MULTI-DOMAIN MULTI-OBJECTIVE OPTIMIZATION OF MECHANISMS: A GENERAL METHOD WITH TWO CASE STUDIES

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Thomas Sullivan

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Master of Science

Dr. James D. Van de Ven

October, 2013

Abstract

While the design of mechanisms is a well-studied field, current optimization techniques generally focus on the kinematics and dynamics and relegate other aspects of the analysis to separate stages of the overall design process, resulting in a loss of optimality when the entire multi-domain system is considered. This thesis presents a general method by which a mechanism optimization problem may be efficiently formulated and solved, considering multiple competing design objectives across multiple analysis domains. Two case studies illustrate the practical application of this general method. The first is the kinematic-structural optimization of a hydraulic rescue spreader ("jaws of life"). The second is the kinematic-dynamic-thermodynamic optimization of a novel six-bar linkage for an internal combustion engine.

A variety of powerful general-purpose multi-objective algorithms are available from the literature. In particular, genetic algorithms are well-suited to multi-objective problems, and the NSGA-II algorithm from this category is employed here. Three strategies are presented to formulate multi-domain mechanism optimizations in a way that can be solved efficiently by a multi-objective genetic algorithm and is free of explicit constraint functions even for complex problems. First, it is shown that the use of non-traditional design variables, such as angles and adaptive interpolations, can result in smaller design spaces to be searched and can guarantee that all optima lie within the selected range of a given design variable. It is also shown that traditional precision-position synthesis techniques can in some cases be employed in a preliminary analysis to reduce the dimension of the design space. Finally, a nested optimization structure is proposed in which kinematic design variables and objectives are optimized in an outer loop, with the non-kinematic problem being optimized in an inner loop at every outer loop iteration, improving the efficiency and stability of the optimization process.

These techniques were applied to the hydraulic rescue spreader problem in order to design a six-bar mechanism that could exert a 10,000 pound force through a pair of jaws over a 24 inch spreading distance while maintaining performance-critical kinematic behavior and remaining light and compact enough to be a handheld tool. The structural stresses in each part of the linkage were modeled, using a combination of analytical methods and finite element analysis. The final optimization result was superior to a similar commercially available model with respect to all four kinematic and structural objectives.

Having successfully optimized a low-speed mechanism with a structural motivation, the method was also applied to a high-speed mechanism with a thermodynamic motivation. A Stephenson-III six-bar linkage was developed in order to optimize the motion of the piston in an internal combustion engine and achieve a cylinder volume as a function of time most conducive to efficient combustion. A number of mechanical objectives relating to balancing and mechanism size were used in order to find a solution capable of practical implementation. A slight increase in thermal efficiency over a purely sinusoidal piston motion was obtained, along with satisfactory values of the mechanical objectives.

Contents

List of Tables

List of Figures

Chapter 1:

Introduction

1.1 Background

The ever-increasing performance of the modern computer has opened vast new realms of possibility to engineering science. The ordinary and partial differential equations governing the physics that are relevant to most engineering problems have been well understood for several centuries in some cases, but it is only in the past few decades that advances in computing power and software design have made the numerical solution of almost any well-posed problem in engineering analysis virtually routine. Even complicated problems over convoluted three-dimensional domains, or with multiple types of co-dependent physical phenomena, are for the most part only a few mouse-clicks from solution with today's powerful commercial software. Perhaps even more significant than the effects of the computer revolution on engineering analysis, profound as they have been, are its effects on engineering optimization. Computerized analysis allows determination of the effects of a given design; computerized optimization, for which analysis is but one of several necessary conditions, allows the determination of the *best possible* design. Optimizing all but the most simplistic cases would be unthinkable without the aid of fast modern computers and effective modern algorithms. The full power of computer-assisted optimization is yet

to be felt, but it has the potential to revolutionize the practice of engineering.

This thesis aims to take a step toward that goal by proposing a general approach to multi-domain, multi-objective problems involving mechanisms. Erdman et. al. [1] define a mechanism as a collection of idealized rigid bodies, or links, connected by various types of joints in such a way as to permit relative motion between them. The purpose of a mechanism is to create a desired pattern of motion, and/or to transmit force and therefore energy from an input to an output. Traditionally, the study of mechanisms has been restricted to kinematics and dynamics. Kinematics is the study of relative motion without regard to the forces that cause it. When only the motion of a mechanism is of interest, and when the links and joints are idealized, a strictly kinematic analysis is possible where pure mathematics is all that is required and physics is no longer involved, perhaps the only branch of mechanical engineering of which this can be said. When the forces associated with the motion in accordance with Newton's second law are of interest, the branch of physics that deals with forces and motion, known as dynamics, becomes necessary. Usually even in a dynamic analysis the links and joints remain idealized.

The theory of the kinematics and dynamics of mechanisms is well developed. Pencil-and-paper mathematics are usually sufficient for the analysis of mechanisms, and for relatively simple synthesis problems powerful and elegant direct techniques are available. More recently, there has been significant effort devoted to bringing the emerging tools of numerical optimization to bear on more complex problems in mechanism design, with substantial success. However, the manner in which the tools of kinematics and dynamics can be used as part of a larger multi-domain optimization has received relatively little attention. The need to expand the conceptualization of the problem beyond the usual methods can arise in one of two ways. First, if the idealization of the links and joints is not a valid simplification for a particular application, traditional techniques lose much of their power. For example, both kinematics and dynamics are silent on the optimal design of link shapes to withstand the stresses induced by external loads and inertial forces. Second, it must be

recognized that the transmission of force, motion, and energy by a mechanism is not done in a vacuum and that there will be complex devices manifesting a variety of physics coupled to the mechanism at the inputs and outputs. In many cases a truly optimal design can only be reached by considering all parts of the system as an integrated whole.

Determining the optimal solution to a complex multi-domain problem is an ambitious goal. It is further complicated by the fact that in the real world it is rare that the quality of a design solution can be measured with a single metric, so that the very definition of optimality becomes uncertain. The field of multi-objective optimization is a relatively young one, but has provided both a re-conceptualization of optimality in the multi-objective case and an ever-expanding collection of algorithmic tools to acheive it. This thesis explores how the integration of traditional mechanism design, multi-domain analysis, and multi-objective optimization can result in a powerful and flexible approach to the solution of a variety of real-world design problems, in large part through two case studies.

The first case study in question is the design of a Watt-II six-bar linkage for a hydraulic rescue spreader. Hydraulic rescue spreaders, also known by the brand name "jaws of life", are hydraulically actuated mechanisms used in emergency situations to remove victims trapped inside wreckage, often as a result of automobile accidents. They are composed of a linkage that converts the motion of a hydraulic cylinder to the spreading action of a pair of jaws. A typical rescue spreader and motion schematic is shown in Figure 1.1. Generally the jaws must exert a large amount of force to deform various metal structures, resulting in large loads throughout the linkage. As the rescue spreader is an emergency tool that must be used with relative speed and ease by a single operator, care must be taken to design the mechanism efficiently to minimize the mass. This is in fact one of the multi-objective optimization goals for this case. The others involve kinematic characteristics of the mechanism: the jaw tips must achieve a large enough spreading distance to be effective without requiring too much rotation of the jaws, which can make it difficult for the tips to grip material effectively. This requirement tends to favor long jaws that can achieve large

spreads with small rotations; however, this trend is structurally less sound and the structural analysis for the mass minimization will drive the design in the opposite direction, as will a third objective, namely minimization of the physical size of the mechanism for ease of use purposes. Finally, variation in the spreading force as the mechanism moves is minimized, as current models often suffer from low spreading force in the closed position compared to the open position.

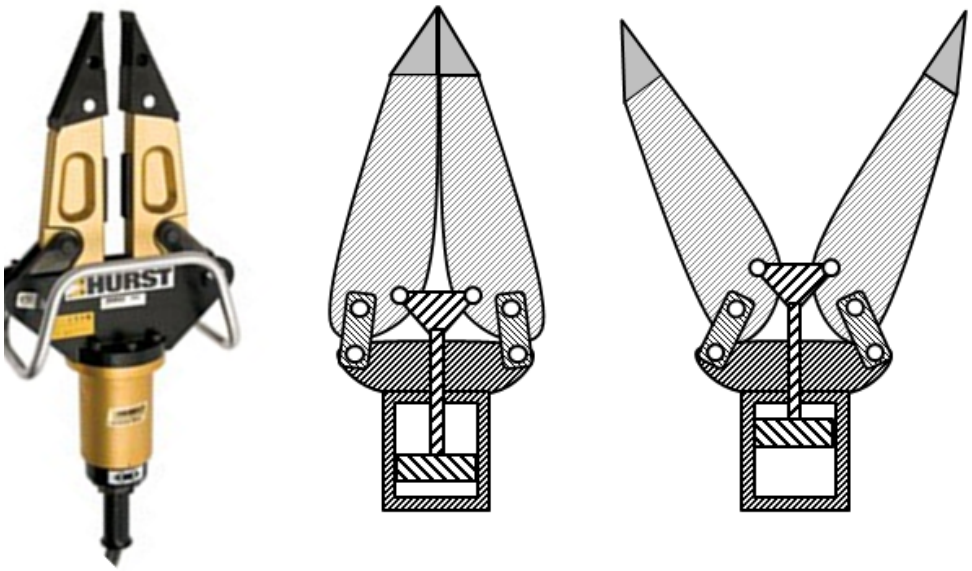


Figure 1.1: A typical rescue spreader and motion schematic [2]

The second case study is a Stephenson-III six-bar linkage to convert the combustion of fuel in an internal combustion (IC) engine into shaft power, as illustrated in Figure 1.2. A number of mechanical objectives are set: minimization of shaking force and moment, side-loading of the piston, and overall mass and footprint of the linkage. As is usually the case in multi-objective optimizations, the objectives tend to drive the design in conflicting directions. For example, there exists an analytic procedure for exactly eliminating the shaking force, which is however not employed as it results in increasing the shaking moment, and simultaneous minimization is instead used. Another example of objective conflict in this problem arises from the fact that mass-radius product terms become important in the

balancing analysis used to minimize shaking force and moment. This leads to an inherent conflict between choosing a low mass and large radius or a high mass and small radius, favoring the mass and footprint minimization objectives respectively. The final objective, and the most novel in the context of the study of mechanisms for IC engines, is improving the efficiency of the combustion process by altering the timing of the cyclical piston motion, which can be quite complex due to the use of a six-bar linkage instead of the usual simple crank-slider. This is a particularly good example of the interdependence of pure kinematics and a complex multi-physics phenomenon in a ubiquitous real-world device.

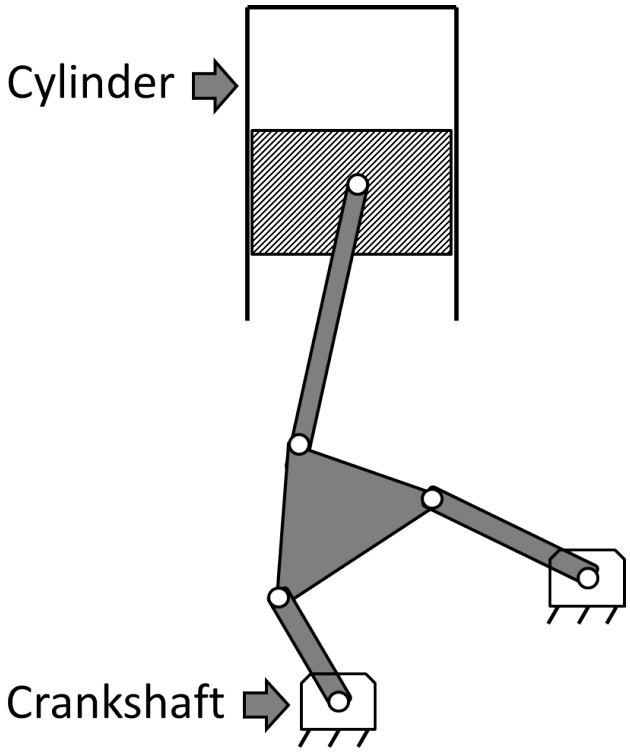


Figure 1.2: Stephenson-III linkage for an internal combustion engine

1.2 Literature Review

This section reviews the state of the literature relevant to this thesis. Broadly speaking, there are three potentially relevant areas, relating to the general optimization approach pre-

sented and to the two specific case studies. The review is therefore divided into three parts. The first part examines what techniques are currently available for the multi-objective, multi-domain optimization of systems in which the kinematics of a mechanism play at least some part. There are two further subdivisions, corresponding to the fact that there are two natural but opposite directions to approach the problem from. One can either attempt to extend specialized kinematic analysis, synthesis, and optimization techniques to more general systems, or take general-purpose multi-objective optimization techniques and attempt to provide refinements to make them for efficient for partially kinematics-specific problems.

The second part deals with some techniques for the optimization of elastic structures needed for the rescue spreader problem. Specifically, different mathematical approaches to the parameterization of free-form shape outlines are examined, as well as the alternatives that exist for efficient analysis of complex structural problems. Finally, the third part summarizes existing studies of six-bar linkages in internal combustion engine applications.

1.2.1 General Multi-Objective Optimization

The literature on general multi-objective optimization is first examined. It should be noted that this thesis does not attempt to contribute to this field, but rather applies existing knowledge in this field to a specific class of problems. Therefore emphasis on this section is on potential usefullness for the optimization of mechanisms. To keep this section concise, a number of concepts are referenced without explanation, so it may prove helpful for a reader who is unfamiliar with optimization in general or multi-objective optimization and Pareto optimality in particular to read Section 2.1, which gives an introduction to these concepts.

The basic challenge involved in solving a multi-objective problem is how to decide what to do when the different objectives drive the design in different directions. This generally entails specifying so-called “preference information”, in which the decision maker makes choices about which objectives to prioritize that are not present in the problem formulation

per se. Given this information, various algorithms can be employed to search for the best solution to the problem subject to the imposed preferences.

There are four possible approaches to the specification of preference information. The first is the use of so-called "no-preference" algorithms. Another option is *a priori* specification, in which the preference information is given before beginning the optimization. The opposite approach is *a posteriori* specification, in which the optimization is conducted without preference information and a decision maker selects from a number of Pareto-optimal results at the end. A middle ground is progressive specification, in which preference information is given during the course of the optimization.

No Preference Specification

This is a bit of a misnomer, as it is impossible to truly avoid preference specification by the very nature of a multi-objective problem. Members of this category instead use very generic and non-arbitrary preference strategies to minimize the loss of generality of solution entailed by the specification of preference information. A common way to do this is by choosing a special value for a parameter in a less generic method. For example, the Min-Max method attempts to minimize the value of the maximum objective value, that is, completely prioritizes the objective that has the worst value at any given time. This is equivalent to the limit of the expression

$$\min\Big[\Big[\sum_{j=1}^{k}\Big(\frac{f_j(x)-f_j^*}{f_j^*}\Big)^p\Big]^{1/p}\Big] \tag{1.1}$$

as p goes to infinity, where f_j is the jth objective function, x is the vector of design variables, f_j^* is the best possible value of f_j, and p is a weighting parameter [3]. For infinite p, the largest (normalized) objective value dominates completely. Choosing finite p corresponds to stronger specification of preference information, in that it determines how much the larger values will dominate the expression.

Another example is the Nash or objective product method, which minimizes

$$\min\Big[\prod_{j=1}^{k}\big[f_j(x)\big]^{w_j}\Big] \tag{1.2}$$

for all $w_j = 1$ [4]. Again, a more arbitrary choice of the parameter would constitute a stronger preference articulation.

These sorts of methods are not suitable for this application since they are too general to give much flexibility to the designer, and only produce a single solution instead of a Pareto-optimal set to choose from.

A Priori Preference Specification

Perhaps the most natural way to approach a multi-objective optimization is to specify the preference information before beginning. There is almost always a certain degree of arbitrariness to this specification; after all, if there were a single non-arbitrary way to combine the different objectives into a single overall metric, one would not be faced with the multi-objective problem in the first place. The specification generally takes the form of choosing values for parameters in a combination equation.

The most basic approach is the weighted sum [4], formulated as

$$\min\Big[\sum_{j=1}^{k}\lambda_j f_j(x)\Big], \qquad \sum_j \lambda_j = 1 \tag{1.3}$$

This approach simply adds the different objective values together with different weights λ_j. Choice of the weights constitutes the preference information. Other approaches that use the same basic idea of a parameterized combination of objectives include equations 1.1 and 1.2 for arbitrary parameter values [4]. It is also possible to take approaches that rank the objectives in order of importance (such as the lexographic approach [5]), which is more general than specifying exact numerical parameter values but of course strongly favors the objectives at the front of the list.

In general, this type of pre-specification approach has the advantages of simplicity and

efficiency but has the major drawback that it will only find a single point on the Pareto front, determined by the choice of parameter values. A common strategy is to repeat the optimization for a range of parameter values, so as to find a variety of points on the Pareto front and approximate it as a whole. However, this technique can become computationally expensive for large numbers of objectives (which requires a high number of parameters in the combination equation) since the number of parameter combinations required to cover all possibile preferences adequately grows rapidly. Additionally, less sophisticated *a priori* methods such as the weighted sum tend to cluster at certain points on the front and ignore others, and/or are unable to find concave portions of the front. Still, the *a priori* approach remains a viable alternative to some of the more sophisticated methods discussed below.

Progressive Preference Specification

Also known as interactive methods, these algorithms entail the active participation of the decision maker in the optimization process. The STEM or STEP method, introduced by Benayoun et. al. [6] in 1971, narrows down the solution space in successive decision steps. The problem is formulated as

$$\min\Big[\sum_{j=1}^{k}\Big(w_j^h\Big(f_j(x)-f_j^*\Big)\Big)^p\Big]^{1/p}, \qquad w_j^h>0, \sum_j w_j^h=1 \tag{1.4}$$

as in equation 1.1, but with the addition of weights w_j^h, with h being the iteration counter. At each iteration, the decision maker evaluates the single solution this formulation produces, and then reformulates the weights and problem constraints to adjust the solution. The other parameter, p, is taken to be something generic such as 1 or infinity and is not crucial to success as the decision maker has direct control over the weights. A more sophisticated but similar approach is taken by Steur and Choo [7].

These strategies can be quickly dismissed for the present application, as they have the important disadvantage of requiring continuous user input. An optimization algorithm that can be programmed, set going, and ignored until it completes is desired for practical utility.

A Posteriori Preference Specification

The basic idea of *a posteriori* specification is to find the Pareto set (or, in practice, some approximation thereof) in the solution space, and then allow decision maker to make a final selection from it. The main advantage of this approach is that it is usually easier for the decision maker to decide on a high-quality preference specification given a number of concrete design options than to do so beforehand in the abstract. There is no arbitrariness introduced into the solution process until the decision maker makes the final selection. The main disadvantage is that *a posteriori* methods are generally more expensive computationally and complex algorithmically than the *a priori* alternatives.

The simplest way to generate the Pareto front is to use an *a priori* optimizer a number of times, changing the preference parameters each time so as to (approximately) cover all possibilities. One advantage of this approach is that it only requires repeated solution of single-objective optimizations with different parameters each time, so that multi-objective capability is obtained without the need to formulate a dedicated multi-objective algorithm. This is particularly convenient if one wishes to use traditional techniques such as nonlinear programming that generally can only handle a single objective. However, use of nonlinear programming is usually only suitable for relatively simple problems, as any calculus-based technique by definition deals with derivative information, and is very likely to be sensitive to starting point and to be in danger of convergence to local minima. These qualities mean that some understanding of not only the objective function but the objective *space* is required, in order to choose starting points, assess local vs. global minimization, define nonlinear constraints, etc., which is not feasible for very complex problems.

This is not to say that nonlinear programming is the only option for repeated use of an *a priori* optimizer to approximate the Pareto front, since a wide variety of single-objective approaches are available. However, the Pareto front can be found directly by operation on a large population of candidate solutions with a single run of a genetic algorithm (GA). Since GAs are relatively simple algorithmically, require no derivative information, and have

global optimization capability, they have received a large amount of research attention in the field of multi-objective optimization in recent years, and are an attractive choice for use in the present application.

