constraints, such as requiring certain points to remain in a design *envelope* during operation, or keeping transmission angles within specified limits, and so on. If constraints such as these existed, additional terms would be included in the objective function.

2.4.4 Analysis Via Kinematic Solver

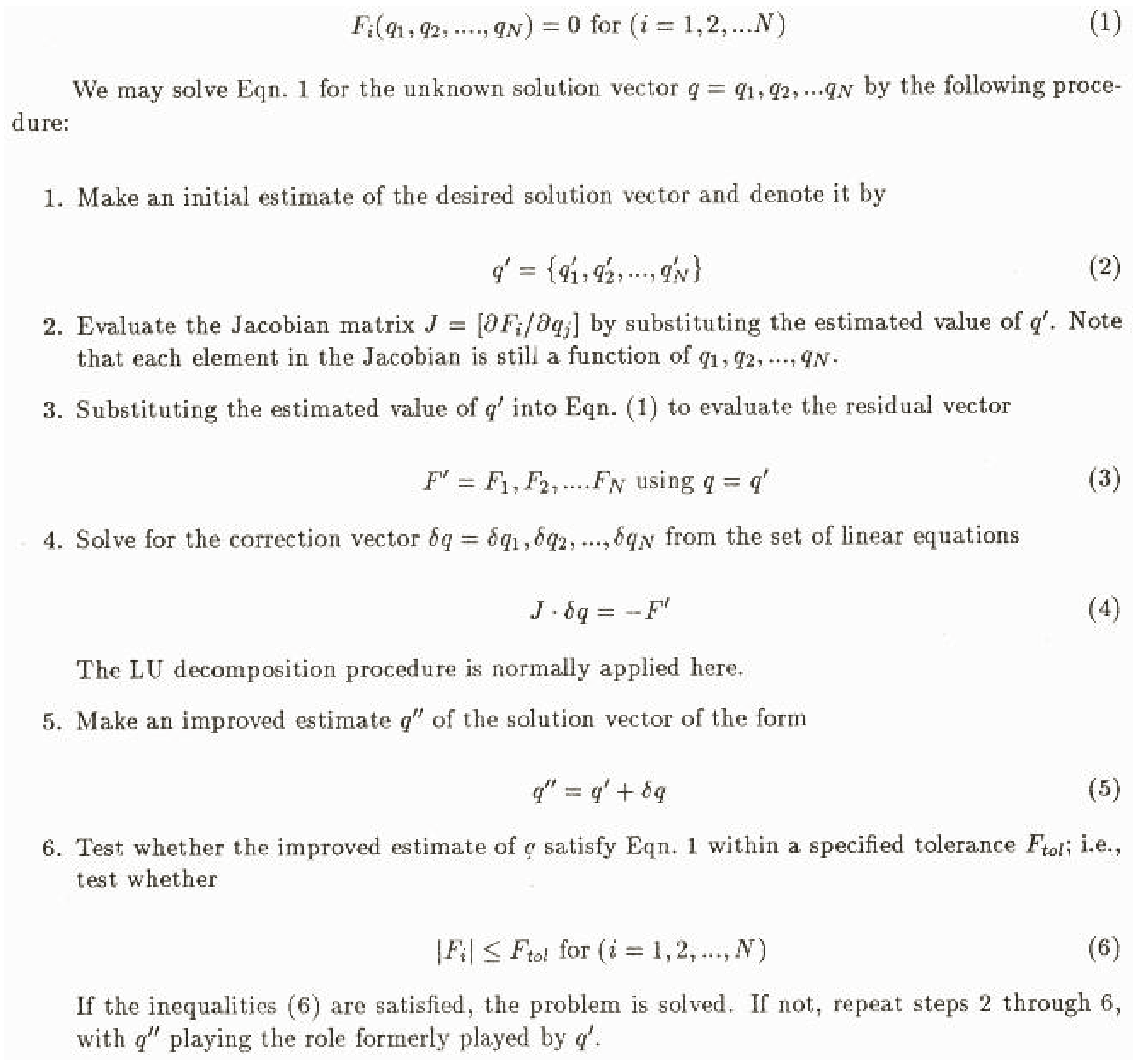
The MSP system contains general purpose 2D and 3D kinematics position solvers. It accepts as input information specifying the mechanism. The mechanism entities currently recognized by MSP are parts, markers, joints, and generators. Parts have markers on them to identify points of interest, like joint connection locations and tracer point locations.