A number of different multi-objective genetic algorithms (MOGAs) have been proposed. One of the earliest is a non-Pareto-based approach known as the Vector Evaluating Genetic Algorithm or VEGA [8]. This essentially applies the methodology of a single-objective genetic algorithm by splitting the population into a number of sub-populations, one for each objective. These are then evaluated on a single-objective basis, and the results of each sub-population evaluation are combined to produce the next overall generation. Fourman [9] uses an essentially similar approach where solutions are compared two at a time in a "tournament" style, with a randomly selected single objective being used to decide each tournament. While these types of approaches are easy to implement, they suffer from crowding at the extremes of the solution space and poor coverage of the Pareto front [3].

The non-dominated sorting approach to multi-objective ranking of a population was first introduced by Goldberg [10]. This method divides the population into successive Pareto fronts and provides a preference-independent partial ordering, an important capability that is made possible by the operation on an entire population simultaneously that is characteristic of a GA. Srinivas and Deb built on this concept with the Non-Dominated Sorting Genetic Algorithm (NSGA), which uses sharing techniques on individuals of the same non-domination ranking [11]. They later published an improvement to the algorithm known as the NSGA-II [12]. Foseca and Fleming take a slightly different approach with their algorithm, which takes the general name of the multi-objective genetic algorithm (MOGA) [13]. Each individual is ranked according to how many other individuals it dominates, with the most dominant individuals being preferred.

The Niched Pareto Genetic Algorithm, developed by Horn et. al. [14], applies Pareto-optimality methods to the tournament approach. Successive pairs of individuals are com-

pared to a subset of the population. If only one is non-dominated in that subset, it wins the tournament. Otherwise, diversity considerations are used as a tiebreaker. Ziztler and Thiele's Strength Pareto Evolutionary Algorithm (SPEA) uses distinct sub-populations for the first non-domination class and for the rest of the population, and uses domination counts combined with the binary tournament system to perform selection [15].

Algorithm Selection

With the overview of the different approaches to a multi-objective optimization complete, a selection is made for use as the primary optimization algorithm in this thesis. An *a posteriori* method is desired, in order to execute autonomously (unlike the progressive approaches) and without the need for arbitrary prescription of preference information or parameter values (unlike the *a priori* approaches). The basic challenge faced by *a posteriori* methods is to compare different candidate solutions and rank them without preference information, an intrinsically difficult task due to the inevitability of tradeoffs between objective values. The wide range of genetic algorithms available have been shown to be effective solutions to this problem as well as powerful general-purpose optimization tools with the global optimization capability and gradientless nature needed to tackle highly nonlinear and incompletely understood situations.

NSGA-II, MOGA, NPGA, and SPEA can be considered the leading modern multi-objective genetic methods. The NSGA-II algorithm was selected as the primary optimization algorithm used in this thesis. It has a simple yet powerful approach that can be applied successfully to a wide variety of problems, making it extremely popular in many areas of optimization (as witnessed by the almost 10,000 citations the original article has received as of this writing), and has been considered to represent the state-of-the-art in multi-objective genetic methods [16], or at least to share that distinction with SPEA [17]. It uses non-domination sorting with a parameterless diversity-based secondary method, both of which require no arbitrary parameters to be specified, but does not sacrifice effectiveness

to acheive this generality.

1.2.2 Mechanism-Specific Optimization

As was previously stated, an investigation of multi-objective optimization involving kinematic and non-kinematic objectives can be approached based on either the general multi-objective literature or the kinematic optimization literature. The former approach has yielded a choice of general optimization algorithm. How it is employed will be informed by knowledge of existing techniques specific to the optimization of mechanisms, which are discussed in this section.

Synthesis Techniques

There is an old and well-developed body of research into techniques for the design of mechanisms to achieve purely kinematic goals. Some of the oldest work deals with exact synthesis of mechanisms to achieve desired motion characteristics at a number of precision positions. If satisfaction of the motion condition at each position is interpreted as an objective, these techniques can be viewed as a special case of general optimization, where the objectives are so simple and well-defined that exact satisfaction of all objectives using direct analytical solution is possible. The limitations of these methods are that only kinematic objectives are possible, and only a small number of positions can be prescribed, with the motion being uncontrolled at other points. The original work on the classic synthesis techniques can be found in Sandor [18], Erdman [19], Kaufmann [20], and Loerch et. al. [21].

Adaptations of the precision point approach that allow higher numbers of points to be specified in exchange for relaxing the precision of the synthesis also exist. Kramer and Sandor presented the Selective Precision Synthesis technique [22], where the synthesized path need only pass within a specified distance of each prescribed point instead of hitting it exactly. A variation is described by Mirth [23], who uses a combination of solving

three exact point prescriptions using Burmester theory with a number of additional quasi-precision points that serve to narrow down an optimized final selection from the infinite number of Burmester solutions. A number of authors have described least-squares type approaches that perform an approximate fit to a prescribed function [24] [25], which is the same sort of strategy but with the prescription information distributed over the entire function rather than concentrated at a few precision or quasi-precision positions.

While these techniques have succesfully addressed the solution of the path, motion, and function generation problems found in kinematic design, they are unable to solve multi-objective, multi-domain optimization problems. Only a single objective, meeting the prescribed kinematic behavior, is possible, and the integration of other disciplines is not part of these methods. The method set out in this thesis leaves room for integrating these kinematic techniques in the preliminary analysis phase (see Section 2.4.2), but something else is needed for solution of the overall problem.

Multi-Objective Kinematic Optimization

Early attempts at multi-objective optimization inevitably suffered from lack of modern computing resources and the undeveloped stage of genetic algorithms and other meta-heuristics with similar capabilities. For example, Conte et. al [26] optimized a number of dynamic characteristics of the pin-jointed four-bar linkage, making the same observation as is made in Section 2.4.2 that the outcome of a traditional synthesis procedure is often a design space of dimension greater than zero (that is, more than one linkage will work), and that a secondary optimization conducted in this reduced space will automatically satisfy the synthesis requirements while doing better at non-kinematic criteria than an arbitrary choice of synthesis solution. However, the optimization technique presented in this paper is unsophisticated, relying on ad hoc sequential application of single-objective nonlinear programming to in essence manually conduct an objective trade-off. A somewhat more sophisticated approach is taken by Kakatsios and Tricamo [27], who conduct an integrated

kinematic-structural optimization considering the dynamically imposed stresses in a high speed mechanism. They use a one-dimensional beam element FEA formulation for their links, and employ what is essentially an *a priori* weighted sum type approach to preference specification by lumping together kinematic path error and elastic deflection of the precision point. However, the use of nonlinear programming means that extension of this technique to problems more complex than the pin-jointed four-bar would be difficult, and their preference specification approach is non-generalizable as it is specific to the combination of path approximation and elastic deflection calculations. Rao and Kaplan [28] and Krishnamurty and Turcic [29] take fundamentally similar approaches, using nonlinear programming and *a priori* preference specification.

These types of early attempts achieved limited success at optimization of specific problems and certainly represented a big improvement over the usual practice of kinematic synthesis without simultaneous consideration of dynamic consequences. However, the emergence of powerful metaheuristics that operate on entire populations with global convergence capability signified a fundamental change in the way mechanisms can be optimized. The use of genetic algorithms with Pareto-based *a posteriori* preference specification has been recognized as being superior in many ways to the fundamentally limited traditional approach of using single-objective or *a priori* nonlinear programming techniques.

This being said, the majority of applications of GAs to date in the kinematics literature have been concerned with single-objective optimization of a kinematic objective only. That is, while better tools are available, the same limited problem is being solved. For example, Kunjur and Krishnamurty [30], Cabrera et. al. [31], Laribi et. al. [32], and Kanarachos et. al [33] all use single-objective GAs to synthesize a linkage to approximate a prescribed precision point motion. Fang [34] and Zhou and Cheung [35] perform similar tasks, but with the addition of novel features such as simultaneous type synthesis and an adjustable linkage respectively.

Treatments of multi-objective optimization of mechanisms do exist as well. Cabrera

et. al. [36] use a Pareto-based GA to simultaneously optimize the kinematics and grasping ability of a robotic manipulator. Nariman-Zadeh et. al. [37] modify the NSGA-II algorithm to solve a two-objective problem which synthesizes a four-bar for the usual path tracking objective but also minimizes deviation of transmission angles from 90 degrees. Khorshidi et. al. [38] similarly optimize for path tracking, transmission angle deviation, and also worst-case mechanical advantage, but introduce an integrated local search refinement that compensates for the tendency of GAs to miss the exact locations of optima with their limited number of objective function calls. El-Kribi et. al. [39] use NSGA-II to minimize required input torque and input velocity fluctuation, and take the innovative approach of using selection from a list of actual motors as a discrete design variable.

These cases of multi-objective optimization of mechanisms differ from the present work in two key respects. First, they are fundamentally single-disciplinary; that is, they restrict themselves to optimization of kinematic and dynamic properties of mechanisms, the realm of traditional kinematics. Second, they are all single-level, and do not make use of a nested optimization approach. This approach is rare in the literature, perhaps due to the fact that its additional computational expense is not justified unless a multi-domain problem must be solved. It is not completely novel, and is used for example in the work of Fathy et. al. [40] who conduct a plant/controller optimization problem and point out that non-nested approaches fail to guarantee system-level optimality. However, the author is unaware of any existing literature having all elements of the present work, that is, that use Pareto-based genetic algorithms to solve a nested multi-domain optimization problem relating to mechanism design.

1.2.3 Optimization of Elastic Structures

As previously stated, a sizeable body of research exists with respect to the optimization of elastic structures, a topic of great importance to the rescue spreader problem. This review's findings are organized according to Figure 1.3, which is based on the comprehensive

and well-organized survey by Saitou et. al. [41].

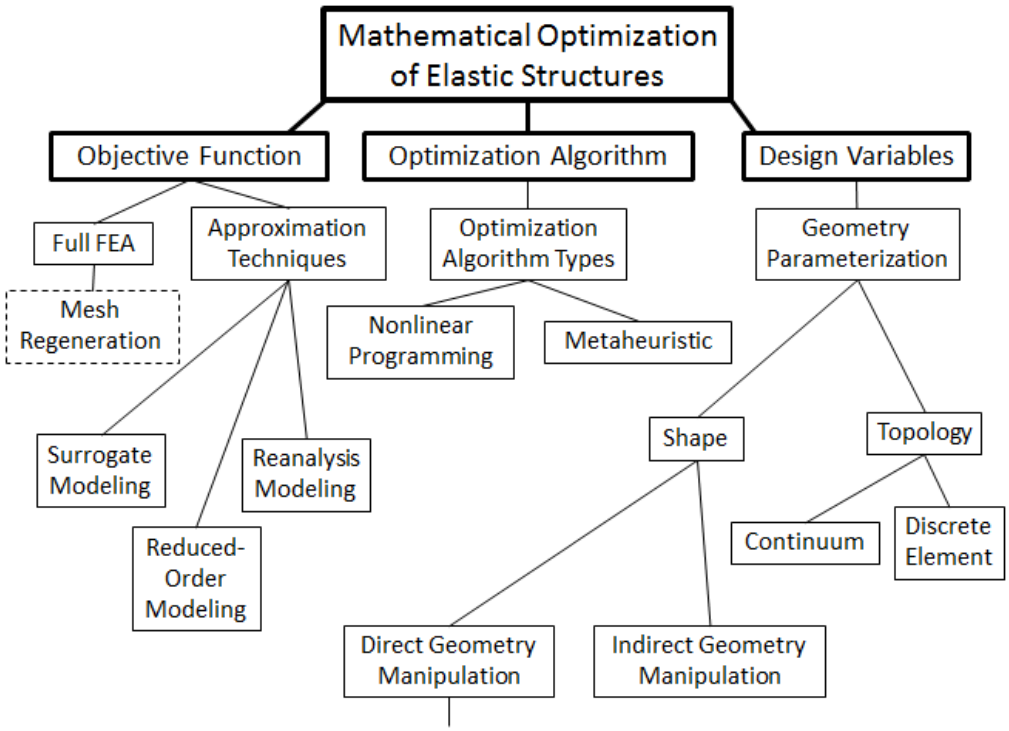


Figure 1.3: Hierarchical organization of the literature on optimization of elastic structures

The structure of a mathematical optimization consists of the choice of objective function, optimization algorithm, and design variables. In the case of elastic structures, the options for the objective function essentially consist of using a full finite element analysis (FEA), which can in most cases provide an arbitrarily close approximation to the exact solution, or one of a number of less accurate approximate techniques if the computation required for finite element analysis is too expensive. Any of the wide variety of general-purpose mathematical optimization algorithms are suitable for use with elastic structures, and as this is in itself a whole field of research that this thesis does not go into in any depth it suffices to distinguish between nonlinear programming algorithms using a serial, gradient-descent approach, and meta-heuristic algorithms such as genetic or particle swarm that use parallel processes inspired by natural phenomena. Finally, the choice of design variables

amounts to the method by which one chooses to parameterize the shape and topology of the elastic structure. Again, as topological optimization does not play a role in this thesis exploration of this sub-field is restricted to distinguishing between the two basic options, the continuum and discrete element methods. Shape optimization, on the other hand, is of great importance to the first case study and the literature in this sub-category receives a special focus.

Objective Function

The preferred method of evaluating a complex structure is a full finite element analysis. Surveying the full FEA literature would be a massive undertaking well outside the scope of this review. The FEA used in this thesis is of a very basic nature, using techniques that have been mature for several decades and that can be found in any standard elementary text on the subject. This is due to the fact that structural analysis of a linear elastic continuum is just about the simplest application that FEA can be applied to.

The one aspect of the FEA approach that is nontrivial in this application, and is thus worth addressing here, is that of mesh regeneration. This is shown in a dashed box in Figure 1.3 to indicate that it is an integral part of the FEA process that is singled out as worthy of attention in this review. Depending on the exact circumstances, mesh generation can be one of the most time-consuming aspects of an FEA procedure. An optimization algorithm that uses an FEA-based objective function will need to generate a large number of meshes, which will likely take an impractical amount of time if the naive approach of simply reusing the procudure for generating the mesh from scratch each time is taken. One can exploit the fact that if the optimization only perturbs the candidate solution by a relatively small amount each time, the geometry will be sufficiently similar that the mesh can be incrementally changed instead of regenerated completely. This is often the case if a gradient-descent approach is used, but not if a metaheuristic method such as a genetic algorithm is used.

One can distinguish between two basic categories of mesh regeneration techniques: structured and unstructured.There is an inherent trade-off between generality and efficiency in meshing. At one end of the spectrum are the powerful generalized meshing algorithms used by commercial software that must be able to deal with almost any conceivable geometry that the user supplies. At the other end is a fixed shape in a custom code that the designer meshes by hand, and that the code need only look up and reapply each time. A structured approach is generally more efficient, and makes use of some assumptions about the shape and topology of the domain to reduce the generality of the meshing task. As this may not be possible for geometries that undergo large changes during the optimization, it is sometimes necessary to use a more expensive unstructured approach, which does not benefit from knowledge of the general nature of the domain and has to "go in blind".

Most structured techniques are based on transfinite interpolation, or TFI [42]. TFI allows the construction of a new function over a planar domain in a way that is guaranteed to match a specified function at the boundary of that domain [43]. Consider the shape perturbation in Figure 1.4. A mesh exists for the original domain A, and the goal is to adapt the mesh for the perturbed domain B without remeshing from scratch. The mesh is fully specified by two sets of data: geometric data that locates the position of each node in the space, and topological data that determines which nodes are connnected to form element edges. Part of the value of using a structured approach for relatively small changes in domain geometry is that the topology of the mesh can be left untouched. It then only remains to determine a function that maps the original position of each node in A to the new position in B. This is accomplished using TFI, which solves the most difficult aspect of the construction of such a function, namely matching the arbitrarily shaped boundaries. TFI has less control over what happens in the interior of the domain, but one does not necessarily care about the details of the mapping as long as excessive element distortion does not occur. Examples of TFI-based techniques can be found in Gaitonde and Fiddes [44], who use exponential blending functions (the blending functions are used to determine

the transformation in the domain interior from the boundary transformation), and in Soni [45], who uses arc length blending functions.

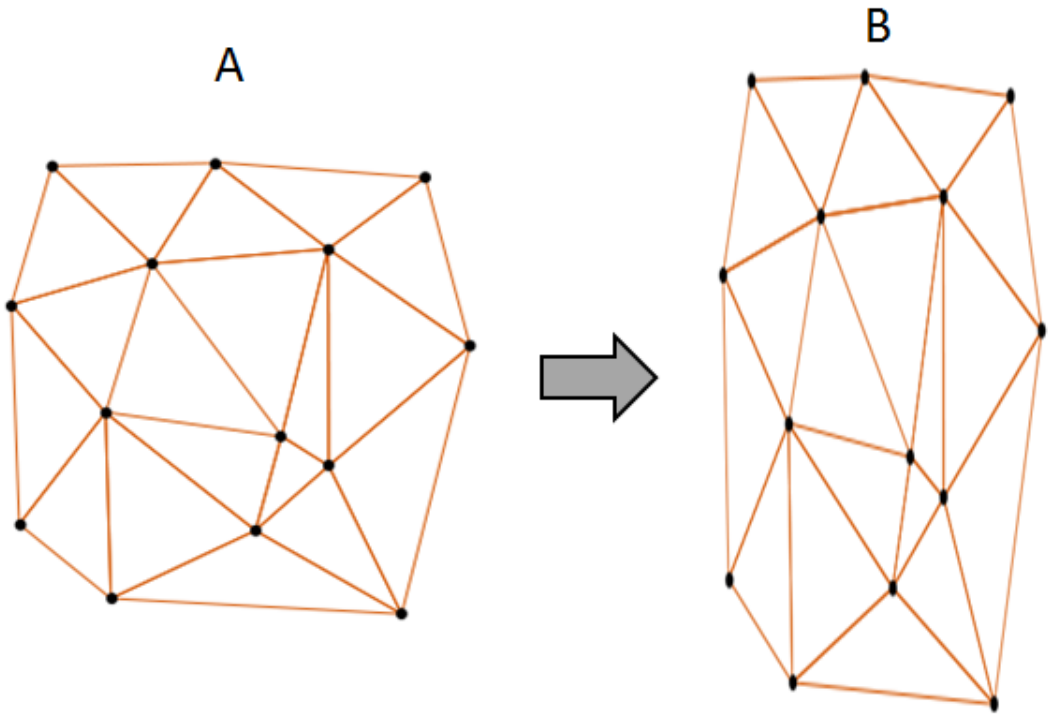


Figure 1.4: Example of structured mesh transformation without topological change

Unstructured techniques face greater challenges for the reasons discussed. Botkin [46] adopts the naive approach of complete regeneration, with an important caveat. He uses a gradient descent optimization algorithm where the gradient is approximated by finite difference calculations at each step. This finite difference calculation requires a small localized change to be made to the geometry for each dimension of the design space. Botkin is thus able to keep the bulk of the mesh unchanged and only remesh a small portion in the vicinity of the perturbed geometry. However, this technique does not necessarily address the need to remesh after an optimization step finally occurs after the gradient is computed, when the entire geometry is changed. Other authors attempt to imitate natural phenomena to automatically deform their meshes in a generalizable way. Batina [47] models his

mesh as a network of springs, with each edge representing a spring of stiffness inversely propotional to the distance between the nodes it connects. The idea is that a natural equilibrium should form where the increased stiffness of shorter edges prevents element aspect ratios and relative sizes from degenerating. Every time the domain boundary is changed, a new equilibrium calculation is used to reposition the interior nodes. However, this requires multiple iterations and essentially amounts to solving a number of simple FEA problems before the solution of the actual FEA problem each time, which reduces the computational cost savings. Chiandussi et. al. [48] take this idea a bit further and essentially solve two FEA problems per cycle. The first is a structural problem where the domain is treated as an elastic body subject to a displacement boundary condition that deforms the body into the desired updated shape. The displacement field in the interior that results from the solution of this problem is used to relocate the mesh nodes, and the updated mesh is then used in the actual FEA problem. This is an imaginative technique that is highly generalizable, but also suffers from a relatively high computational cost as an additional FEA problem must be solved. If the solution of the second problem takes much longer, which is quite possible as many FEA problems are much more complex than the simple linear elastic model needed to solve the first problem, than the relative cost may be low enough that this method is a good choice. On the other hand, if the actual FEA problem is of the same order of complexity as the shape deformation problem, the computational cost will roughly double and this may not be the best method to use.

The most effective way to mitigate the computational cost of FEA is to find a replacement technique that obviates the need to do it at all. The most common approaches are surrogate modeling, reduced-order modeling, and reanalysis modeling.

Surrogate modeling can be thought of as a "black box" method, where a relatively simple mathematical function is used instead of the full objective function to assign an output value to a particular set of design variable values. This replacement function is not derived from any consideration of the underlying physics of the process but is rather an

empirical fit to a set of input and output data considered abstractly. This technique is of course dependent on having an existing set of input and output data, perhaps from pre-computation of a number of objective function calls, and can be thought of as applying a curve fit to a data set. An example of the application of surrogate modeling can be found in Forsberg and Nilsson [49], who use polynomial regression to avoid high-cost calls to a crash simulation objective function. While this method can be very effective under the right circumstances, its main limitation is the need for some type of data set to already exist to base the surrogate model on, which is not the case in this thesis.

Reduced-order modeling, in contrast, does use a physically grounded mathematical formulation, but at a lower complexity level than a full FEA analysis. This method can be thought of as applying an analytical first-approximation to a problem whose full complexity does not admit analytical solution. For example, Lyu et. al. [50] model an automotive torsion-beam suspension by breaking the assembly into a number of discrete components, each of which is modeled as a nonlinear bending or torsion spring. The stiffnesses of each component are determined by pre-computation of a number of representative cases using the full FEA approach, which is however less expensive for an isolated component than for the entire assembly. Surrogate modeling is employed to extrapolate from the test cases to a continuous model, and the stiffnesses thus obtained are used in the spring-system analytical model for the entire assembly. While this was an excellent approach given the nature of the system to be modeled, it would be difficult to apply in the rescue spreader case as each link is essentially a monolithic entity and cannot be sensibly reduced to structural subcomponents.

Reanalysis modeling uses the results of a single full-complexity analysis to estimate the results of small changes to the design. It is analagous to linearization of a nonlinear function in a small neighborhood about a particular point. This technique is best used for fine-scale optimization based on an approximate optimum generated by a different technique, or for design sensitvity analysis. Examples of both topological [51] and geometrical [52]

reanalysis are performed by Kirsch and Papalambros. It would be difficult to use this technique in instances, such as the rescue spreader problem, where the solution has not been narrowed down to a particular neighborhood. An attempt by the author to use this type of method for optimizing a novel rescue spreader design is presented in [53], which did not achieve sufficient accuracy due to too large a departure from the fully analyzed reference design.

While the approximation techniques presented here are quite powerful given the right conditions, none of them are useful in the present case as they are generally limited by the need to have existing solution information to work with, which is unfortunately not the case here.

Design Variables

Having dealt with the literature surrounding objective function options, and omitting an exploration of the literature on optimization algorithms for the reasons stated previously, it now remains to examine the means by which the geometry of the elastic body can be parameterized so as to formulate a set of design variables. It proved both desirable, in order to reduce the size of the design space and the complexity of the FEA calculations, and possible, given that a planar linkage is being examined, to treat all elastic bodies as 2D parts of constant thickness in this work. The following discussion will be made with this in mind, but can be extended to three dimensions without difficulty.

Consider Figure 1.5a, which shows an I-beam cross-section. This is a 2D part of (large) constant thickness, the geometry of which can be parameterized easily by specifying the relevant dimensions. Now consider Figure 1.5b, which shows a general 2D body. Since the boundary is represented by an arbitrary function which in general cannot be put in terms of the usual arsenal of mathematical functions such as polynomials, lines, arcs, etc., it is not clear how to represent the shape of the boundary with a finite set of numerical values, i.e. design variables.

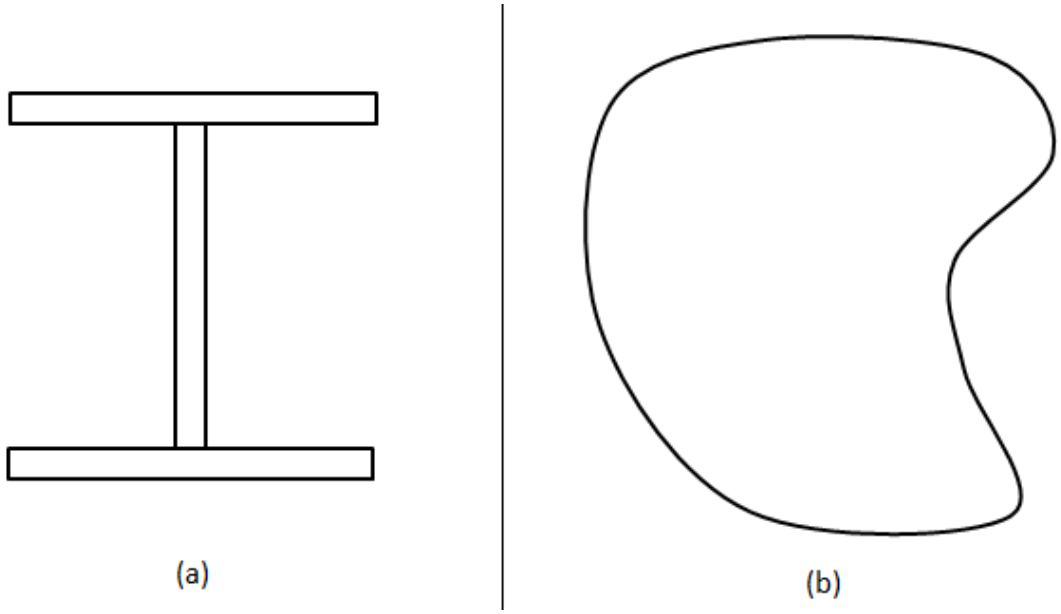


Figure 1.5: Modeling explicit vs. arbitrary geometry

In addition to the need to represent the shape of the boundaries, the topology of the part must also be parameterized in some way, unless a fixed topology is acceptable. The difference between shape and topological change is illustrated in Figure 1.6. Since this thesis deals with fairly simple parts the necessary topology can be manually selected, and topology parameterization is not needed. The interested reader is referred to the survey of topological methods by Papalambros and Shea [54].

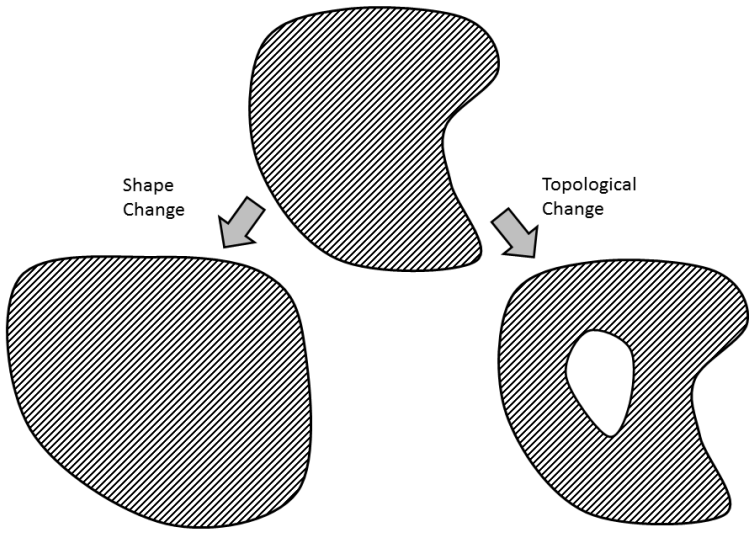


Figure 1.6: Shape change vs. topological change

Turning to shape parameterization, there are two possibile approaches: direct and indirect geometry manipulation. In the direct approach, the design variables directly represent the boundary shape in some way, for example as points on the boundary to be interpolated between, or as control points for a spline representation. In the indirect approach, the design variables indirectly influence the boundary shape, for example constituting fictitious loads which act to deform the boundary.

Dealing first with the indirect approach, a good example is to be found in Belegundu and Rajan [55], who optimize a fictitious load on the part boundary to deform the part so as to best support a fixed real load. This method is very similar to, and easily integrated with, the previously mentioned unstructured mesh regeneration technique used by Chiandussi et. al. [48]. A similar but more mathematically sophisticated method is given by Azegami et. al. [56]. Again, as the cost of the solution of the fictitious problem is of roughly the same magnitude as solution of the actual problem when the latter is also a linear elastic analysis, this is not a suitable method to use in the rescue spreader case.

A more detailed look is warranted at the direct approach, as this is the approach selected

for use in this thesis. This section is partly based on the excellent survey of the relevant literature by Samareh [42], and is subcategorized as shown in Figure 1.7.

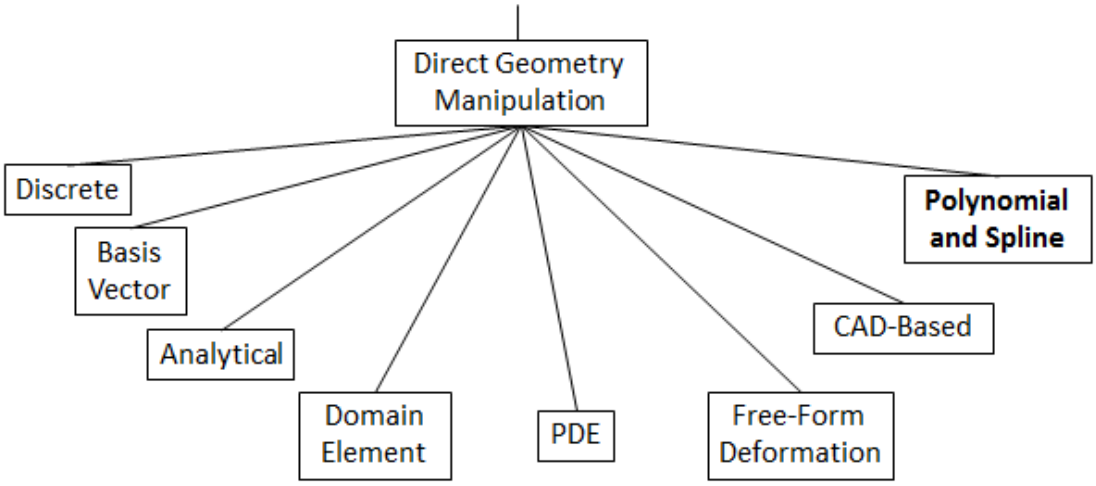


Figure 1.7: Direct geometry manipulation methods

The most obvious approach, known as the discrete approach, is to simply treat the locations of the finite element mesh nodes on the part boundary as the design variables. This approach suffers from two major drawbacks, however. The first is that unless the mesh is extremely coarse, there will be a very large number of design variables, which will result in very long run times for the optimization algorithm. The second is that the optimization is blind to the need for the smooth part boundaries that are necessary for a realistic design. It has been well known for decades that this type of naive approach will produce jagged and unusable results. For example, Figure 1.8 is taken from Haftka and Grandhi [57]. Figure 1.8a shows the initial state of a design for a square plate with a square hole in the middle under outwards loading, with Figure 1.8b showing the result after an attempted optimization. Figure 1.8c shows the results of an attempt to relax the constant thickness requirement for a 2D part and allow the thickness of each element to vary, again producing a nonsensical result.

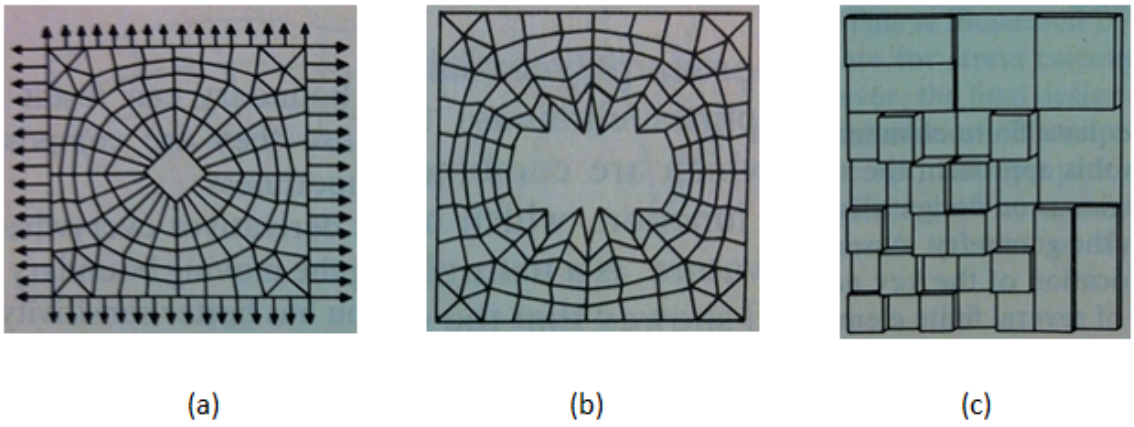


Figure 1.8: Typical poor results from an unconstrained discrete approach

There have been a number of attempts to improve the discrete approach. A couple of recent examples are the work of Scherer and Steinmann [58], as well as that of Arnout et. al. [59]. Scherer and Steinmann use a fictitious deformation energy constraint to limit the amount of permissible deformation, which prevents the formation of a jagged boundary. However, the problem of the large set of design variables leading to slow convergence persists. Arnout et. al. apply what is essentially an exponentially weighted average of the element stresses (the Kreisselmeier-Steinhauser function) in an attempt to eliminate the high local stress sensitivity and achieve a similar result, again with some success. Generally speaking, it seems unlikely that adopting the discrete method in the interests of simplicity is a wise choice, as the trouble one must go to to impose an effective constraint will likely be greater than that required to simply parameterize the geometry in a different way.

The basis vector approach can also directly operate on nodal coordinates, but instead of individually varying each one a set of predetermined vectors of design variable values are chosen, forming the basis of a vector space of substantially smaller dimension than that formed by considering each design variable independently. In the approach used by Pickett et. al. [60], $\bar{R}$ is a full set of design variables, i.e. a completely specified shape, given by

$$\bar{R} = \bar{r} + \sum_{n} \bar{\nu_n}\, \bar{U_n} \tag{1.5}$$

where $\bar{r}$ is the baseline shape, $\bar{U_n}$ are the basis vectors of the reduced space, and $\bar{\nu_n}$ are the quantities to be optimized specifying the point in the vector space, i.e. the particular linear combination of predetermined sets of design variable values used. This method strikes a good balance of compactness and flexibility, although it suffers from the need to define the basis vectors carefully in order to maximize the possibility that the optimal design lies in the subspace of the original design space that is spanned by the reduced basis. In other words, if the predetermined sets of design variables are poorly chosen, it might be the case that there is no way to combine them to yield the optimal solution, which will remain undetected.

The analytical method is quite similar to the basis vector method, but uses linear combinations of predetermined shape functions instead of predetermined design variable vectors. The classic example of this approach is the 1976 paper by Kristensen and Madsen on the optimal shape of fillets in a 2D plate [61]. The outline of the fillet is given by a univariate function in polar coordinates $r(\theta)$, defined as

$$r(\theta) = r_0(\theta) + \sum_{i=1}^{I} t_i\, f_i(\theta), \qquad 0 \leq \theta\ \leq \theta_0 \tag{1.6}$$

where f_i are the predetermined shape functions, r_0 is the baseline shape function, and t_i are the set of I design variables. Similar curves are placed at radial intervals running roughly parallel to the fillet edge, and the mesh nodes are placed at the intersections of the set of curves thus formed and radial lines at angular intervals. The relevant illustration from the paper is reproduced here as Figure 1.9.

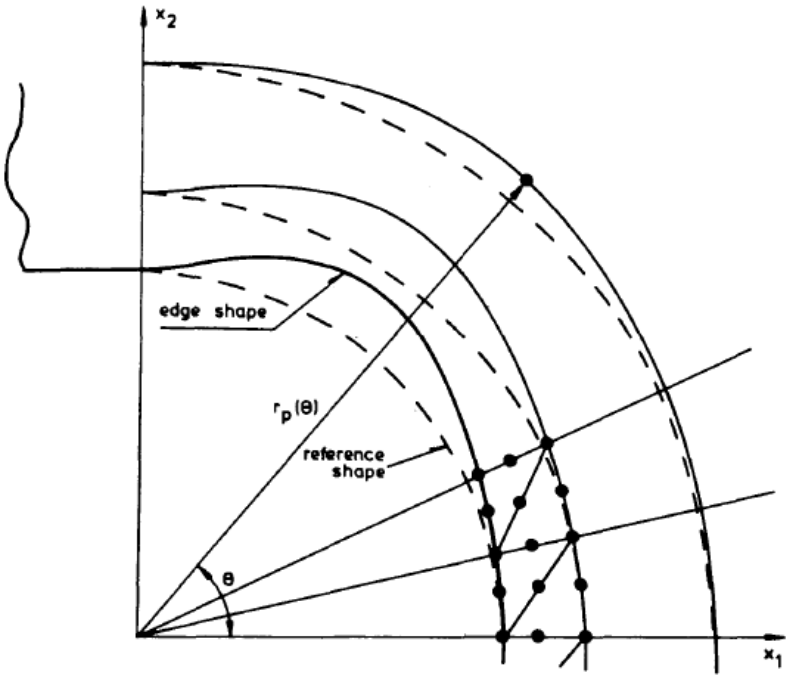


Figure 1.9: Geometry and mesh construction in Kristensen and Madsen [61]

This method shares the same basic strengths and weaknesses as the basis vector method. In the case of a fillet in a plate it works well as the choice of shape functions is straightforward for such a simple geometry. For more complex geometries the limited, arbitrary basis for the solution space will likely be a liability.

The domain element approach allows parameterization of the mesh using a reduced set of nodal coordinates, with geometry parameterization occuring as a consequence of changing the location of the mesh boundary. The mesh is divided into a set of macroelements, each of which contains multiple mesh elements. When the macroelements are deformed, their component elements deform as well, maintaining a constant parametric location within the macroelement, as shown in Figure 1.10.

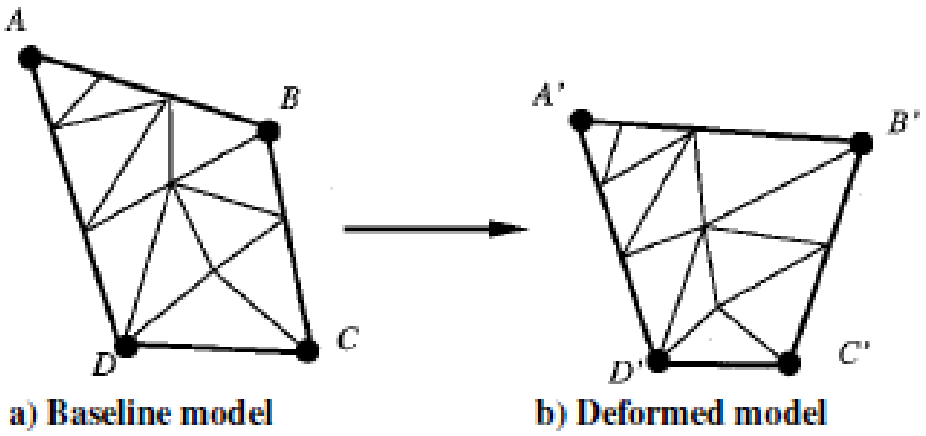


Figure 1.10: Illustration of domain element approach [42]

This technique is very simple and efficient and behaves like discrete parameterization of a coarse mesh, while achieving the resolution of a fine mesh. It may suffer from the jagged boundary problems of a discrete method, albeit at reduced severity due to the reduced spatial density of the mobile nodes, and large deformations of the macroelements will result in poor aspect ratio of the component elements.

Bloor and Wilson [62] propose the imaginative approach of treating domain boundary shapes as the solutions to elliptical partial differential equations, with the design variables being parameters in the PDEs that alter the shape of the solution surfaces when varied. This method can represent complex surfaces using only a few variables, but being tailored toward representation of complex surface geometries in three dimensions is not well suited to the task at hand.

The free-form deformation approach uses techniques originally developed for computer graphics to apply global tranformations to a 3D body. Thus specifying a small number of transformation types and parameters can generate a variety of global changes to a baseline geometry. Examples can be found in Barr [63] and Sederberg and Parry [64]. The implementation of this method is complex, and the global 3D transformations it relies on are not appropriate for the simpler 2D modeling with local shape refinement required in this thesis.