The mechanism must be a single degree-of-freedom (dof) system with a generator to prescribe the motion input. Forcing the system to have zero dof’s qualifies the mechanism as kinematic. If the system has dof’s that are not prescribed by a generator, then the system is dynamic. Analysis of a dynamic system requires mass properties, an integration type solver, etc. Therefore, the user must ensure the mechanism is kinematic for MSP to operate properly.

The input to the kinematic solver contains part locations, the location of markers on these parts, the number and type of joints connecting part markers, the location and type of generator(s) imparting motion to the mechanism, and identifies which part is the ground part. Generator values are specified (typically only one generator exists) and initial guesses given for part locations and orientations. The output of the kinematic solver are the actual part positions.

The Kinematic Solver (KS) is basically a general numerical procedure which solves a system of non-linear algebraic equations using Newton-Raphson method. This iterative technique can be found in many books about numerical methods The essential features of the method will be reviewed here.

Given a set of non-linear algebraic equations expressed in the form of:



$$F_i(q_1, q_2, ..., q_N) = 0 \text{ for } (i = 1, 2, ...N) \tag{1}$$

We may solve Eqn. 1 for the unknown solution vector $q = q_1, q_2, ...q_N$ by the following procedure:

1. Make an initial estimate of the desired solution vector and denote it by

$$q' = \{q'_1, q'_2, ..., q'_N\} \tag{2}$$

2. Evaluate the Jacobian matrix $J = [\partial F_i / \partial q_j]$ by substituting the estimated value of q'. Note that each element in the Jacobian is still a function of $q_1, q_2, ..., q_N$.
3. Substituting the estimated value of q' into Eqn. (1) to evaluate the residual vector

$$F' = F_1, F_2, ...F_N \text{ using } q = q' \tag{3}$$

4. Solve for the correction vector $\delta q = \delta q_1, \delta q_2, ..., \delta q_N$ from the set of linear equations

$$J \cdot \delta q = -F' \tag{4}$$

The LU decomposition procedure is normally applied here.

5. Make an improved estimate q'' of the solution vector of the form

$$q'' = q' + \delta q \tag{5}$$

6. Test whether the improved estimate of q satisfy Eqn. 1 within a specified tolerance F_{tol}; i.e., test whether

$$|F_i| \leq F_{tol} \text{ for } (i = 1, 2, ..., N) \tag{6}$$

If the inequalities (6) are satisfied, the problem is solved. If not, repeat steps 2 through 6, with q'' playing the role formerly played by q'.

The Newton-Raphson method is used to solve a system where the total number of unknowns is equal to the total number of equations. Therefore, Kinematic Solver can only be applied to a zero-degree-of-freedom mechanisms. In other words, the total number of part coordinates (*x, y, z, , ψ, θ,* and *φ* for 3D and (x, y, *θ)* for 2D should be equal to the total number of constraints.

Constraint Equations

A mechanism is simply a “constrained multi-body system”. Constraints are applied between any two parts in a mechanism. In kinematics, there are various types of constraint groups, i.e., joint, ground, generator, joint primitive, coupler, user-defined constraints,..., etc. Each group actually consists of several basic constraint elements. For example, if two parts are connected by a spherical joint, the *x, y,* and *z* coordinates of the two joint markers should be coincident with each other. Therefore, a spherical joint actually introduces three constraint equations to the system (or removes three degrees of freedom from the system).

The constraint value is the objective to which a constraint equation is supposed to reach. For joints and joint primitives the objective is zero, for ground the objectives are the $x, y, z, \psi, \theta,$ and φ of the ground part respectively; and for generators, they are equal to specified generator values.

Jacobian

A Jacobian function is the partial derivative of a constraint equation with respect to a specified coordinate. For example, if the first constraint equation, F_1, in Eqn. (1) is a rotational generator applied between, say, part 1 and 4, then the element at the first row on the first column of the Jacobian matrix is the value of the partial derivative of equation F_1 with respect to the x-coordinate of part 1.

All the Jacobian functions in the Kinematic Solver were derived explicitly.