CAD and FEA software packages have become fundamental to structural design, and the ease of interface between, for example, SolidWorks and ANSYS has gone a long way towards streamlining commercial workflows and putting powerful capabilities in the hands of nonspecialists in structural analysis. It is only natural then that adding optimization capability to this package is a goal that has received a significant amount of attention. Chen and Tortorelli [65], for instance, optimize a 3D connecting rod using a CAD-based approach. However, they restrict themselves to variation in the sizes of predefined features, and this approach generally seems to be more attractive from a commercial convenience standpoint than on its fundamental merits. In any event it is not usable in the case of this thesis as the computer implementation is done entirely in Matlab, in order to eliminate the computational overhead of generating a cross-program call for every FEA analysis and to take advantage of the powerful general-purpose programming environment Matlab provides.

Polynomial functions are natural choices for fitting to arbitrary data points given their flexibility and capacity to fit any number of points if a sufficiently high order is selected. This approach was tried early on in the development of numerical shape optimization, notably by Bhavikatti and Ramakrishnan [66], who built off the work of Kristensen and Madsen and optimized flat and round fillets using a polynomial representation. This approach worked well for the simple shape of a fillet, which only required a low-order representation, but more complex arbitrary geometry creates serious difficulties. High-order polynomials tend to be numerically unstable, and evaluating the interpolating function becomes quite expensive computationally. Furthermore, convergence of the optimization algorithm suffers as local changes to a data point result in global adjustment of the function, so that partial derivatives of the objective function value with respect to individual design variables become very difficult to establish.

In response to these difficulties, a number of spline approaches were proposed. The use of B-splines, introduced by Braibant and Fleury in 1984 [67], is the method adopted in

this thesis. The B-spline method will be explained in detail later on, but for now it suffices to qualitatively describe some important properties. A spline of any order may be used, with by far the most popular being cubic, which acheives a good balance of flexibility and simplicity. The spline is defined by a set of control points, with only a subset of these points (four in the cubic case) influencing the curve at any given interval of the spline. This results in the local control property, wherein changes to a design variable (i.e. a control point) have only local effects, in contrast to the troublesome global perturbation to a simple polynomial. Additionally, as the spline is of low order, function evaluations are much cheaper than in the polynomial case. A complex shape can be represented by a few control points without significant loss of accuracy, as illustrated in Figure 1.11, taken from [42]. Finally, the spline always lies within the convex hull of the control points; in fact, a stronger version of this property holds wherein a given curve segment lies within the convex hull of the subset of control points that define it. This is very convenient for easily and efficiently applying constraints, e.g. ensuring that the curve does not enter a certain region as the control points are moved.

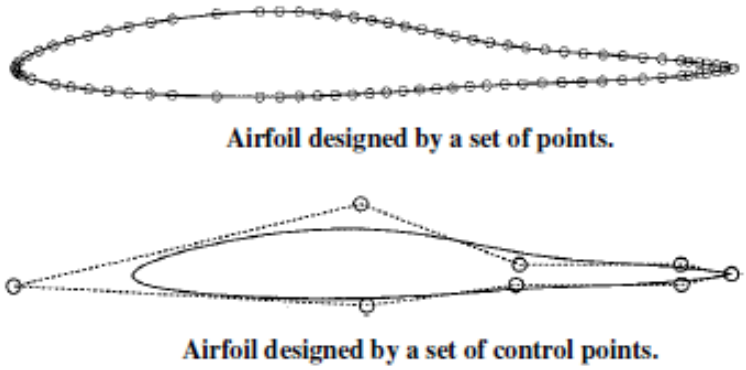


Figure 1.11: Airfoil represented directly and by a closed B-spline [42]

The only real drawback to the B-spline method is the inability to represent conic sections exactly. This is not an issue in some cases, as in this thesis where such shapes need not be specified, but if this capability is desired a slightly more complex spline formulation

known as NURBS (non-uniform rational B-splines) is available. See Schramm et. al. [68] for a case where a particular application (optimization of thin-walled beam cross-sections) requires this capability.

1.2.4 Six-Bar Linkages in Engine Design

Turning now to the second case study, that of the six-bar linkage for piston profile control in an IC engine, a brief discussion is given of previous related applications. The use of a six-bar linkage to achieve a variable compression ratio (VCR) capability is a fairly well-studied problem, although it is distinct from the optimization of a fixed piston profile. An SAE paper published by Pouliot et. al. [69] describes the design and mechanical implementation of such a linkage, with extensive experimental validation. At the other end of the spectrum is a type synthesis study by Freudenstein and Maki [70] that exhaustively examines suitable linkage topologies. More recent work by Ozcan and Yamin [71] and Yamin and Dado [72] approaches the problem from a numerical combustion modeling standpoint, proving the utility of VCR mechanisms in terms of performance and efficiency.

As none of this work is on the present problem per se, it is not directly relevant, but two important points can be made. First, the use of complex linkages in IC engines is a practical and well-studied proposition, so that a theoretically promising but complex linkage should not be rejected out of hand based on worries about practicality. Second, the need for a multi-domain synthesis of different areas of expertise is particularly evident in examining this area of the literature. For example, Freudenstein and Maki's topological study is very impressive from a kinematic standpoint, but does not go beyond the area of type synthesis into determining the actual dimensions that would optimize the combustion objectives, which is, after all, the whole point. Conversely, the authors of [71] and [72] exhibit great expertise in their detailed simulation of the thermodynamic behavior associated with the mechanism, but are unable to relate this to the kinematic aspect. In fact, Yamin and Dado are reduced to measuring the link dimensions from a figure generated by Freudenstein and

Maki in a patent resulting from their type synthesis in order to have something to plug into their simulation to generate the piston trajectories. As will be seen in Chapter 4, the method described in this thesis is able to capture the entire scope of the problem in a unified optimization.

While the adjustable VCR linkage has received much attention, the optimization of a fixed piston trajectory has been attempted as well. In his Master's thesis [73], T.A. Good tackles exactly the problem discussed in Chapter 4 of this thesis, that of optimizing the non-adjustable Stephenson-III linkage for maximum combustion efficiency. Furthermore, this work represents a rare multi-domain mechanism optimization, in that it includes both combustion simulation and kinematic analysis. However, the exact means by which Good attempts to solve this challenging problem differs significantly from the strategy adopted here. The basic procedure he uses is

1. Run a preliminary optimization using a combustion simulator as the objective function to determine the most efficient piston trajectory. The design variables are purely mathematical parameters of the trajectory shape such as the height of the peaks, and no thought is given to the mechanical implementation that will generate this curve.

2. Find a mathematical function that has these characteristics, namely by use of a fourth-order finite trigonometric series representation.

3. Run a kinematic optimization using an overall-error minimization objective function (similar to the least-squares type family of synthesis techniques) to find linkage dimensions that will generate the desired trajectory as closely as possible.

4. Once a solution is obtained, run a counterweighting optimization to balance the mechanism (similar to the inner loop optimization in this thesis).

Traditional nonlinear programming or gradient-descent optimization techniques are used throughout. It will also be noted that each optimization in this sequence is single-

objective. By the author's own admission, avoidance of local minima using nonlinear programming is very challenging given the complexity of the problem, and, given the absence of control over overall size and mass as well as the complete lack of transmission angle control, the solutions obtained "may not make practical, or even possible, mechanisms". Additionally, while both combustion and mechanical objectives are considered, the mechanical objectives are only optimized for the single solution to the combustion optimization. This in effect completely prioritizes the combustion objective over the mechanical one. If for example there were a solution that were 1% worse with respect to efficiency and 1000% better with respect to balancing, this approach would never find it.

Since exactly the same problem is being solved, it is useful to contrast the method presented in this thesis with that described above. The global optimization capability of the genetic algorithm used overcomes the difficulties presented to gradient descent techniques by the nonlinearity of the objective function. Use of the problem formulation strategies described in Chapter 2 guarantees constructable mechanisms with acceptable transmission angles, without even the need to specify constraints. The Pareto-based, *a posteriori* multi-objective basis of the optimization allows simultaneous consideration of all combustion and mechanical objectives. Finally, the use of the linkage dimensions as the design variables means that the piston trajectory is parameterized directly by the linkage, avoiding the awkward process of mathematically parameterizing the trajectory, approximating by a trigonometric series, and conducting an error-minimization synthesis, each step of which requires an arbitrary parameterization and introduces approximation error.

In Good's defense, his work dates from 1995, at which point the approach taken in this thesis would have been completely infeasible computationally on all but a powerful supercomputer. Even after two decades of Moore's Law increases in processing power, a typical problem still takes several days to solve using the present approach. Additionally, a significant portion of the theory and algorithm development in the field of multi-objective genetic algorithms has taken place fairly recently; for example, the paper describing the

NSGA-II algorithm used in this thesis was only published in 2002.

1.3 Overview

This thesis presents a generalized approach by which the multi-domain, multi-objective optimization of a mechanism may be efficiently formulated, and applies this approach to two case studies. Chapter 1 has introduced the growing importance of computational optimization techniques and the current dearth of research into integrated optimization of mechanisms. A general approach to this task is presented in Chapter 2, wherein indirect design variable formulation, design space reduction, and a nested structure are used when formulating the optimization. Chapter 3 discusses the first case study, that of the rescue spreader, the main feature of which is integration of finite element modeling into the inner level optimization to achieve efficient structural design. Chapter 4 then covers the second case study, that of the IC engine, which combines thermodynamic analysis of the combustion process with mechanical analysis of the linkage. Concluding remarks are then given in Chapter 5.

Chapter 2:

General Method

This chapter lays out a generalized approach by which the multi-domain, multi-objective optimization of a mechanism may be efficiently formulated. As there is an endless variety to the types of design situations that may be encountered, it is impossible to reach a high level of specificity without losing generality, and the application of the method to a specific problem must be left up to the designer. However, there is still much to be gained by the elucidation of the overall technique, which consists of three main strategies: indirect design variable specification, dimensional reduction of the design space using preliminary kinematic analysis, and the use of a nested optimization structure.

2.1 Optimization Theory

As the remainder of this thesis is concerned with the solution of optimization problems, it makes sense to begin with a brief introduction to optimization theory, with an emphasis on the multi-objective case.

2.1.1 Conceptual Introduction

In any engineering task, there are many different ways to construct a device to accomplish the designer's goals. A given design can be described (in simplified form) by specification of a set of numerical values representing selected quantitative aspects of the design. For example, there are many ways to make a rectangular table, but the design can be completely specified, within a certain level of abstraction, by listing the length, width, height, and number of legs that the table will have. The quantities that specify the choice of design will be referred to in this thesis as *design variables*, although they are also often referred to as the *parameters* of an optimization. Similarly, the quality of a particular design choice can be expressed quantitatively by defining one or more *objectives*. Returning to the table example, the quality of a particular design might be evaluated by the weight that it can support without collapsing. It is usually the case that the design variables can be directly specified *a priori*, but the objective values can only be calculated in an *a posteriori* fashion. That is, one can build a table given the length, width, height, and number of legs that the table will have, but one cannot build a table given only the weight that it will be able to support.

It is, however, possible to determine how much weight a *particular* table will be able to support once one specifies the length, width, height, and number of legs. This process, the evaluation of a particular fixed design, is known to engineers as analysis. The process of analysis parallels the structure of the mathematical entity known as a function; inputs are accepted, calculations made, and outputs returned. Analysis is a one-way function, however; one can calculate the objectives given the design variables, but not vice versa. Analysis can only evaluate a candidate design solution, and cannot by itself determine the best design solution.

Optimization is the process of determining the set of design variable choices that will produce the best objective value. Unless stated otherwise, in this thesis the convention

of assuming that objective values are being minimized is followed, that is, "the best" is taken to mean "the smallest". It is in principle possible to find the optimum analytically by finding zeroes of the derivative of the functions, but this is rarely possible in practice as the functions that arise in real engineering problems are far too complex. Thus numerical optimization algorithms must employed, in which successive guesses at the proper choice of design variables are made and various strategies are employed to improve each guess based on the result of evaluating the previous guess. In general, there are four components to an optimization:

- The design variables, which specify what design choices are being made for a particular candidate design
- The objectives, which represent the quality of a particular candidate design
- The objective function, which incorporates the analysis process by which objective values are computed given design variable values
- The optimization algorithm, which determines how to improve the next guess at the best design variable values each time the objective function is used to evaluate a given set of design variables

2.1.2 Pareto Optimality

In order to determine which design produces the best set of objective values, it must be possible to compare two sets of objective values and determine which is better. Following the minimization convention, this is done by following the rule that "for each objective, a smaller value is better than a larger value". For the single-objective case, this suffices to distinguish between sets of objectives, simply because each set has only one member. For example, if one is choosing a hotel based on cost, one can always make a selection successfully by following the rule "cheaper hotels are better than more expensive hotels". This is illustrated in Figure 2.1.

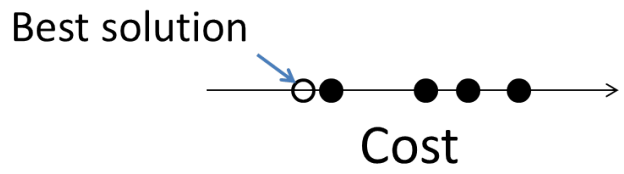


Figure 2.1: The single-objective case

The multi-objective case, however, is indeterminate without more information. If one is choosing a hotel based on price and distance from the airport, there are two selection rules: "cheaper hotels are better than more expensive hotels" and "hotels closer to the airport are better than hotels farther from the airport". Consider two possible comparison scenarios:

1. Hotel A costs $100/night and is 1 mile from the airport; Hotel B costs $150/night and is 2 miles from the airport

2. Hotel A costs $100/night and is 1 mile from the airport; Hotel C costs $75/night and is 2.5 miles from the airport

In scenario 1, the two selection rules can be simultaneously followed by choosing A over B, and it is said that B is *dominated* by A since A is better with respect to all objectives. However, in scenario 2, no best choice can be made that satisfies both selection rules; neither A nor C dominates the other. The objective values in the example are shown in Figure 2.2, with the rectangular regions associated with each solution representing the portion of the objective space in which there must exist at least one other solution in order to dominate that solution.

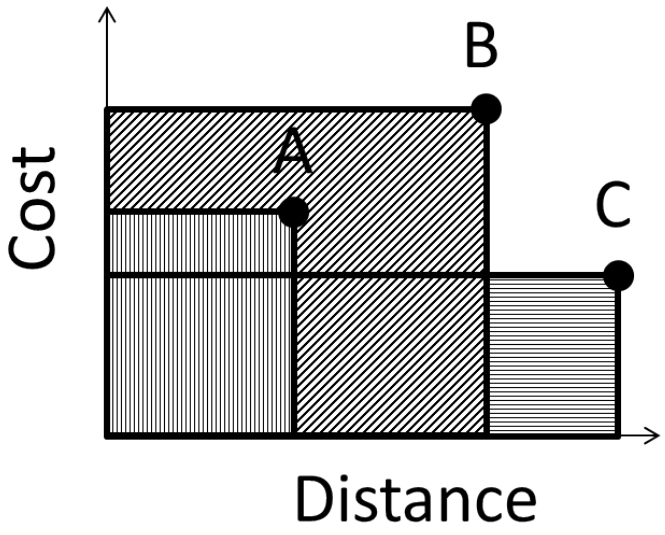


Figure 2.2: Hotel example

If one has a whole list of hotels to choose from, the best one can do using the two selection rules is to narrow down the list to the set of hotels that are *non-dominated*; the final selection must be made using some other criteria. Non-dominated solutions are also referred to as Pareto-optimal, and the set of Pareto-optimal solutions is known as the Pareto front.

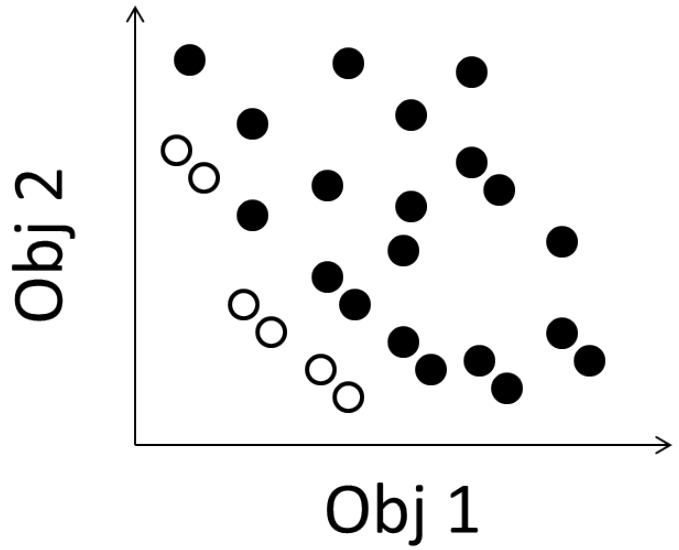


Figure 2.3: The multi-objective case; the unfilled circles are Pareto-optimal

2.1.3 The NSGA-II Algorithm

Computing the location of the Pareto front in practice, given a particular multi-objective optimization problem, is a non-trivial task for which a number of different algorithms have been proposed. One of the most popular, and the one selected for use in the work discussed in this thesis, is the Non-Dominated Sorting Genetic Algorithm (Second Version) or NSGA-II.

Before explaining this algorithm, it is necessary to give some background on how a generic single-objective genetic algorithm works. This approach to optimization is inspired by the effectiveness of biological natural selection at solving the formidable problem of optimizing an organism's physical characteristics for survival in a complex environment. In a genetic algorithm (GA), each candidate solution (i.e. set of design variable values) is treated as an individual "organism". The individual has a genotype and a phenotype; the genotype represents the "genetic" information and is generally implemented as a binary string, which can be uniquely converted into the individual's "phenotype", or actual design

variable values. This process is analagous to the way that DNA contains instructions for building an organism.

The GA considers a population of some number of individuals (around a hundred is a typical number to have). The phenotypes are constructed from the genotypes from each individual in the population, and the objective function is then used to evaluate the quality of each phenotype. Each individual is then assigned a fitness value reflecting how well its phenotype did at accomplishing the objective. Finally, a number of individuals are selected for "breeding", in which pairs of individuals are taken and a new "child" individual is generated by combining genetic information from the two "parents". The individuals with higher fitness values have a better chance of being selected for breeding, so that the new generation will hopefully be fitter than the current one. Additionally, "mutation" is implemented in which some portion of the children have random changes introduced into their genotypes, promoting diversification of the solution pool. This process is repeated using the child population, and a number of successive generations are evaluated until a prescribed limit is reached. The best individual from the last optimization then represents the approximate solution to the optimization problem.

NSGA-II builds off this basic strategy in order to address the more complex task of multi-objective optimization. Evaluation of a generation begins in the usual way with construction of the phenotypes from the genotypes and evaluation of each individual with the objective function. Since the problem has multiple objectives, each individual will have not just a single objective value but a set of values, one for each objective in the problem. Much of the merit of the NSGA-II algorithm lies in its approach to selecting the next generation for the multi-objective case.

First, *every* individual is selected for breeding, and a child population of the same size as the parent population is produced. The two are then concatenated to form a double population. The double population is then sorted using the idea of the non-domination front, which is a simple extension of the idea of the Pareto front. The first non-domination

front F_1 is the Pareto front, the second non-domination front F_2 is the Pareto front of the set of solutions with F_1 removed, etc. Figure 2.4 shows an example of a small population in a two-objective problem where each dot represents the objective values of an individual. The example population has three non-domination fronts.

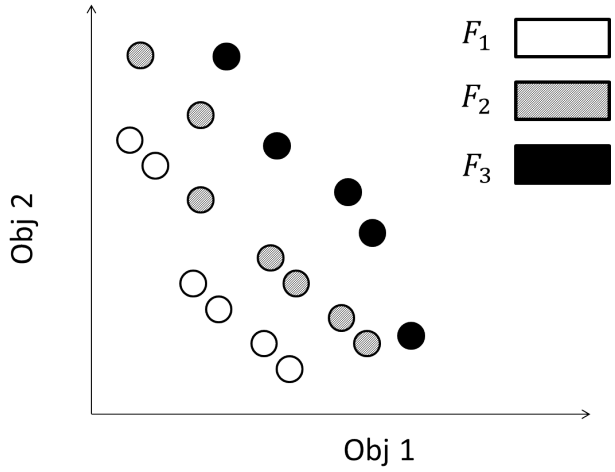


Figure 2.4: Non-domination fronts

Once the non-domination sort is complete, the double population is reduced to a single population using the following procedure. The non-domination fronts are added to the single population in order of quality (i.e. starting at F_1) until the population is "full", that is, until the sum of the number of individuals in the fronts added so far is equal to the required number of individuals in the population. Figure 2.5 shows a schematic of this process.

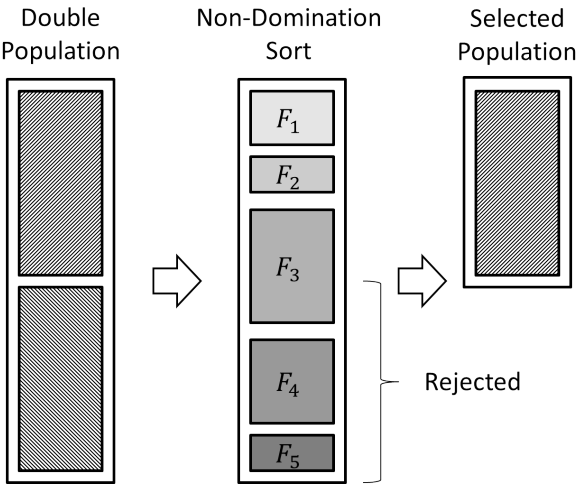


Figure 2.5: Selection of next generation

A complication is that since in general each front has a different (and unpredictable) number of members it is very unlikely that for any n, taking F_1 through F_n will give exactly the required number of individuals, so a "tiebreaker" must be used to add only part of the last front F_n. This is done through the use of a secondary criterion known as "crowding", in which solutions that are further away from their fellows in either the design or objective space are preferred to solutions that are close to another solution, in order to maintain diversity in the gene pool.

A brief qualitative overview of the process has been given here; for more information the reader should consult the original paper that this algorithm was presented in [12], which is remarkable not only for the utility of the algorithm it presents but for the clarity of its explanation. The NSGA-II algorithm can be applied to problems from any scientific discipline, and its application to the optimization of mechanisms is of little theoretical import in and of itself. The next section looks at the structure of a strictly mechanism-based optimization problem, in order to investigate more specific techniques.

2.2 Categorization of Design Variables and Objectives

In considering the optimization of mechanisms, it is necessary to define a distinction between *kinematic* and *non-kinematic* design variables and objectives. Kinematic design variables are defined to be those whose values will have an impact on the kinematics of the mechanism, whereas non-kinematic design variables are only relevant to other aspects of the multi-domain problem and do not affect the kinematics. Similarly, a kinematic objective is one whose value can be found during an objective function evaluation by reference solely to the mechanism kinematics, i.e. is completely determined by specification of the kinematic design variables only, whereas computing a non-kinematic objective requires specification of at least one non-kinematic design variable (and may or may not involve the kinematic design variables). The non-kinematic aspect of the problem can involve structural mechanics, heat transfer, electromagnetics, fluid dynamics, or virtually any other subset of engineering science.

Two simple examples, involving kinematic-structural and kinematic-fluid-mechanical problems respectively, are now given to illustrate the distinction. Imagine in case 1 that a four-bar linkage is being designed to move the coupler link along some path at high velocity, and that it is also important to minimize the coupler mass, so a dynamic finite element procedure is used to determine the stresses from inertial loading and help design a structurally efficient coupler geometry. In this case the kinematic design variables are the link lengths of the linkage, and the non-kinematic design variables are those that determine the shape of the coupler structure, since as long as the joints are in the same place this has no effect on the motion of the mechanism. The kinematic objectives might be a least squares fit to the desired coupler curve and maximization of velocity over a certain portion of the motion, while the non-kinematic objective is minimization of the coupler mass (without structural failure). A simple representation of the mechanism is shown in Figure 2.6, with non-kinematic design variables circled.

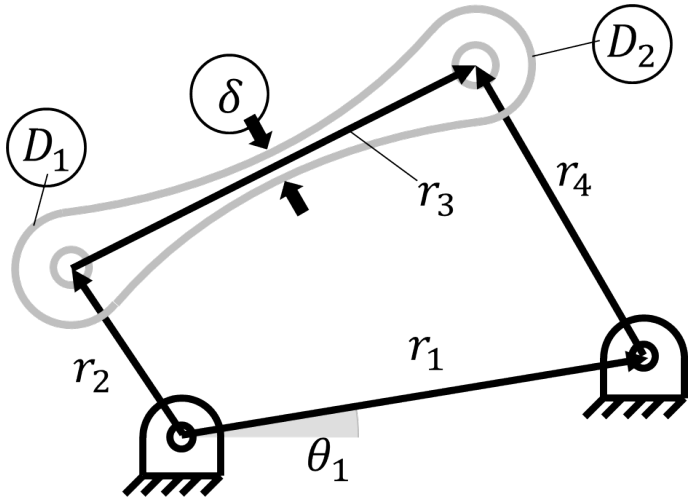


Figure 2.6: Case 1

In case 2, a simple crank-slider mechanism is used as a hydraulic pump, and it is desired to minimize the footprint of the mechanism for packaging reasons while simultaneously maximizing the efficiency of the pumping action. The kinematic design variables are the crank and connecting rod lengths, while the non-kinematic design variables could be the bore to stroke ratio of the piston and the clearance of the seal between the piston and the cylinder wall. If joint size is neglected, the mechanism footprint is determined by the link lengths and is a kinematic objective, whereas the pumping efficiency is a non-kinematic objective. A simple representation of the mechanism is shown in Figure 2.7, with non-kinematic design variables circled.

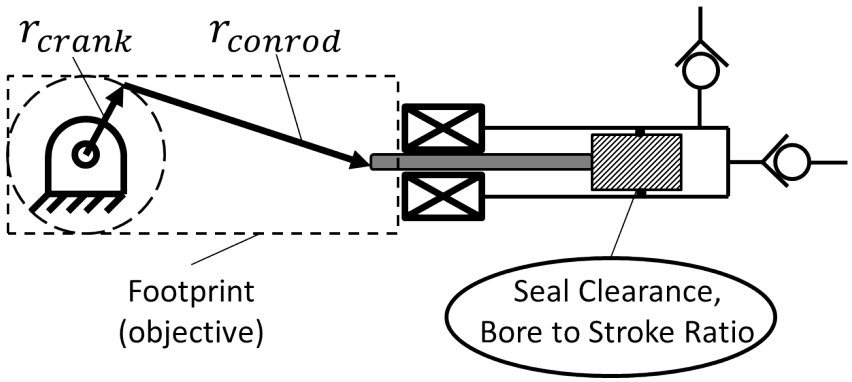


Figure 2.7: Case 2

2.3 Naive Approach

In order to demonstrate the value of using the three strategies for formulating the optimization, it is helpful to begin by looking at what a naive formulation of the optimization would entail and pointing out the problems created. The obvious means of proceeding would be to define one's design variables, with the kinematic ones most likely being the link lengths and maybe an angle or two, and simply plug in the problem to one's preferred multi-objective optimization algorithm, making no algorithmic distinction between kinematic and non-kinematic design variables and objectives. While in some cases this may be enough to solve the problem, there are a number of serious potential drawbacks to the naive approach.

2.3.1 Unworkable Mechanisms

It is well known that for most mechanisms it is possible to specify dimensions that are physically impossible to construct. For example, if in a four-bar linkage one of the links is longer than the sum of the other three the linkage is invalid. Mathematically, this will manifest itself during computation of the objective function as a physically meaningless quantity such a complex-valued angle. A less drastic version of this problem can occur

when a linkage can be constructed but fails to meet certain criteria for proper operation, the two most notable examples of which are the Grashof criterion and minimum transmission angle constraints. Certainly in the case of complete failure, and potentially in the case of Grashof or transmission angle failure depending on the specifics of the optimization, the objective function cannot compute any meaningful values and the only recourse is to abort the objective function evaluation and return "no solution" instead of a set of objective values to the optimization algorithm. For carelessly written code, this can result in a terminal error, or worse in a non-terminal error that gives false results. Even for code with proper inclusion of exception handling, this creates two problems, the first being simply that whatever computation time was needed to get to the point that caused the error has now been wasted. The second problem comes from the impact on the progress of the optimization of a failed objective function call. If the optimization algorithm is carelessly constructed, this may derail it entirely, although usually it is possible to make most algorithms robust enough to handle this type of return fairly easily. However, even in the best case where both the objective function and optimization algorithm are properly prepared, the objective function call has still failed to yield any useful information. Even a bad solution is valuable for steering the optimization in a different direction, but no solution contributes nothing. If the proportion of calls that fail in this way is high enough, it may be difficult or impossible for the optimization to converge correctly. Blindly trying different combinations of link lengths over a large range is very likely to lead to this difficulty.

The use of constraints between the design variables to define a feasible region is one way to address this issue. While this is preferable to blind combination, it is still unsatisfactory for two reasons. First, depending on the complexity of the mechanism and the choice of design variables it may prove difficult or impossible to formulate constraints that exclude all unworkable candidate mechanisms while still including all valid candidates. Second, even if this is done successfully, some objective function calls will still land outside the feasible region. The consequences will be alleviated by the use of constraints, but

it is still the case that an optimization that has trouble staying inside the feasible region will likely have more difficulty converging quickly than one that does not. The use of indirect design variable specification to address this issue will be discussed shortly.

2.3.2 Choosing Design Space Bounds

In general, it is desirable to set bounds on the allowable value of each design variable that are as restrictive as possible in order to minimize the required scope of an optimization. However, in practice this is often difficult to do without running the risk of accidently excluding the unknown locations of optimal designs. The quality of an optimization formulation can be greatly improved if the design variables are chosen such that they have natural bounds, where they cannot meaningfully take on values outside of a certain finite (and hopefully small) range. This simultaneously guarantees inclusion of all solutions in the design space and limitation of the size of that space. If natural bounds cannot be found, it is up to the designer to choose either one or both of the upper and lower bounds on the basis of incomplete information or often pure intuition, which to avoid the danger of exclusion of optima must generally be done very conservatively, resulting in an overly large design space.

To look at a specific instance, link lengths are a common choice of kinematic design variable, and have the advantage of having a natural lower bound of zero as lengths cannot be negative. However, there is generally no way to find a natural upper bound, and the designer's intuition of at what point the length becomes unreasonable is the only substitute. The aforementioned need for conservative choice of bounds in poorly understood situations, combined with the gradual transition from reasonable to unreasonable length, can result in large design spaces for link length design variables.

2.3.3 Computational Inefficiency and Instability

Other types of difficulties can arise from the failure to distinguish between kinematic and non-kinematic quantities. Consider the case where two sets of design variables are evaluated where all the kinematic design variables are the same and differences are only present in the non-kinematic design variables. Whatever kinematic calculations are performed by the objective function will be duplicated exactly since non-kinematic design variables by definition have no effect on the kinematics of the mechanism. For many problems, the kinematic analysis performed by the objective function will consist of evaluation of link positions, and possibly velocities and/or accelerations as well, at a number of discrete positions in order to approximate the continuous values of these quantities over the motion of the mechanism. When one considers that the evaluation at each position usually requires use of several computationally expensive trigonometric and inverse trigonometric functions, and that a large number of positions may be needed to achieve acceptable accuracy of approximation, it is evident that a large amount of computation time is being wasted by repeating these calculations. Storing the results of the first instance of each kinematic computation for later lookup may be a solution, but is likely to be very memory-intensive if there are a large number of different candidate linkages (which is usually the case), and if a large number of positions are evaluated, resulting in significant memory demands for each linkage. There will also of course still be some processing time penalty associated with indexing and read/write operations on large amounts of data.

A related but more subtle problem is potential instability in the relationship of the non-kinematic objectives to different combinations of kinematic and non-kinematic variable sets. To take a simple example, in case 2 (see Figure 2.7) there are contributions to pump inefficiency both from leakage flow past the piston seal doing lost pressure-volume work, and from sliding friction in the seal itself. Clearly changing the seal clearance will improve one of these at the other's expense, and for each particular mechanism an optimum seal

clearance can be expected that minimizes the combined losses. However, one would expect this to be dependent on the velocity of the piston, as higher velocity would increase the relative importance of sliding friction. Since velocity is of course dependent on the kinematics of the linkage, it can be seen that even in this simple problem there is an inherent cross-coupling between the current value of the kinematic design variables and the optimal value of the non-kinematic design variables for each candidate mechanism. That is, a favorable seal clearance value for one candidate linkage could be very unfavorable for another with significant kinematic differences, so that the optimization algorithm becomes "confused" about what seal clearance value it should converge on as the current best case fluctuates unpredictably as new candidate linkages are generated. It may be difficult for the algorithm to converge quickly and accurately in complex instances of this type of interdependence.

2.3.4 Unavailablity of Discipline-Specific Optimization Techniques

In many cases the designer will have access to techniques for dealing with *part* of the problem in a much more efficient way than a general-purpose multi-objective optimization. This could be a specific technique available in the literature, a commercial or difficult to adapt software, or just the knowledge of a different optimization algorithm that would be effective for a subset of the design variables pertaining to a particular discipline but that cannot be applied to the full set. Examples might include a traditional analytical synthesis method on the kinematics side that fails to take into account non-kinematic objectives, or perhaps a structural optimization technique that cannot be applied due to the type of cross-coupling discussed in the previous paragraph. Blanket application of a general-purpose optimization to the entire problem eliminates the option of applying these sorts of specific techniques, whereas as will be seen partitioning the optimization into multiple nested levels allows their application in many cases.

2.4 Reformulated Approach

Having discussed the difficulties that can result from the naive approach to the optimization problem, three methods of reformulating the problem in order to overcome them are now suggested.

2.4.1 Indirect Design Variable Formulation

Section 2.3.1 discussed the problems that can arise from choosing design variables that result in impossible or unacceptable linkages, and the limited effectiveness of constraint application as a remedy. Additionally, in section 2.3.2, the desirability of choosing design variables with natural bounds, as well as the difficulty of doing this for link lengths, was pointed out. In certain cases these two problems can be solved at a single stroke by the use of an adaptive interpolation to define one or more design variables, which kinematic design variables are particularly well-suited for. Consider for example the choice of design variables for a four-bar that must be a Grashof crank-rocker. In the naive approach, the four link lengths and the ground link angle are blindly co-specified, and a check is then performed that an acceptable linkage can be constructed. This approach is illustrated in Figure 2.8.

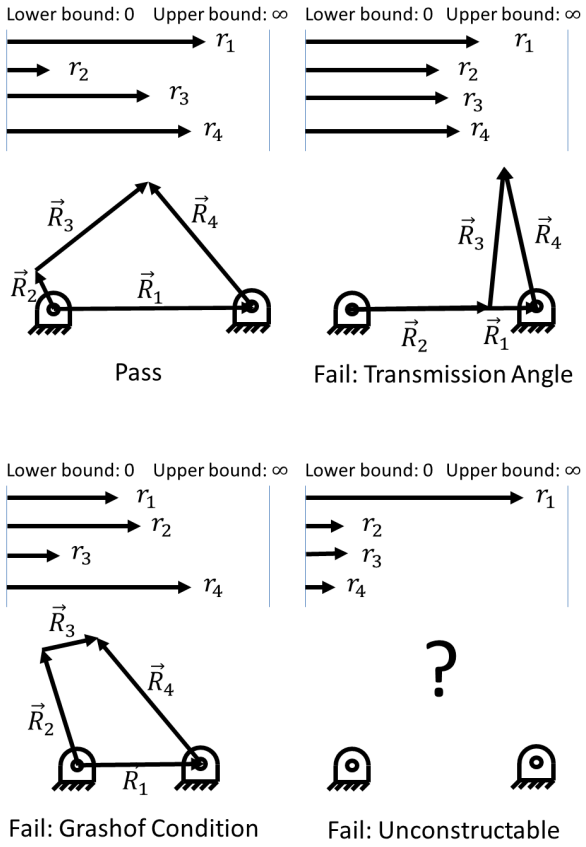


Figure 2.8: Naive design variable selection outcomes for Grashof crank-rocker

A better approach is as follows. First, normalize the linkage by r_2, that is, set r_2 to length 1 and specify that all additional lengths will be given in ratio to r_2 rather than as absolute quantities. Next, specify normalized values for r_3 and r_4 greater than 1. At this point upper and lower bounds on r_1 can be determined by a simple application of the law of cosines. Since it is important to consider the transmission angles in the mechanism in order to achieve acceptable quality of motion [74], a minimum acceptable transmission angle θ_{min} is defined.

$$
\begin{aligned}
r_{1,min} &= \sqrt{r_3^2 + r_4^2 - 2r_3 r_4\ cos(\theta_{min})} + r_2 \\
r_{1,max} &= \sqrt{r_3^2 + r_4^2 - 2r_3 r_4\ cos(\pi - \theta_{min})} - r_2
\end{aligned}
\tag{2.1}
$$

This calculation is also illustrated in Figure 2.9. Note that this determination is only valid for this particular candidate linkage.

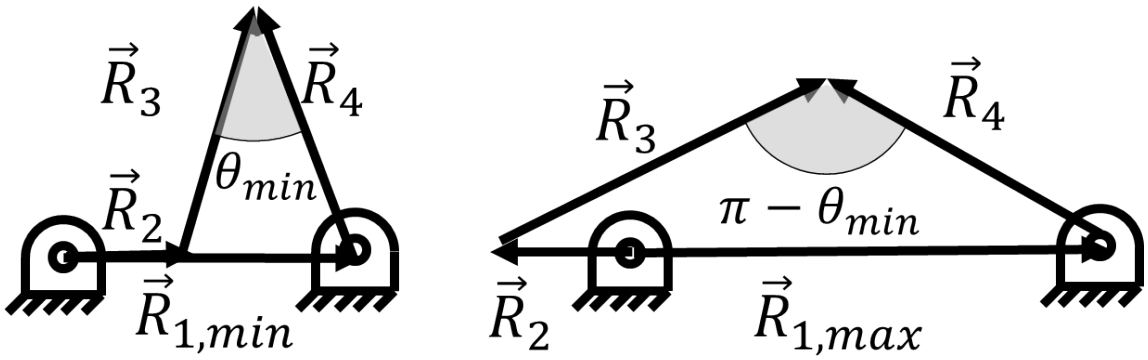


Figure 2.9: Determination of candidate-specific bounds on r_1

Instead of an explicit value of r_1, which may or may not have fallen inside this range of validity, the objective function was supplied with an index of interpolation $t_1 \in [0, 1]$ for the link 1 design variable. Now that the bounds on r_1 have been determined, r_1 is recovered from t_1 by a simple linear interpolation:

$$
r_1 = (1 - t_1)\ r_{1,min} + (t_1)\ r_{1,max} \tag{2.2}
$$

When using this technique, it is important to be aware of the fact that for $\theta_{min} > 0$ there exist r_3 and r_4 values that will result in $r_{1,min} > r_{1,max}$, meaning that the interpolation will produce non-Grashof linkages. Figure 2.10 shows the contours at which $r_{1,min} = r_{1,max}$ for different values of θ_{min}, meaning that points below these contours will produce non-Grashof linkages and must be avoided. This can be done simply by having the lower bounds

on r_3 and r_4 be greater than r_2.