2.4.5 Optimization Procedure

The underlying optimization theory behind MSP uses non-linear mathematical programming for solving the optimization problem. Mathematical programming is an iterative process. An initial "design" (i.e., the set of values for all DV's) is modified in such a way so as to decrease the objective function. This new design is then modified to determine an even better design, and so on. Each determination of a new design is usually called an optimization step. Each optimization step can be considered (with some simplification) to be composed of two tasks, determining a *direction* to step in, and determining how *far* to step in that direction.

Step Direction:

The step direction is more formally called the *gradient.* It is the vector of partial derivatives of the objective function with respect to the DV's. In many optimization applications an explicit equation exists for the objective function and one merely has to use rules of calculus to mathematically calculate the partial derivatives. With the synthesis method presented in this paper there is no such closed form equation, and therefore, *finite differencing* must be used to determine the gradient. To use finite differencing to calculate the gradient, each DV is perturbed (temporarily increasing its value by a small amount, say .01%) and the change in the objective function is calculated.

For instance, assume the value of the objective function is 20.0 for the current design. To calculate the gradient we would first increase the value of DV#1 (which is, say, the length of link **#3** with an initial length of 10) by .01%, making its new length 10.001. All other DV values are kept at their current value. We then recalculate the objective function. This is accomplished by performing a kinematic analysis of the mechanism at each target point (as well as calculations to determine the point on the tracer curve closest to the target point). The square root of the sum of the target point distances squared is the objective function (assuming path generation objectives only). For our example this equals 20.14. Therefore, the partial derivative of the objective function with respect to DV#1 is:

$$(20.14 - 20.0)/(10.001 - 10.00) = 140.$$

This will be the first element in the gradient vector. A similar procedure is performed for each of the other DV's, to produce the full gradient vector.

Step Length:

The next task is to determine the step length, that is, the factor we will multiply the gradient by which will give us the amount to change each of the design variables. This itself is a sub-optimization, often referred to as "line minimization" since only one variable is involved. We want to take "trial steps" along the gradient direction in order to determine the step size which will give us the lowest objective function value. A small initial trial step size is first taken. If the objective function has decreased we take a longer step; if it has increased we take a shorter step. This process can be thought of as trying to move toward the bottom of a valley in the solution space. A step too long will over-shoot the valley bottom and start going "uphill", while a step to short will not reach the valley bottom. This search is continued until a specified tolerance is reached. The tolerance defines when the objective function is changing slowly enough that the slope is almost zero indicating the bottom of the valley has been reached.

The step size that we now have is a factor which, when multiplied by the gradient vector and added to the current design vector, will give a new design which will come closer towards satisfying the design objective. This new design is then used as the starting design for the next optimization step iteration.

3 TECHNICAL COMPLEXITIES IN OPTIMAL SYNTHESIS OF MECHANISMS

Several issues have been explored relating to numerical simulation and optimization. A summary of findings are in this section. Techniques for improving system performance and reliability are also outlined.

Nested optimization

The mechanism synthesis method outlined here actually consists of several optimization problems nested within one another.

At the highest level is what we would call the main optimization.[6] In this optimization problem, the DV's are the length of links, the position of joints, supports, and tracer point, etc. The objective function is based on the distances from the target points to the tracer curve.[7] These distances, however, are dependent on what point on the tracer curve we use. We want to use the point on the tracer curve which is *closest* to the target point in question. Each target point will have a different *closest tracer point.* It is not obvious which point on the tracer curve is closest to the target point. One way to determine the closest tracer point is to perform an optimization to minimize the distance between the target point and the tracer curve. This is the second level of optimization, which has to be performed separately for each target point.

For this second level of optimization there is only one DV, the generator angle. The objective function is the distance from the target point to the tracer point. Since there is only one DV this problem

[6] Depending on the algorithm used for the solution of the "main" optimization, another level of optimization may exist within the main optimization to perform the line minimization to determine the best step length, as already discussed.

[7] Other *constraints* can be added to the objective function as penalty function terms.

can be considered a form of line minimization and an appropriate optimization method used. The global position of the tracer point, needed in the calculation of the objective function, must be determined in some way. The kinematics solver is used for this purpose, and this forms the third level of optimization.

The third level of optimization exists within the kinematics solver to solve for the position of the mechanism parts, and thus the global position of any points on these parts. Each 3D part will have six DV's: the coordinates (x, y, *z)* of its origin, and its Euler angles *(* ψ, θ, and φ) specifying the orientation of the part local axes. Thus, the total number of DV's for this third level of optimization will be equal to six times the number of parts in the mechanism. The objective function is the maximum of the closure error at any of the joints. That is, we want the joint endpoint markers (which are on separate parts) to be within a very small tolerance. The Newton—Raphson method in matrix form is ideal for this type of problem, and is used here.

3.1 Numerical Analysis and Multiple Solutions

When attempting to optimize a four-bar linkage to pass close to a number target points, the solutions provided by the kinematic solver may not all correspond to the same way of assembling the linkage. This problem is well-known and is encountered in linkage design tools currently available:
the mechanisms they generate may require disassembly and reassembly in the middle of operation which is not practical. The number of different ways to assemble a mechanism grows combinatorially with the number of links; e.g., there maybe two ways to assemble a four-bar, eight ways to assemble a six-bar, etc. In general, there are often multiple solutions to the system of constraint equations being solved by the kinematic solver and some method must be devised to ensure convergence to the correct one if relatively large changes to the mechanism are being performed.

If the initial starting design is sufficiently close to the correct solution, the Newton-Raphson technique will converge to it. However, "sufficiently close" does *not* mean "closer than any other solution". It is now understood that the convergence regions of Newton-Raphson, even for such simple problems as finding the (complex number) solution to $x^4 = 1$, are extremely complicated, with fractal boundaries. Starting points in a significant proportion of the design space result in convergence to solutions other than the closest, and may converge slowly. The use of transcendental functions in system equations seems to foster this problem.

A short-term, partial solution to this problem is to try to ensure that starting designs given to the kinematic solver are always close to the desired solution. This can be achieved by: a) requiring the user to assemble (or nearly assemble) mechanisms before beginning simulation or optimization and b) starting with the answer previously provided by the solver in a closely similar situation (e.g. the optimizer's previous closest approach to the same target point). These steps should considerably reduce the number of assembly errors, but *they are not guaranteed to eliminate them.* Any optimizer may take a step in its search that is just a little too big, or is into the fractal region between solutions.

3.1.1 Inability to assemble a mechanism

The kinematics solver is sometimes given a mechanism which cannot physically be assembled (recognized when the Newton-Raphson algorithm exceeds, say, 20 iterations). This can happen in two situations:

- A generator angle is given which is outside the mechanism's range of motion. This occurs during the sub-optimization algorithm search for the generator angle which yields the mechanism position closest to a particular target point.

- As the optimizer is taking trial steps, the design variables describe a mechanism that cannot be physically assembled.

To avoid this problem a penalty function must be applied to the objective function to force the synthesis system to move back into the feasible region where the mechanism can be assembled.

3.1.2 Speed

By using an iterative kinematic solver for generality of the application the system will be much slower than if a specific closed form solver is written for the design problem at hand. This trade off was made early in the development of MSP to broaden the range of linkages that could be handled. Areas for potentially speed improvements have been identified:

- Avoid performing a line minimization for finding the points on the tracer curve closest to the target point when calculating gradients.

- In special cases, such as planar 4-bar and 6-bar mechanisms, substantial speed-up can be achieved by using special closed-form kinematics solver routines. Therefore, these special solvers will be investigated.

3.1.3 Optimization-related problems

Two issues have been identified with respect to optimization in the synthesis system:

First, when near the optimum the optimizer zig-zags, taking very small steps (and a correspondingly long time to reach the minimum). A related problem also exists concerning finding the true global optimum, as opposed to a local optimum which is not close enough to the desired objective. Using an initial design which is sufficiently close to the global optimum may reduce this problem, however, convergence to an acceptable solution may not be guaranteed in all problems. This is a problem inherent in non-linear mathematical optimization. Tools are provided to the user to interactively take corrective action to "get the system moving again".

Second, in the kinematic solver and the optimization algorithms there are various tolerances, factors, scalings, etc., that affect the success of the optimization. These currently seem difficult to compute on a problem-independent basis. Therefore, the solution to this requires examining many design problems, and trying to find good values for the optimizer's control parameters. If no set of values works for all problems, then a method will found for computing them on a problem-specific basis. Tests are being run to investigate what scheme for modification of these parameters will be successful.

4 EXAMPLE: 6-Bar Stamping Mechanism

We will use a 6-bar stamping mechanism as an example demonstrating a typical session using MSP to perform path generation mechanism synthesis. The user procedure can be broken down into 4 tasks: defining the initial mechanism, specifying the objective, defining design variables, and optimizing.