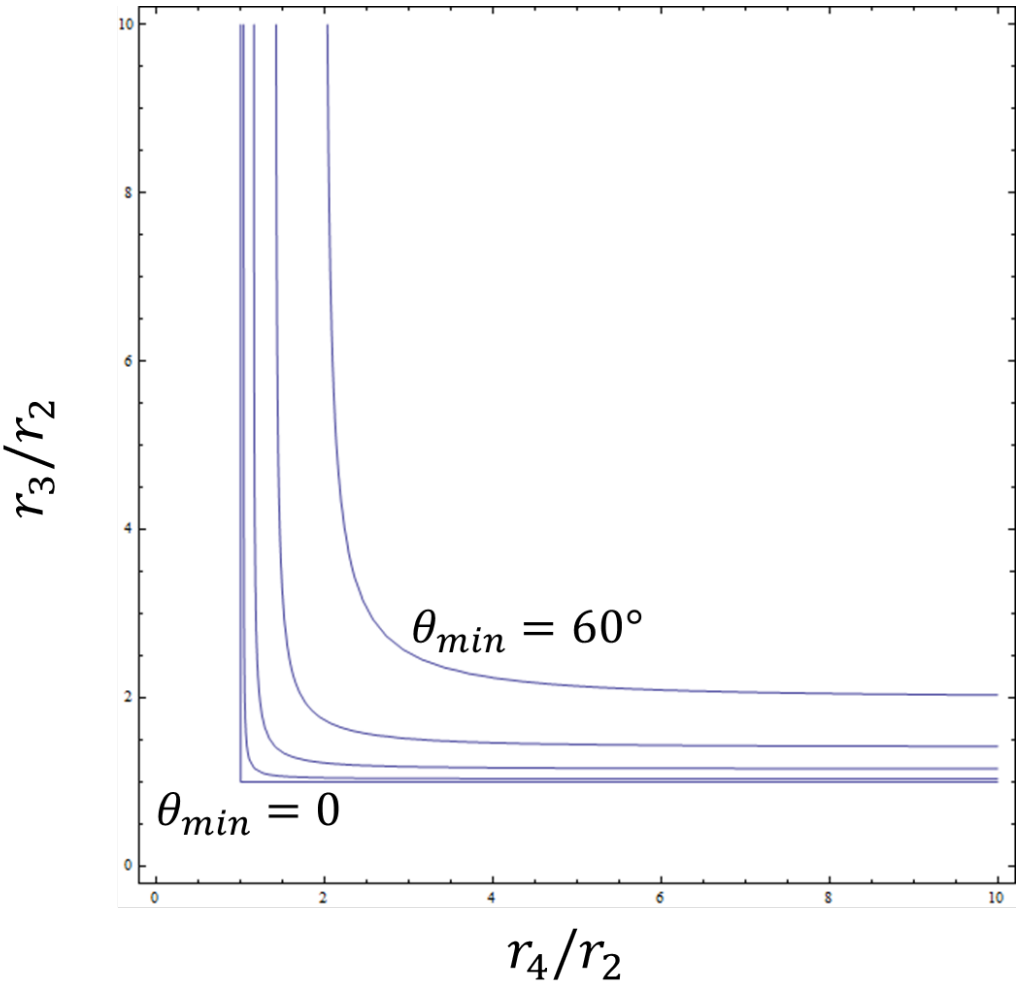


Figure 2.10: Guaranteed-Grashof contours for $\theta_{min} = \{0, 15°, 30°, 45°, 60°\}$

Finally, the linkage must be unnormalized in order to obtain real lengths rather than ratios to r_2. There are two possible ways to do this. The preferred method is to compare the linkage to some fixed length scale. If for example the chord of the arc traversed by the $\vec{R}_3 - \vec{R}_4$ joint is required to be a specific length by the problem description, there is only one scaling of the normalized linkage that will match correctly, and there is no need to introduce another design variable. If no length scale comparison is available, a final design variable must be introduced that scales the linkage by some factor. Note that a design variable was eliminated by normalization in the first place, so in the first case the net effect is a reduction

by one of the dimension of the optimization and in the second case the normalization and final scaling offset each other in this respect.

What has been gained by this more complicated procedure? If a length scale comparison was available, the dimensionality of the optimization was reduced by one, which usually results in a substantial decrease in the difficulty of the optimization. Even if this was not the case, the reformulation of the design variables has resulted in the feasible region becoming the entire design space; that is, any possible combination of design variable values in the specified ranges will result in a valid linkage, without even the need to apply constraints. There is no longer any need to even check for the failure modes of Figure 2.8. Furthermore, the use of r_1, which had only a lower bound, was replaced with the use of t_1, which has both an upper and lower bound, achieving the goal of a naturally closed interval for this variable.

The substitution of t_1 for r_1 is known as as an adaptive interpolation, and as was demonstrated, can be used to guarantee that the value of the design variable is always valid, even if no constraint can be found for the general case, by computation of candidate-specific bounds during every objective function call and interpolation between them. In addition to guaranteeing the validity of the design variable value, no constraints need be added to the optimization, and the index of interpolation is naturally bounded on the interval $[0, 1]$. This technique can be applied in a variety of situations but requires the designer's judgment to determine when it is appropriate. The Grashof crank-rocker case is used in the IC engine case study, and a different application of adaptive interpolation is used on angular design variables in the rescue spreader case study.

2.4.2 Reduction of Design Space by Kinematic Analysis

As was discussed in the literature review, powerful direct synthesis techniques exist for path, motion, and function generation problems to match a desired kinematic behavior. If the mechanism is subject to *prescriptions* of its motion (in constrast to optimizing some

aspect of the motion) that fit into one of these categories, it may be possible to reduce the dimension of the optimization by one or more by applying the kinematic synthesis technique and choosing the design variables accordingly before beginning the optimization per se. For example, suppose one is conducting a multi-domain optimization of a four-bar mechanism where the coupler point is required to pass through four precision positions. In this case the optimization design space reduces from the full six dimensions needed to describe an unconstrained four-bar to the two dimensions corresponding to a choice of points on the Burmester curves for each dyad, which are generated by solving the corresponding synthesis problem [75]. In traditional applications of Burmester theory, having one or more infinities of solutions is almost an inconvenience rather than an asset, and the choice between multiple solutions that satisfy the synthesis requirements is generally made by vague intuitive methods such as convenient location of ground pivots. When used as the design space for an optimization involving non-kinematic criteria this flexibility becomes much more useful.

While it rare to have an optimization that is subject to four prescribed positions, similar if less drastic dimensional reductions apply to more likely cases involving fewer prescribed positions for path, motion, and function generation problems. The design variables will have to be properly chosen so as to parameterize the reduced space, which may not be as straightforward as simply taking physical linkage dimensions as the design variables, but the reduction in dimension should be well worth the added trouble. For example, the rescue spreader optimization achieves an important dimensional reduction by restricting the design space to solutions to a two-position motion generation problem.

2.4.3 Nested Optimization Structure

In the naive approach, once the design variables and objectives were chosen the entire problem was plugged into a general-purpose multi-objective optimization algorithm. Such general-purpose algorithms are often quite powerful and their generality makes them

convenient to use in an "off-the-shelf" sense, and for many relatively simple optimization problems this may be all that is required. However, many multi-diciplinary optimization problems can be very computationally expensive, and the designer may be in need of a more efficient method. On general principles one would expect that more powerful techniques become possible as the problem becomes more specific, since they can be tailored to take advantage of specific knowledge of the nature of the problem.

According to this reasoning, it would be wise to take advantage of the guarantee of a kinematic aspect existing, which allows formulation of a more specific method than in the completely general case. This can be accomplished by the partitioning of the optimization into two nested levels. The primary or outer level is purely kinematic; the outer objective function takes only the kinematic design variables as inputs, and thus only the kinematic objectives can be immediately computed. The non-kinematic aspect of the problem comes into play through the use of a secondary or inner optimization. Evaluation of each primary objective function call entails solution of the secondary optimization for that specific candidate mechanism. The details of how the solution to the secondary optimization affects the primary optimization depend on whether the secondary is a single or multi-objective problem; that is, whether there is one or more than one non-kinematic objectives. It is easiest to use the simpler case of a single-objective inner optimization for purposes of explanation. To make use of a concrete example, consider again the simple case 2 optimization, shown again for ease of reference in Figure 2.11. The design variables and objectives are given in Table 2.1.

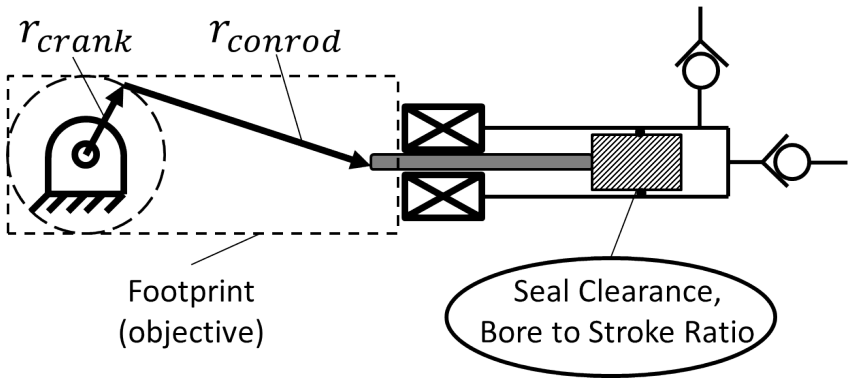


Figure 2.11: Case 2 example problem

Table 2.1: Design variables and objectives for case 2 example problem

	Design Variables	Objectives
Kinematic	r_{crank}, r_{conrod}	Minimize footprint
Non-kinematic	Bore-to-stroke ratio, seal clearance	Maximize pumping efficiency

A qualitative description of the nested optimization procedure for this problem is as follows. The outer optimization algorithm generates a call to the outer objective function, passing in the appropriate kinematic design variable values, i.e. the lengths of the crank and connecting rod for the current candidate design. The outer objective function immediately computes the kinematics of the mechanism, which will be the same regardless of what values of bore-to-stroke ratio and seal clearance are chosen, and without further ado determines the value of the kinematic objective, i.e. the footprint area. Now the inner optimization is solved. A number of calls are made to the inner objective function, specifying a different value for the bore-to-stroke ratio and seal clearance each time. Eventually the inner optimization algorithm is able to converge on an answer, and determine what values of these non-kinematic design variables correspond to the highest efficiency *for this particular candidate mechanism*, as well as what that efficiency is. The outer objective function now has access to the value of all kinematic and non-kinematic objectives, and returns.

The outer level optimization considers this information and either generates another objective function call with improved design variable values or terminates if the optimality or maximum search time conditions are satisfied. This procedure is also shown in Figure 2.12.

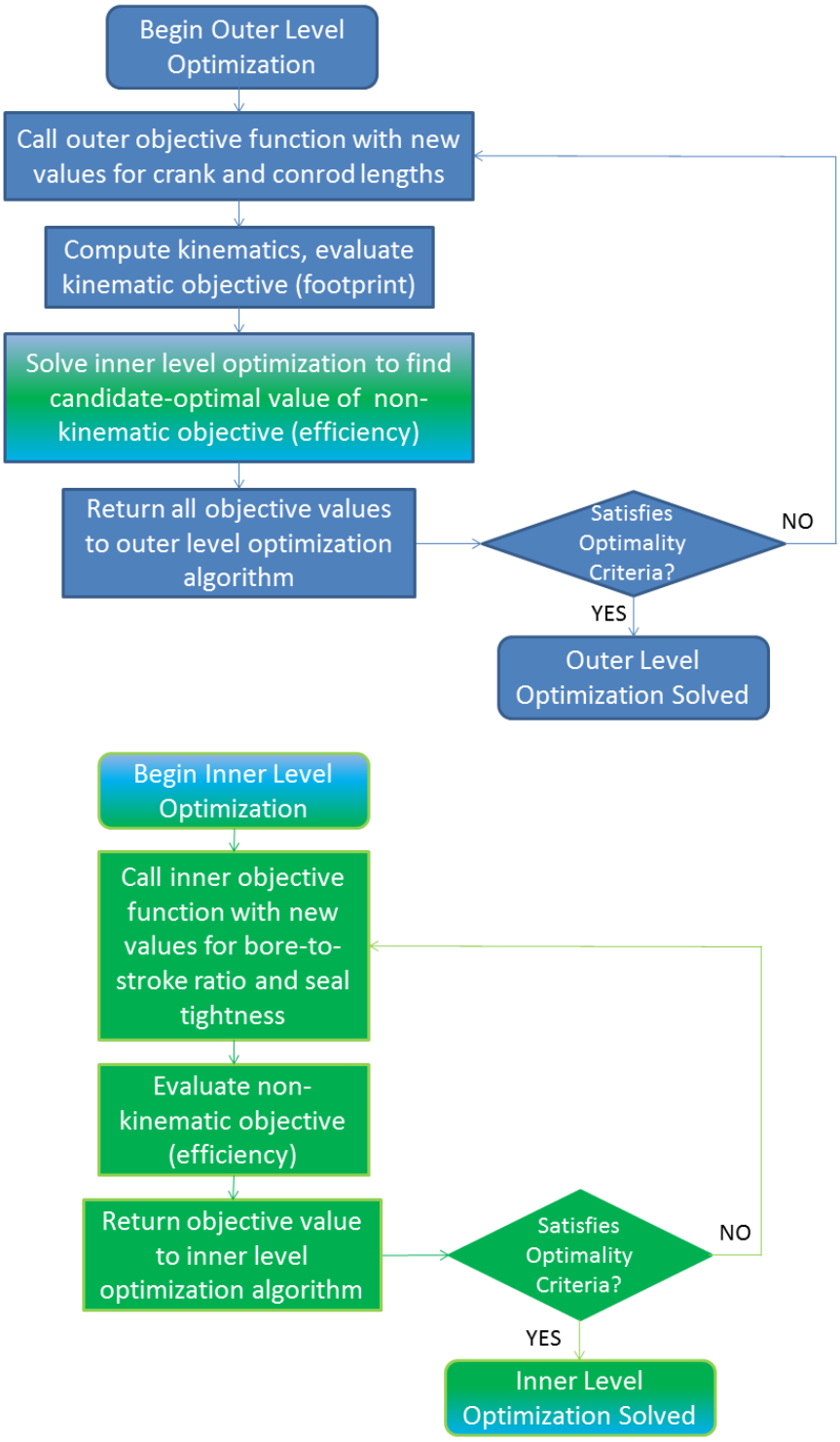


Figure 2.12: Case 2 nested optimization

An important aspect of the nested structure is that it effectively makes the non-kinematic objectives a function of only the kinematic design variables at the outer level. In this example, for each candidate mechanism (that is, choice of crank and connecting rod lengths), the inner optimization finds the best value that the non-kinematic objective can assume, by determing what choice of bore-to-stroke ration and seal clearance is best for the current candidate mechanism. In other words, when the outer level optimization algorithm asks the outer objective function "what if we try these crank and connecting rod lengths?", the objective function answers "then the footprint will be this, and the efficiency will be this assuming the best design is chosen subject to those parameters". This means that the design space now effectively consists of only the kinematic design variables.

Of course, the dimension of the problem has not really been reduced, only partitioned, so one might wonder whether this approach is indeed an improvement. If there are N kinematic design variables and M non-kinematic ones, the question then is whether it is better to solve an M-dimensional optimization at every objective function call of an N-dimensional optimization, or an N+M dimensional optimization once? One might be tempted to say the latter seems more efficient. Consider, however, that there are a number of qualitative factors in favor of the nested approach.

Perhaps the most readily apparent advantage is that the inefficient recomputation of the same kinematics during different objective function calls discussed in Section 2.3.3 is no longer an issue. Kinematic computations are done only once at the beginning of each outer level objective function call, and the entire inner optimization process uses the data from this one-time computation. The other problem discussed in that section was that attempting to optimize the entire multi-domain system at once could easily lead to instability and convergence difficulties due to complex interdependence of the objectives on different kinematic and non-kinematic design variable combinations. The simple example of this effect present in the hydraulic pump problem is that the optimal seal clearance to balance flow leakage and sliding friction losses is dependent on the piston velocity and hence mechanism

kinematics. The nested approach provides a divide-and-conquer capability with respect to the multi-domain nature of the problem that eliminates this issue. For any given inner optimization, the kinematic design variables have a single fixed value, so all that is left is a straightforward single-disciplinary optimization. At the outer level, only the kinematic design variables are present, each combination of which is assigned a single fixed value of the non-kinematic objectives by solution of the inner optimization.

Another advantage of the nested approach is its increased modularity and adaptability. Since the partition of the design variables between the inner and outer levels is done on a kinematic vs. non-kinematic basis, the difficulty of applying preferred discipline-specific optimization techniques to the full multi-domain problem, as discussed in Section 2.3.4, is largely overcome. In particular, if the non-kinematic portion of the problem is particularly involved, it may be of great help to apply a known optimization technique from the relevant literature. All the inner optimization has to do to interface with the rest of the procedure is accept any relevant data from the kinematic analysis done at the beginning of the outer objective function call and return the optimal non-kinematic objective values. This makes the procedure as a whole quite modular, as almost any optimization technique can be plugged in as the inner level with a minimum of modification. Even if a known technique does not exist for the particular problem, it may still be advantageous to use a different general-purpose algorithm at the inner and outer levels. For example, in the rescue spreader problem, the NSGA-II algorithm is used at the outer level, but the inner optimization is solved by Matlab's fmincon gradient-descent approach.

2.4.4 Nested Generalization of the NSGA-II Algorithm

One complication that can arise from the use of the nested approach occurs when there are multiple objectives at each level. The NSGA-II algorithm explained above requires each objective function evaluation of a candidate solution (or an individual using genetic terminology) to be single-valued with respect to each objective. That is, each individual

must correspond to a point in the objective space (Figure 2.13), in order to determine the Pareto front and other non-domination fronts and execute the algorithm as described earlier in this chapter.

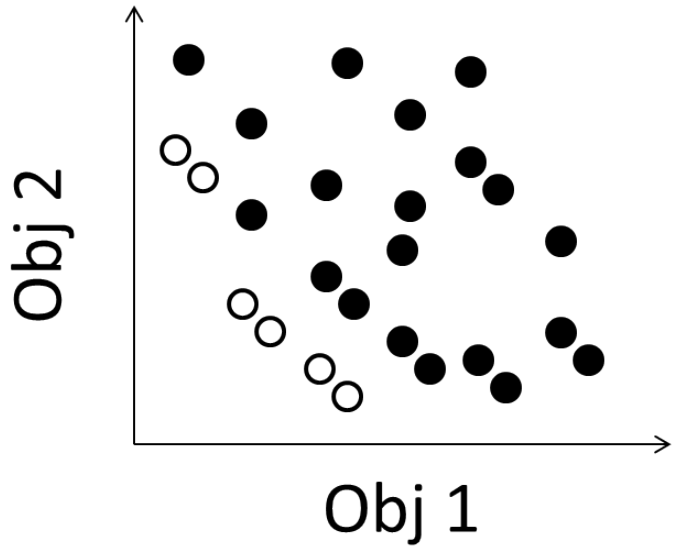


Figure 2.13: Standard case: individuals are single-valued with respect to each objective

However, if NSGA-II is used at the inner level, each individual at the outer level will correspond not to a point but to an entire Pareto set returned by the inner level optimization, as shown in Figure 2.14. Here the different shapes each correspond to a different candidate at the outer level. For example, if a kinematic-structural optimization were being conducted, the squares would represent a single set of linkage dimensions, with each square being a different structural implementation.

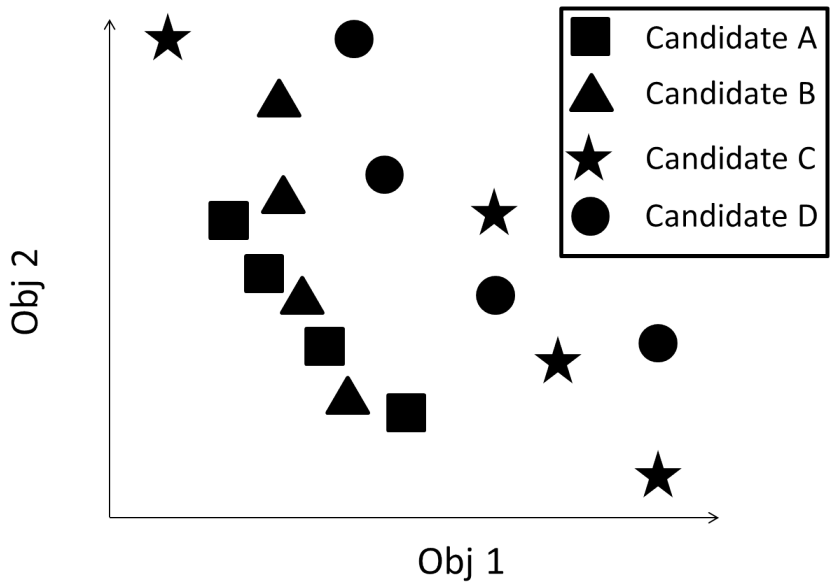


Figure 2.14: Nested case: individuals are *not* single-valued with respect to each objective

This situation complicates the task of deciding which outer-level individuals to select for the next generation. The whole point of proceeding on a Pareto-oriented basis in a multi-objective optimization is to avoid ruling out solutions based on arbitrary or imperfect weighting of the relative importance of the various objectives. On the other hand, one must choose something to discard in order to proceed to the next generation.