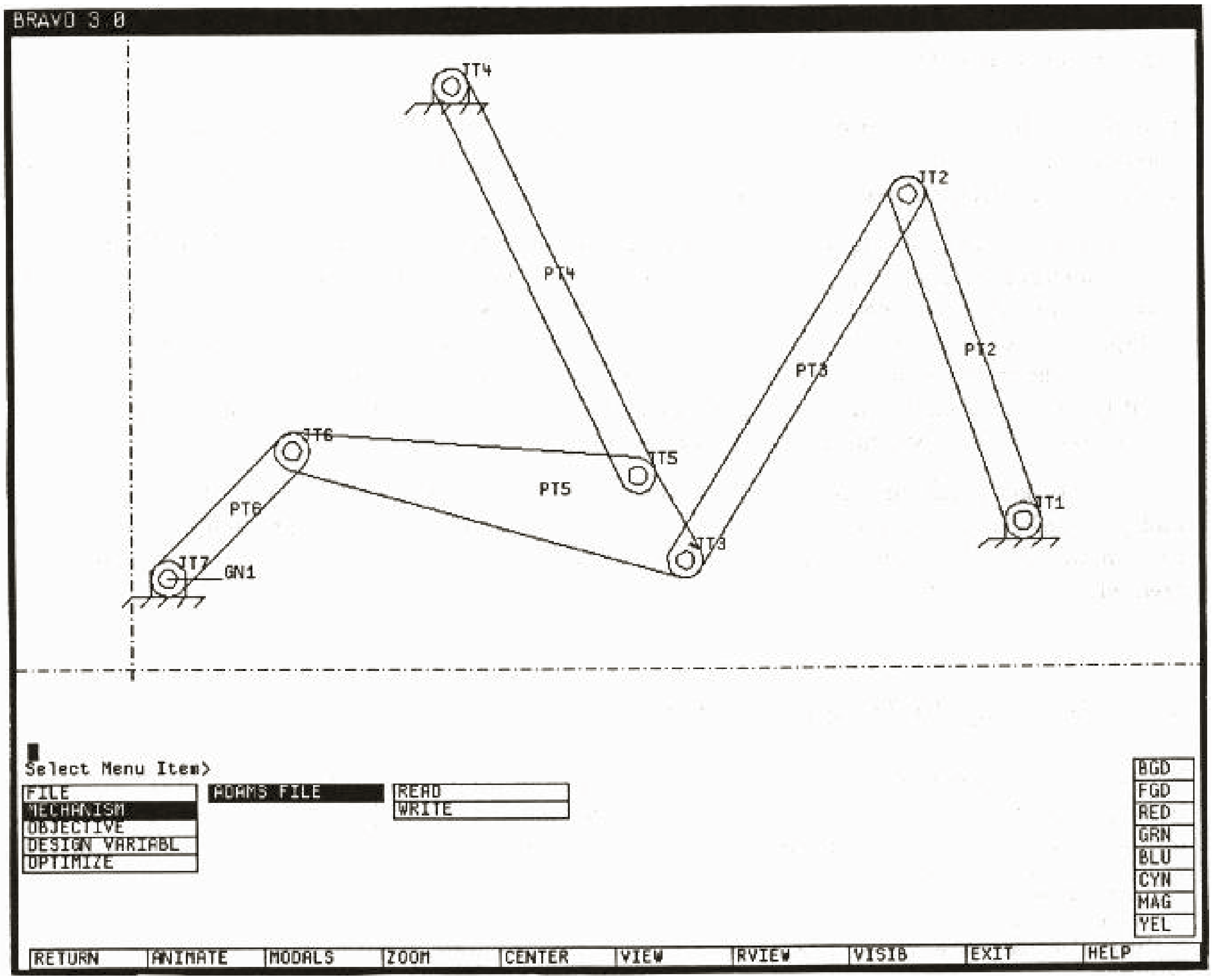


Figure 1: The initial “Stamping Mechanism” and the MSP user interface

4.1 Defining the Initial Mechanism

The first step in mechanism synthesis is to specify the type of mechanism to be used.[8] The user of MSP is expected to draw on his experience to select the type of mechanism which will best suit the task at hand. For this example we have chosen to use a 6-bar mechanism, since it is a relatively simple mechanism and its tracer curve exhibit qualities which make it suitable for a stamping mechanism.

[8] *Type synthesis* may in the future provide the engineer with help on determining the best type of mechanism to use for a particular task.

We must also provide initial design dimensions for the mechanism. Even though many of these dimensions will later be changed by the optimization process, we do want to make our initial mechanism come as close as possible to our objective.

The mechanism data is input to MSP via an ADAMS format data file. This can be created by a text editor or a mechanism modeler. The initial mechanism data must specify a valid kinematic mechanism which can be properly assembled and which operates correctly. Refer to Figure 1 showing the original mechanism and the graphic user interface of the MSP program.

4.2 Specifying the Objective

The design objective in our example is to have a point on the mechanism pass through two target points during its motion. We convey this information to the optimization package by making the proper menu selections and entering the point locations.

The tracer point is created by selecting menu items OBJECTIVE TRACER_PT CREATE, and then either picking the part that it is on or entering the part's ID. The coordinates of the tracer point, in the part's local coordinate system, are then entered[9] by picking the position on the screen or typing the coordinates. When the tracer point is created the graphic display is automatically updated: the part shape is changed to the shape of a triangle; a symbol representing the tracer point (a cross in a circle) appears on the part; the tracer curve is drawn, showing the path of the tracer point as the mechanism goes through its motion. See Figure 2.

The target points are created by selecting menu items OBJECTIVE TARGET_PT CREATE, and picking locations on the screen or entering global coordinates. Each target point is represented as a small cross. The new value of the objective function is also displayed as each target point is created.

4.3 Defining the Design Variables

We can create any combination of the five different design variable types provided in MSP. However, in engineering design the user will want some part dimensions and joint locations to remain fixed due to actual implementation requirements. For instance, maybe the generator position can not be changed, or certain part lengths are standard and thus fixed. In our example we will assume similar restrictions and only specify certain DV's.

[9] If motion generation was being performed Euler angles specifying the tracer points orientation would also be entered.