Examination of the literature did not yield any guidance on the conduct of a nested multi-objective optimization, so a generalization of the NSGA-II algorithm to deal with this situation was developed. The sub-solutions are sorted into non-domination fronts as usual, as shown in Figure 2.15.

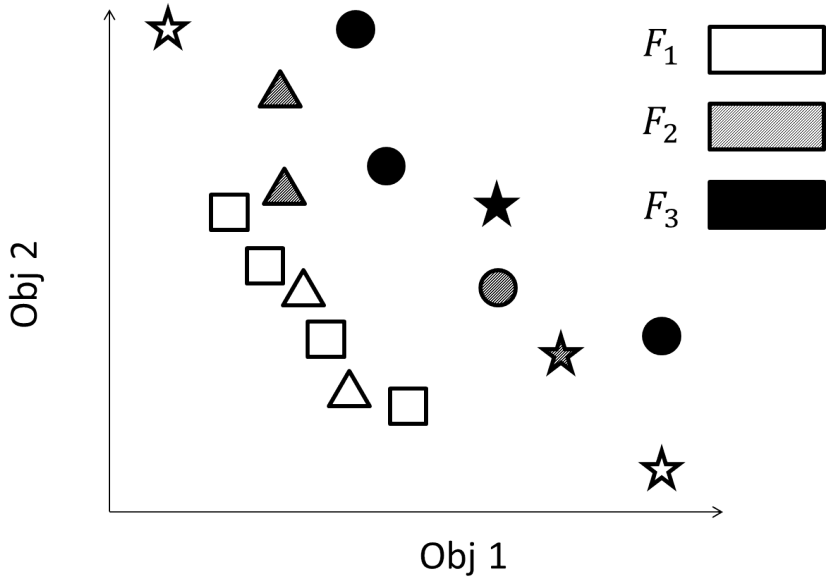


Figure 2.15: Non-domination sort in nested case

Once this is accomplished, all outer level candidates (or "shapes" in this example) with at least one representative in F_1 are selected for retention in the next generation. This is then repeated for F_2, F_3, etc. until enough outer level candidates have been selected to complete the generation.

The need for a good tiebreaker is much greater than in the standard algorithm; since it is harder for solutions to dominate each other than in the single-valued case, one expects to have to resolve larger sets of tied solution in the last non-domination front to (partially) make it into the next generation. In the example shown in Figure 2.15, the double population has four members, so only two can be kept for the next generation. Looking first at F_1, squares, triangles, and stars are all represented. Since the required number of candidates is already more than met just from F_1, F_2 and F_3 are not examined, and thus the circle solution is rejected. There is now a tiebreaker requirement; only two of the three in F_1 can be selected. This is done by preferring solutions with representatives in the current non-domination front with better uninterrupted coverage of the objective space in order to maintain diversity. Thus the star solution is rejected by the tiebreaker, as it only covers the

ends of the space and has a big gap in the middle.

This strategy allows the use of the standard NSGA-II algorithm (or any other Pareto-based algorithm) at the inner level, while capturing the essence of its behavior at the outer level as well.

Implementation Details

As this nested generalization was developed by the author, a more precise description of the novel aspects is needed. Due to the complexity of the required data management, an object-oriented approach was used. Each outer-level objective function call on an individual returned a "super-solution" class object as defined by the kinematic design variable set. Each super-solution object contained a number of "sub-solution" objects corresponding to different non-kinematic design variable sets. In terms of the symbols of Figure 2.14, each shape type is a super-solution and each instance of that shape is a sub-solution. Each sub-solution contained the following information:

- Kinematic design variable values (same for all sub-solutions in a given super-solution)
- Non-kinematic design variable values (different for each sub-solution)
- "Parent Number" keeping track of which super-solution it belonged to
- Objective values for both kinematic and non-kinematic objectives

Using this object-oriented framework, the nested NSGA-II executed as follows during each generation:

1. A primary objective function call was generated for each individual at the outer level.
2. During each of these calls, a super-solution object was created and assigned the appropriate kinematic design variable values.
3. The kinematic objectives were evaluated and recorded.

4. The inner level optimization was executed for each individual using the standard NSGA-II, resulting in a Pareto-optimal set.

5. Each member of this set generated a sub-solution object, which was assigned the appropriate non-kinematic design variable and objective values, the appropriate parent number, and attached to its parent super-solution object.

6. Once all the primary objective function calls were complete, all sub-solutions from all super-solutions were concatenated into a large sub-solution population.

7. The usual non-domination sort was performed on this sub-solution population (as illustrated in Figure 2.15). After each non-domination front was identified, the parent numbers of the member sub-solutions were queried and a list of the super-solutions represented in the front was assembled. This continued for successive fronts until the required number of super-solutions (i.e. half of them) was obtained, usually necessitating a tiebreaker when the last front had too many members.

8. The super-solutions that won the tiebreaker, as well as those from superior non-domination classes, then constituted the next generation.

The tiebreaker procedure was as follows.

NOBJ = number of objectives in the problem

find maximum and minimum value of each objective in the last non-domination class

for each super-solution *i* in the last non-domination class:

NSUB = number of sub-solutions associated with this super-solution

solnMat = NSUB x NOBJ matrix of sub-solution objective values

maxGapSum = 0

for each objective *j*

ordObj = column j of solnMat sorted in ascending order

add min and max value of objective j to ends of ordObj

differences = vector of differences between consecutive elements of ordObj

maxGap = max(differences)/max value of objective j

maxGapSum = maxGapSum + maxGap

end for

record maxGapSum for this super-solution

end for

the super-solutions with the smallest values of maxGapSum win the tiebreaker

This procedure is illustrated graphically in Figure 2.16. The tied non-domination front from Figure 2.15 is shown, along with its projections onto the Obj 1 and Obj 2 axes. The Obj 1 and Obj 2maxGap values for (in ascending order of size) squares, triangles, and stars are shown. It can be seen that stars have the biggest maxGap value for both objectives, and therefore the biggest value of maxGapSum. Thus squares and triangles win the tiebreaker and stars are rejected, since there are only two spots available in the next generation. The idea was to prefer solutions that had the most comprehensive coverage of the objective space to discourage clustering and maintain diversity.

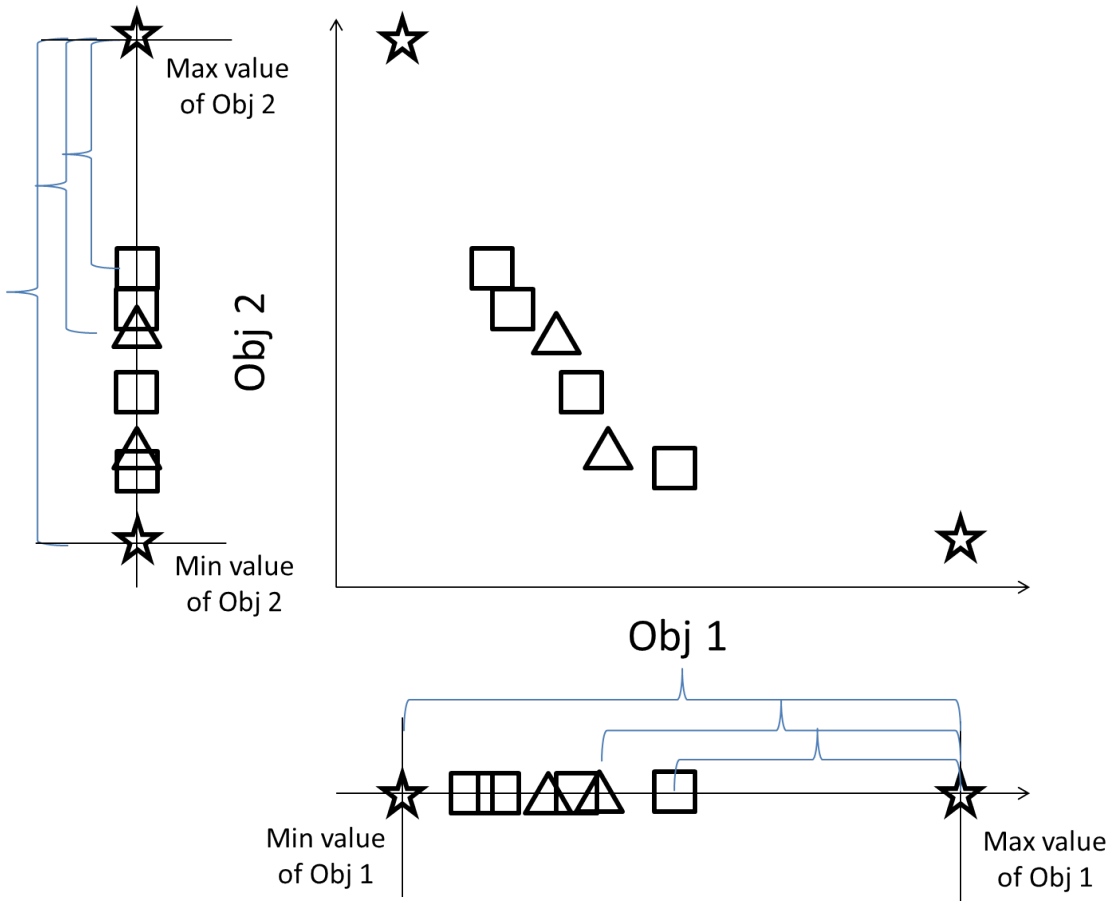


Figure 2.16: Tie-breaker strategy in nested version of NSGA-II

2.5 Summary

In this chapter, a number of potential pitfalls associated with an unsophisticated approach to a multi-domain optimization of a system involving a mechanism were discussed, and a three-part strategy for a better approach was proposed. The naive approach is susceptible to failed objective function calls from invalid mechanisms, unbounded design variables, computational inefficiency and poor convergence, and lack of opportunity to apply existing optimization techniques for parts of the system. These problems were addressed by indirect design variable formulation using normalization and adaptive interpolation, reduction of design space by preliminary kinematic analysis, and the use of a nested optimization struc-

ture. The nested structure required a generalization of the NSGA-II algorithm to handle the case where both levels are multi-objective. With the nature, advantages, and motivation for the general approach understood, each of the next two chapters details a case study where this approach is applied to a specific problem.

Chapter 3:

First Case Study: Hydraulic Rescue Spreader

The first case study involves the optimization of a hydraulic rescue spreader. Hydraulic rescue spreaders, also known by the brand name “jaws of life”, are hydraulically actuated mechanisms used in emergency situations to remove victims trapped inside wreckage, often as a result of automobile accidents. The most common design comprises a Watt-II six-bar linkage that converts the motion of a hydraulic actuator to the spreading action of a pair of jaws. A schematic of a typical rescue spreader is shown in Figure 3.1.

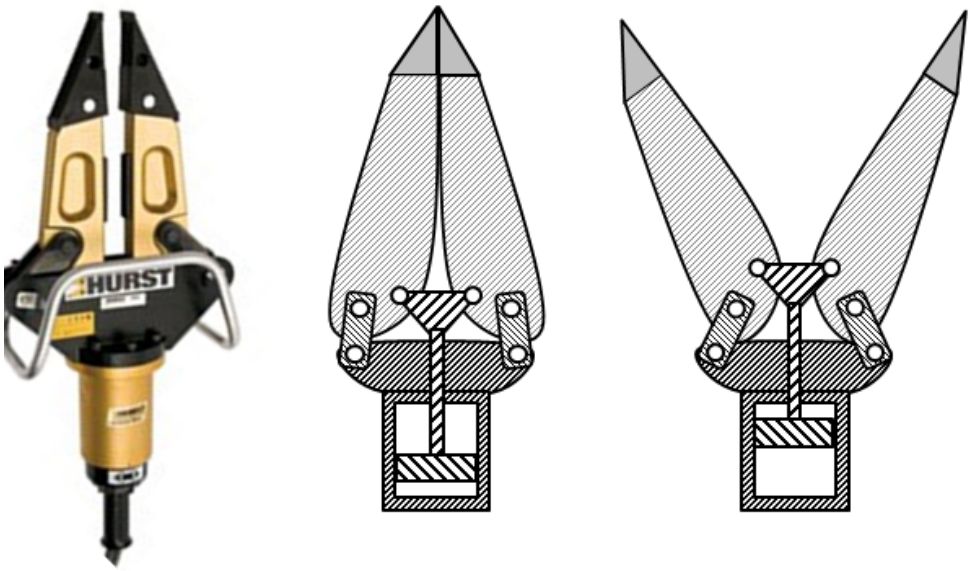


Figure 3.1: A typical rescue spreader and motion schematic [2]

Generally the jaws must exert a large amount of force to deform various metal structures, resulting in large loads throughout the linkage. As the rescue spreader is an emergency tool that must be used with relative speed and ease by a single operator, care must be taken to design the mechanism efficiently to minimize weight, which as can be seen in Table 3.1 and Figure 3.2 can be up to 60 lbs depending on the required spreading force and spreading distance.

Table 3.1: Spreading force, spreading distance, and mechanism mass data for selected commercially available rescue spreaders [76] [77] [78] [79]

Model	Force (lbf)	Distance (in)	Weight (lbm)
Hurst			
SP-300	7419	24	37.7
SP-310	8767	28	44.3
SP-510	12274	31.5	55.1
SP-512	17310	24	58.2
ML-28	8317	28.3	49
KL-32	6969	32.4	50.5
Holmatro			
4240	8476	27.375	44.5
4242	8800	27.25	42
4260	13085	32.75	58
Genesis			
S 35	7785	24	35
S 49 XL	11250	28	45
S 55 XL	9743	32.3	45.6
S 60 XL	14175	31.9	54.9

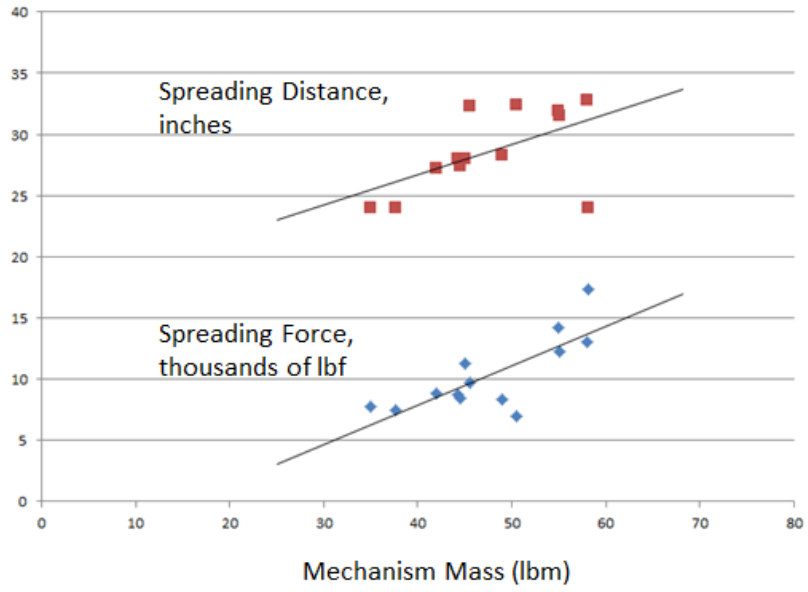


Figure 3.2: The data of Table 3.1 in graphical form

3.1 Design Considerations and Objectives

In addition to the need to minimize weight, there are a number of important mechanism design considerations. Note that the inner pivots on the jaws can only translate in a direction parallel to the symmetry line of the mechanism, meaning that the spreading distance acheived by the tips is solely due to rotation of the jaws. Clearly a jaw that is long and has high rotation between the open and closed positions will have a greater spreading distance than one that is short and has low rotation. However, having long jaws, in addition to creating a greater bending moment due to the load at the tips, can produce mechanisms that are a meter long or more and are too difficult to handle. Large amounts of jaw rotation mean the faces of the jaw tips become too far from perpendicular to the direction of spread, risking loss of grip on the target material. Thus a delicate balance exists between jaw rotation, jaw length, and available spreading distance. Finally, many existing designs suffer from poor spreading force in the closed position, due to high variation in mechanical advantage from

sub-optimal kinematic design. An ideal spreader would have perfectly uniform spreading force at all points in its motion.

It can be seen, then, that the design of such a device is not a trivial task. It is in fact a perfect candidate for a multi-domain optimization, using both kinematics and structural mechanics. To begin the formulation of the optimization, the following objectives are defined:

Kinematic Objectives

- Minimize mechanism length
- Minimize jaw rotation
- Minimize variation in spreading force

Non-kinematic Objectives

- Minimize mechanism mass

Additionally, the mechanism is required to achieve a spreading distance of 24 inches, and to have a spreading force in the closed position of 10,000 lbf, both of which are based on the performance of existing spreaders. Finally, transmission angles are required to be greater than 30 degrees to ensure correct function of the mechanism.

While there are potentially many different linkage topologies that might be suitable given these criteria, it was decided to restrict the optimization to the usual Watt-II six-bar design. A detailed topological analysis of possible rescue spreader configurations is presented in earlier work by the author [53], which shows that for a symmetrical mechanism the six-bar is the simplest possible solution. As the general method presented in this thesis assumes that topology selection has been completed by other methods prior to its application, no attempt has been made to incorporate this stage of the design process.

3.2 Preliminary Kinematic Analysis

With the objectives and required performance defined and an assumed topology given, in accordance with the strategy outlined in Chapter 2 a preliminary kinematic analysis is now performed. The most obvious aspect of the kinematics is the half-symmetry of the mechanism, which means that only the four-bar half-linkage shown in Figure 3.3 need be considered.

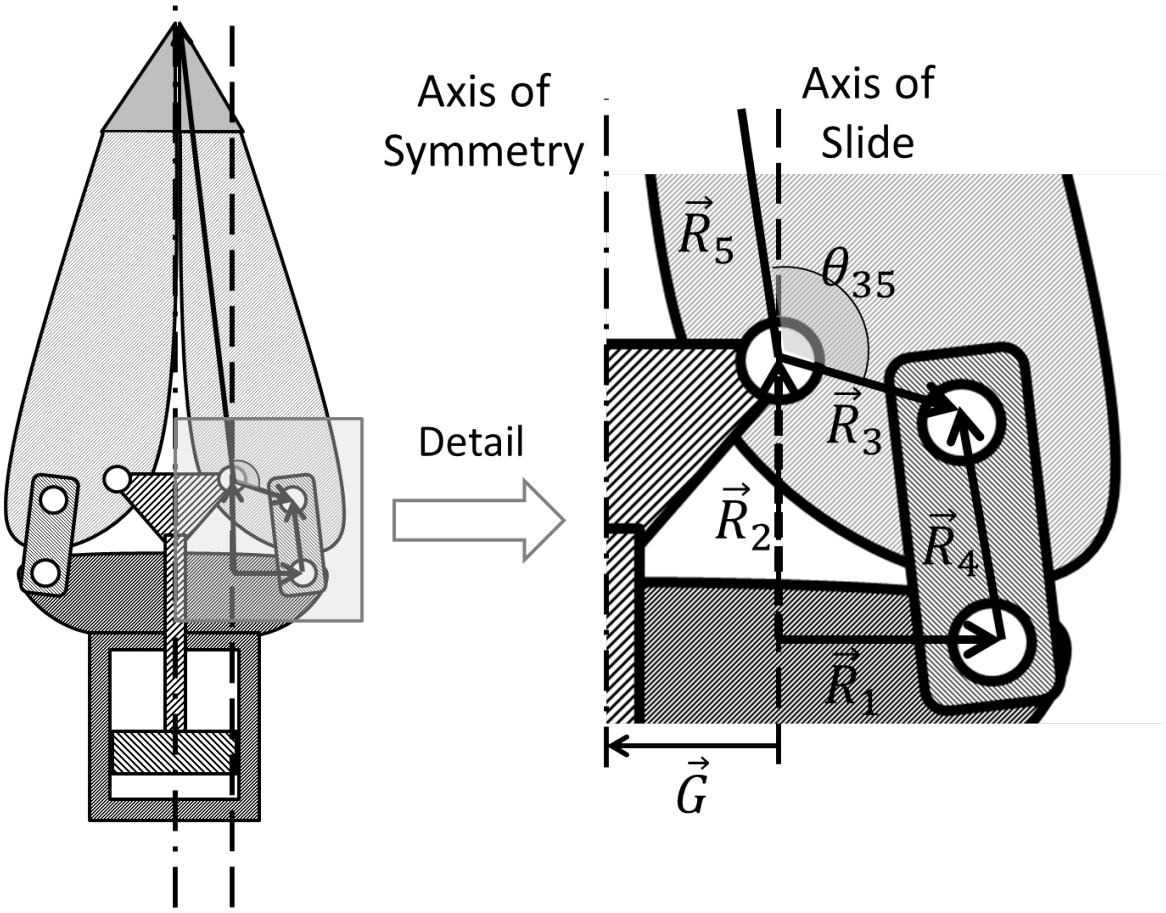


Figure 3.3: Half-linkage

Note that $\vec{R}_1$ is defined to be horizontal, and $\vec{R}_2$ is defined to be vertical and is of variable length. Also note that in the detail of Figure 3.3 $\vec{R}_5$ is only partially shown, and terminates at the jaw tip out of the frame.