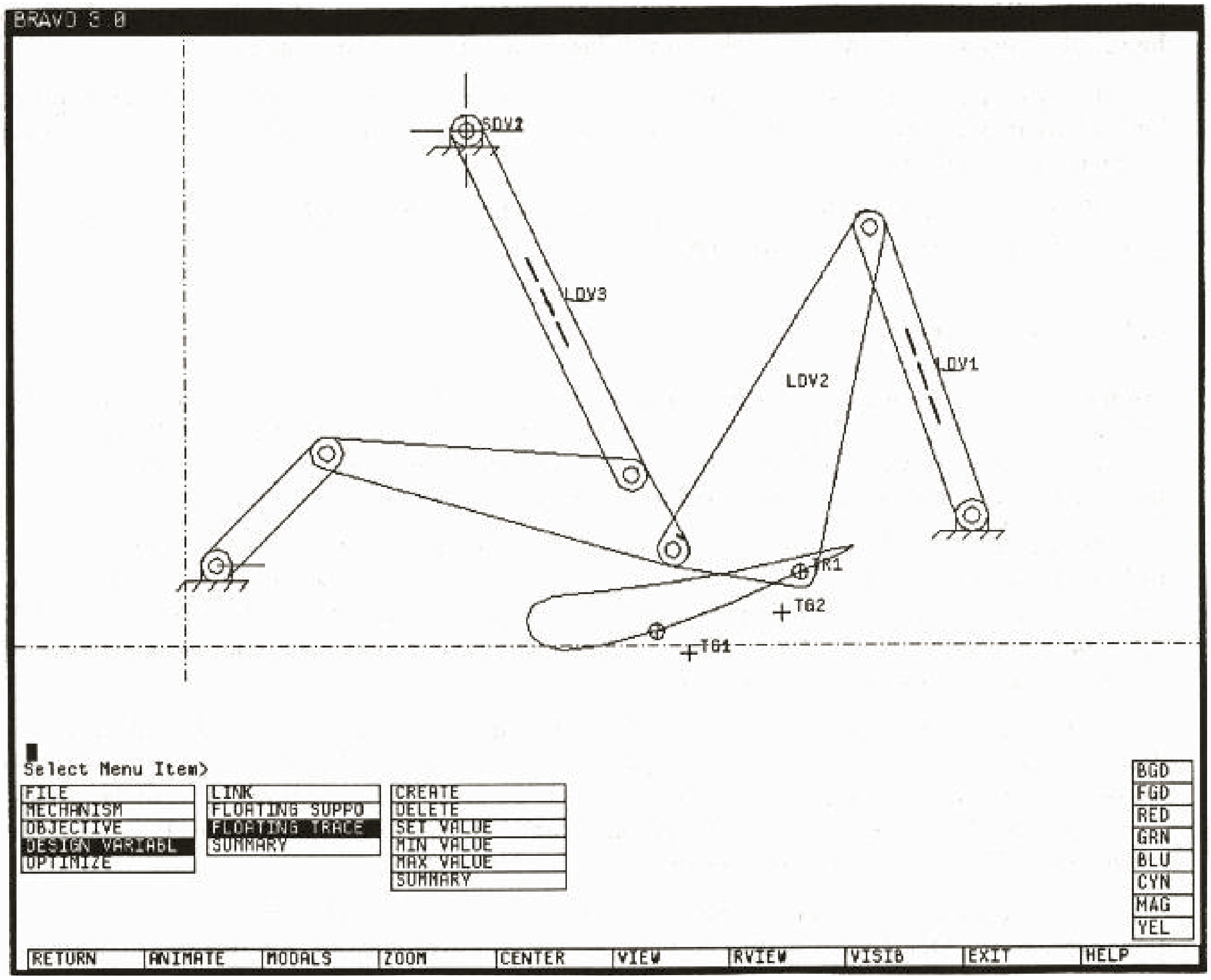


Figure 2: The mechanism with the objective and design variables defined. The objective is the tracer point and two target points. The design variables are denoted by the triple dashed lines.

Referring to Figure 2, lets say that we want the lengths of parts #2, 3 and 4 to be variable. We select menus items DESIGN_VARIABLES LINK CREATE, and pick the parts from the screen. A triple dashed line appears in the middle of the these parts, indicating the a DV exists.

Ground supports which are not necessarily required to remain in their initial position can be made DV's by selecting FLOATING..SUPPORT CREATE. For our example, lets assume that joint~1 can float in the X- and Y-directions, and that joint#4 can only float in the X-direction. Triple dashed lines represent the DV and show the directions that the support can float along.

Next, we can specify that the location of the tracer point on its part does not have to be fixed, assuming other factors in the mechanism design do not preclude this. We select FLOATING_TRACER_PT CREATE, pick the tracer point, and select the local directions that we want to be variable, say x and y. Again, triple dashed lines represent the float directions.

Many design situations will require limits on where supports may be located, or how short or long a part may be. We can accomplish this by specifying maximum and minimum values for each of the design variables.

We now have defined eight design variables: three link DV's, three DV's for floating supports, and two DV's for the floating tracer point.

4.4 Optimizing

We now are ready to begin the optimization process by selecting the OPTIMIZE menu item. The software will perform the formulation of the optimization problem, that is, it will convert the objective and design variable information into mathematical equations and data arrays which can be understood by the optimization package. Formulation takes only a few seconds.

We can now tell the optimization to "take a step", that is, to start the optimization process and determine a new design. Each step can take several seconds to a few minutes depending on the type of mechanism and the number of target points and design variables. The FIND_OPTIMUM menu option is used to have MSP continually take steps until the optimum solution is found, or at least until the optimization can find no *better* design.

After any optimization step we can interactively modify our synthesis model. We can add, delete, or relocate target points. We can add or delete design variables, or explicitly set their values. If the optimization proceeds in a way which we don't like we can reset the design to any previous step. We can set various optimization parameters and tolerances in order to improve the convergence. We can print out summaries and design histories.

The original objective function value for this example problem was 4.0 12. After 20 optimization steps, which took approximately 30 minutes, the objective function was reduced to .050.

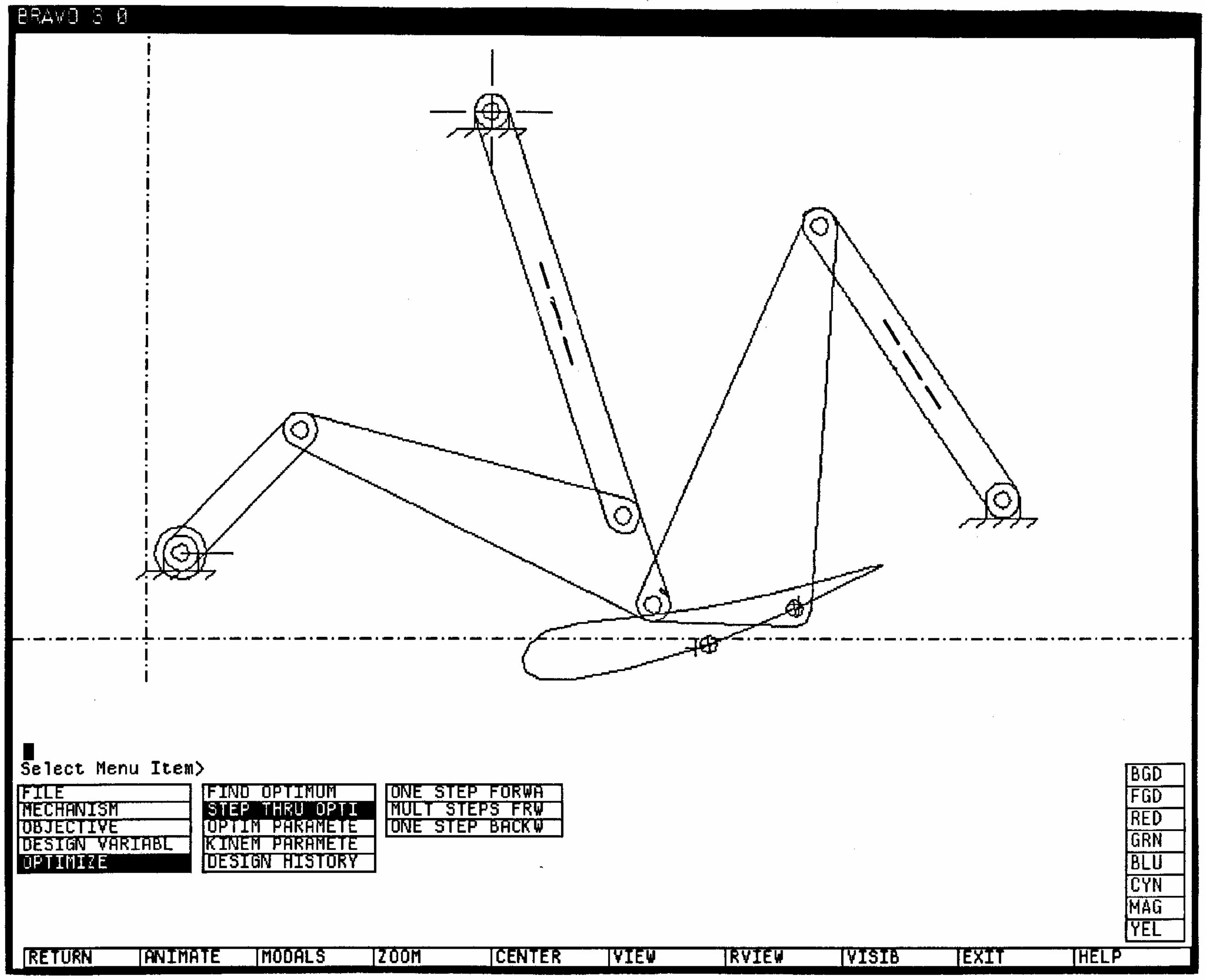


Figure 3: The final mechanism with the objective.

5 CONCLUSION

Applying the principles of optimization and synthesis to mechanism design has been demonstrated. This was done through the discussion of a graphic oriented software program. Discussion of how the system could be used in the design environment was also presented. An overview of the theory and architecture of the system was outlined. A discussion of numerical complexities was also provided with either algorithmic or tactical solutions suggested. The authors feel that the technology presented in this paper constitutes a step forward in design automation. The understanding provided by this project can continue to be built upon to further the integration of design, analysis, and manufacturing.

6 ACKNOWLEDGEMENTS

The authors would like to acknowledge the effort of the following colleagues. Harry Barrow and Glenn Kramer of Schlumberger Palo Alto Research, Stephen Chen, Mike Prebenda, Mike Collingridge, and Jeff Cooper of the CAD/CAM division. Their efforts have contributed to making this paper possible.