At first glance, there appear to be eight kinematic design variables: $\{g, r_1, r_{2,i}, r_{2,f}, r_3, r_4, r_5, \theta_{35}\}$, where the subscripts i and f correspond to "initial" and "final". However, by restricting allowable linkages to the solutions to a two-position motion generation problem, this is reduced to five.

3.2.1 Formulation of Motion Generation Problem

The linkage geometry is shown in Figure 3.4 in its initial and final positions. Note that the revolute joint connecting the jaw to the hydraulic piston, which is confined to vertical motion, has been conceptually replaced by a slider on a vertical axis of slide.

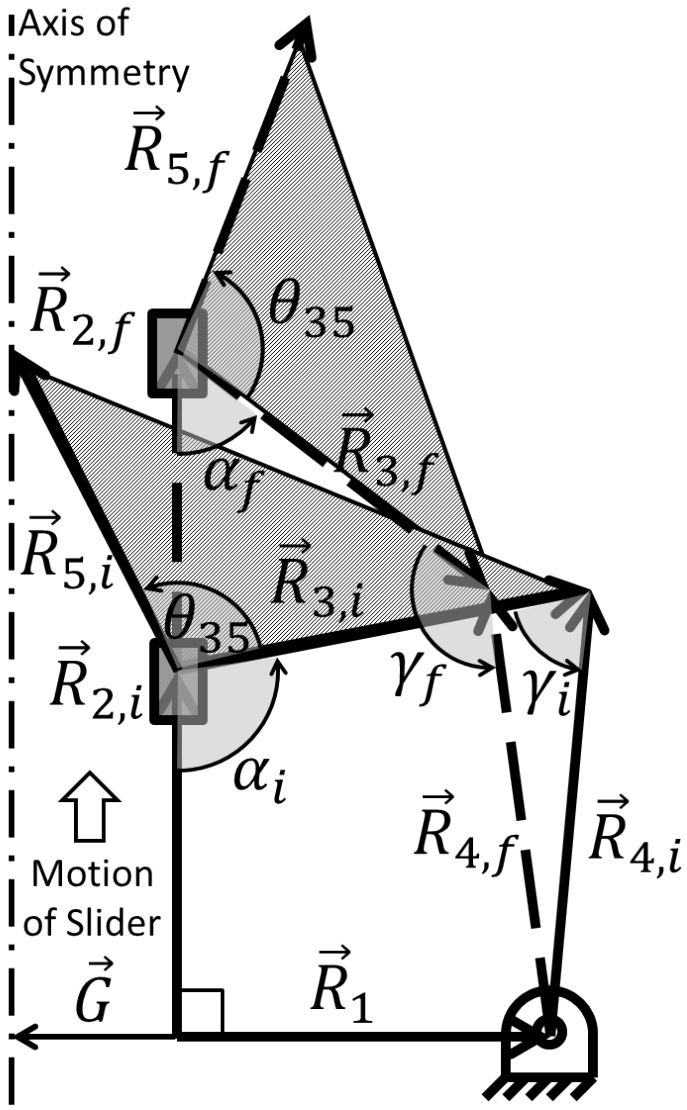


Figure 3.4: Motion Generation

As this is a complicated diagram, a verbal description may prove helpful. The four-bar linkage is shown in the initial (solid) and final (dashed) positions and is defined by the vectors $\vec{R_1}$, $\vec{R_2}$, $\vec{R_3}$, and $\vec{R_4}$, with $\vec{R_5}$ defining the location of the coupler point (jaw tip). The jaw is shaded for clarity. $\vec{R_1}$ is fixed and horizontal, $\vec{R_2}$ is vertical but increases in length from the initial to final positions to represent the extension of the hydraulic actuator. θ_{35} defines the fixed angle between $\vec{R_3}$ and $\vec{R_5}$. α is the interior angle from $\vec{R_2}$ to $\vec{R_3}$, and

γ is the interior angle from $\vec{R}_3$ to $\vec{R}_4$. Additionally, define the x direction as parallel to $\vec{R}_1$ and the y direction as parallel to $\vec{R}_2$, with the origin at their intersection. Finally, $\vec{G}$ is the horizontal vector describing the distance from the axis of slide to the axis of symmetry of the mechanism.

There are two requirements for the motion shown in Figure 3.4. It was mentioned previously that the spreading distance is required to be a prescribed value (24 inches). This means that the x-displacement of the tip of $\vec{R}_5$ from the initial to final positions must be half that (12 inches). Additionally, the amount of jaw rotation is prescribed. While the reader will recall that minimizing jaw rotation was set as one of the objectives, it proved most convenient to also make it a design variable, where arbitrary minimization is of course precluded by the deleterious effect this would have on the other objectives. Based on jaw tip gripping considerations, a possible range of 30-60 degrees was chosen for jaw rotation.

The situation can be recognized as being essentially a motion generation problem, where a line on the coupler link must undergo a prescribed rotation and the point on the end of the coupler must travel a prescribed distance in the x direction, although the fact that only the x distance is prescribed is a departure from the usual practice. Note that the prescribed half-spread distance is defined with the jaw tip starting on the axis of symmetry, so that the two tips are in contact in the closed position of the mechanism.

There are twelve variables involved: $\{g, r_1, r_{2,i}, r_{2,f}, r_3, r_4, r_5, \theta_{35}, \alpha_i, \alpha_f, \gamma_i, \gamma_f\}$. The vector loop closure equation on $\vec{R}_1$, $\vec{R}_2$, $\vec{R}_3$, and $\vec{R}_4$ in the initial and final positions yields four scalar equations. The requirements that the jaw tip start on centerline and end a prescribed distance away in the x direction each yield another scalar equation. Finally, there is the prescribed jaw rotation, for a total of seven equations, bringing the number of independent variables down to five. Note that this is three fewer than the eight that initially appeared necessary in the previous section. The choice of which of the twelve variables to take as the independent (design) variables must be considered carefully, as it will have a significant impact on the difficulty of the optimization.

It is important to recall at this point the reasoning of Section 2.3.2, in which the desirability of having bounded design variables was discussed. Given the minimum transmission angle requirement of 30 degrees, or $\frac{\pi}{6}$, γ_i, and γ_f must be in the range $[\frac{\pi}{6}, \frac{5\pi}{6}]$. As will be seen shortly, solution rectification considerations restrict α_i and α_f to the same range. It therefore makes sense to take these angles, which are naturally fully bounded, as design variables. Due to the constraint $\theta_{rot} = \alpha_i - \alpha_f$ (where θ_{rot} is the prescribed jaw rotation) exists, both α_i and α_f cannot be taken as design variables, but taking only α_i, γ_i, and γ_f still leaves just two more choices. The lengths g and r_1 are chosen, since as will be seen later their sum determines the size of the hydraulic actuator, so a fairly strong natural constraint exists on their maximum sizes. Thus the five design variables are chosen to be $\{g, r_1, \alpha_i, \gamma_i, \gamma_f\}$.

3.2.2 Solution Rectification

To verify the wisdom of using the interior angles α and γ as design variables, the implications of this choice for solution rectification must be investigated.

Constructability of Linkage

As was just stated, it proves necessary to impose the restriction

$$\frac{\pi}{6} < \alpha < \frac{5\pi}{6} \tag{3.1}$$

on the interior angle between R_2 and R_3 and

$$\frac{\pi}{6} < \gamma < \frac{5\pi}{6} \tag{3.2}$$

on the interior angle between R_3 and R_4. Also, the condition $\theta_{rot} = \alpha_i - \alpha_f$ in conjunction with inequality 3.1 implies

$$\frac{\pi}{6} + \theta_{rot} < \alpha_i < \frac{5\pi}{6} \tag{3.3}$$

and

$$\frac{\pi}{6} < \alpha_f < \frac{5\pi}{6} - \theta_{rot} \tag{3.4}$$

These conditions are simply describing the range of values that the two interior angles can assume. It must now be determined if, given these possible ranges, it can ever be the case that the specified values of these interior angles will result in unsolvable linkages. This could happen in one of two ways. The first failure mode can be envisioned by imagining that α is very large, resulting in R_3 pointing upwards at a relatively steep angle. If γ is also large, it is then possible that R_4 will be pointing above the horizontal, meaning that no matter what length it has it will never be able to reach the ground pivot below it. This case is shown at the top of Figure 3.5.

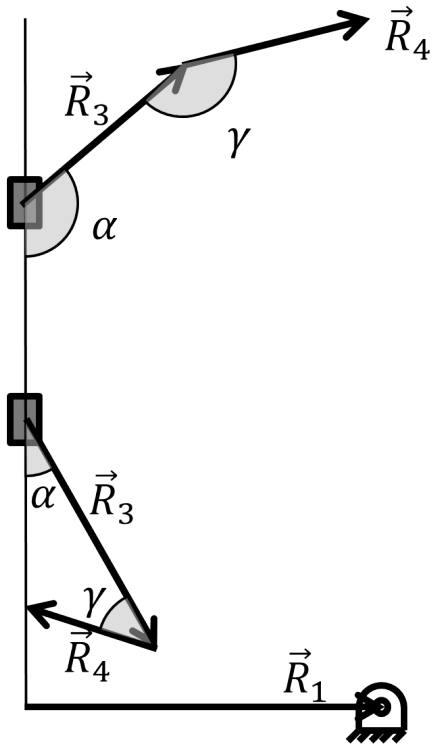


Figure 3.5: Unsolvable angle combinations

Conversely, if α is very small, R_3 will point down at a steep angle, and if γ is also very small $R-4$ will point above the horizontal in the other direction, creating the same problem. This is shown at the bottom of Figure 3.5. In both these cases, the inability of R_4 to reach the ground pivot means that the vector loop closure equation for the four-bar cannot be solved given the specified values of α and γ, and no linkage can be constructed.

To see if these failure modes can occur given the restrictions on the values of the two angles, the first step is to realize that these modes are precluded by satisfaction of

$$\alpha_i + \gamma_i > \frac{\pi}{2} \tag{3.5}$$

which ensures that the sum of the angles is too big for the bottom case to occur, and

$$\alpha_f + \gamma_f < \frac{3\pi}{2} \tag{3.6}$$

which ensures that the sum of the angles is too small for the top case to occur. Inequality 3.5 is evaluated first. Employing proof by contradiction, it is initially assumed that a violation has occured, resulting in the condition

$$\alpha_i + \gamma_i \leq \frac{\pi}{2} \tag{3.7}$$

Substituting in the relations $\frac{\pi}{6}+\theta_{rot} < \alpha_i$ from inequality 3.3 and $\frac{\pi}{6} < \gamma$ from inequality 3.2,

$$\frac{\pi}{6} + \theta_{rot} + \frac{\pi}{6} < \frac{\pi}{2} \tag{3.8}$$

which since $\theta_{rot} \geq \frac{\pi}{6}$ results in the contradiction $\frac{3\pi}{6} < \frac{\pi}{2}$, and it is seen that this pathological case cannot occur. It should be noted that any reduction in the lower bound on θ_{rot} will invalidate this result, as the pathological case has been escaped by only an infinitesmal amount.

Following the same procedure for inequality 3.6, a violation is again initially assumed, resulting in the condition

$$\alpha_f + \gamma_f \geq \frac{3\pi}{2} \tag{3.9}$$

Substituting in the relations $\alpha_f < \frac{5\pi}{6} - \theta_{rot}$ from inequality 3.4 and $\gamma < \frac{5\pi}{6}$ from inequality 3.2,

$$\frac{5\pi}{6} - \theta_{rot} + \frac{5\pi}{6} > \frac{3\pi}{2} \tag{3.10}$$

which since $\theta_{rot} \geq \frac{\pi}{6}$ results in the contradiction $\frac{9\pi}{6} > \frac{3\pi}{2}$, and so this pathological case cannot occur either, although again with an infinitesmal safety margin. The fact that in both cases the failure modes are avoided by an infinitesmal margin justifies the imposition of inequality 3.1, which was until now stated arbitrarily and not justified. If this range were

expanded at all, one or both failure modes would become possible.

Correct Direction of Slider Travel

Although this issue was resolved successfully, another consideration remains. Reference to Figure 3.4 indicates that in order for the direction of slider motion to be upwards and the coupler to rotate CW as desired, it must be the case that

$$\gamma_f > \gamma_i \tag{3.11}$$

in order to lengthen rather than shorten the hypotenuse of the right triangle formed by $\vec{R_1}$ and $\vec{R_2}$. If the only constraint on γ is inequality 3.2, then the only constraints on γ_i and γ_f will be

$$\frac{\pi}{6} < \gamma_i < \frac{5\pi}{6} \tag{3.12}$$

and

$$\frac{\pi}{6} < \gamma_f < \frac{5\pi}{6} \tag{3.13}$$

and satisfaction of inequality 3.11 is not guaranteed. In other words, the only restriction on γ_i and γ_f is that they be in the same range, which does nothing to ensure that $\gamma_f > \gamma_i$ as required for the motion to be correct. The range of possible values of γ_i and γ_f using this method is shown in Figure 3.6.

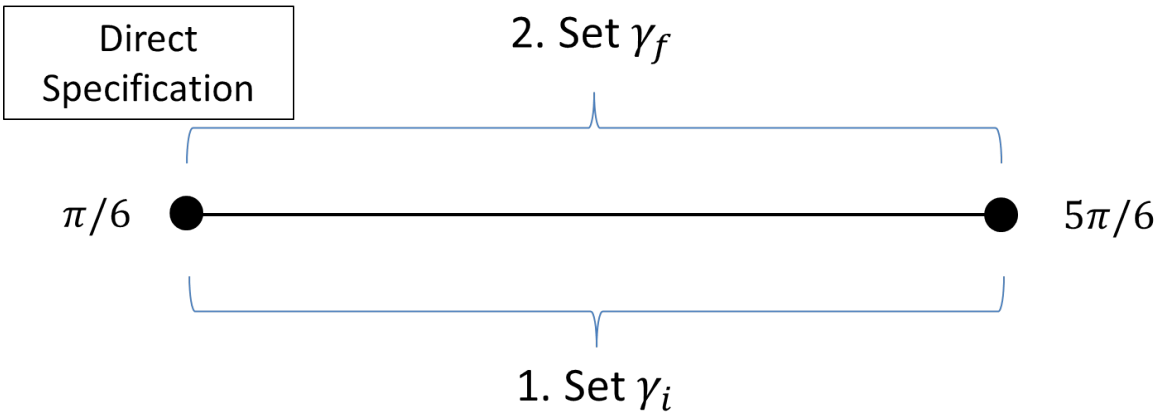


Figure 3.6: Determination of γ by direct specification

This difficulty can be solved by first choosing γ_i in the full range of inequality 3.12 and then requiring

$$\gamma_i < \gamma_f < \frac{5\pi}{6} \tag{3.14}$$

In other words, the initial value is picked anywhere in the allowable range, and the final value is then picked in the portion of the range higher than the choice of the initial value. This is a perfect situation in which to apply the technique of adaptive interpolation discussed in Section 2.4.1. Instead of directly specifying the free variable γ_f, an interpolating parameter $\Gamma_f \in (0, 1)$ is supplied that is then used to find γ_f by linear interpolation as in equation 3.15.

$$\gamma_f = (1 - \Gamma_f)\gamma_i + (\Gamma_f)\frac{5\pi}{6} \tag{3.15}$$

This process is diagrammed in Figure 3.7.

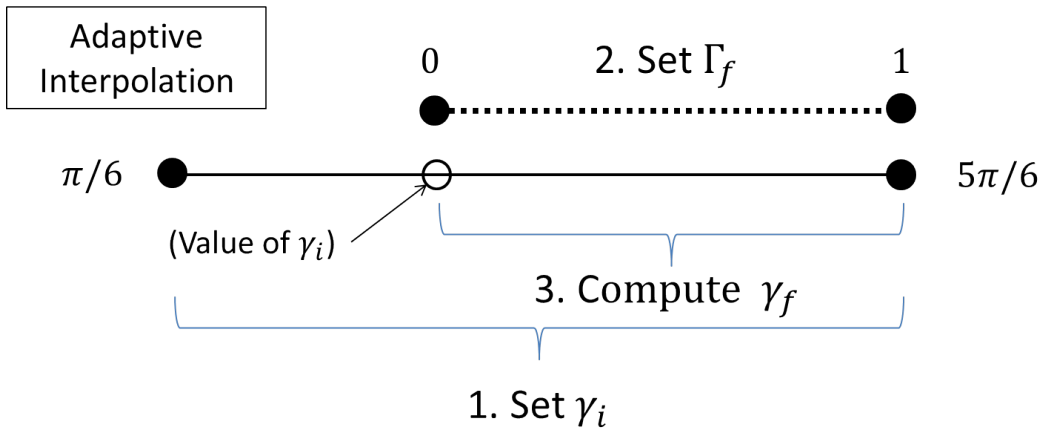


Figure 3.7: Determination of γ by adaptive interpolation

Note that it is now the case that, as recommended by Chapter 2, any design variable combination will result in a constructable linkage. As will be seen, it is unfortunately the case that a few pathological cases are possible where the linkage is constructable but invalid, but since these are few in number they can be handled adequately by *post hoc* exception handling in the optimization.

3.2.3 Solution of Motion Generation Problem

With the motion generation problem formulated, the free variables chosen, and solution rectification addressed, the next step is to explain the procedure by which the remaining seven dependent variables that describe the linkage can be solved for upon specification of the five design variables during an objective function call. The linkage geometry is shown again in Figure 3.8 for the reader's reference. Note that all angles have been specified except for θ_{35} (and technically α_f is not specified per se but rather determined by α_i based on the prescribed jaw rotation), but all lengths except for those of $\vec{R}_1$ and $\vec{\Omega}$ are unknown.

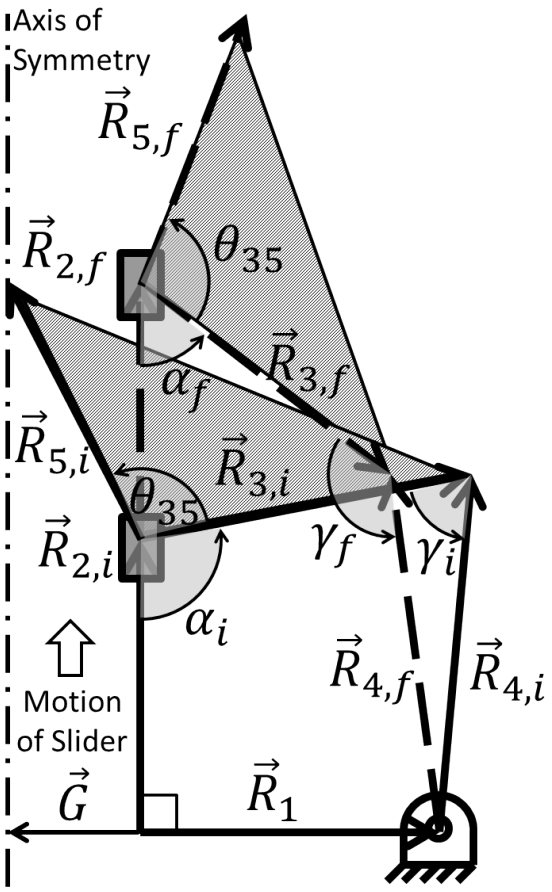


Figure 3.8: Linkage Geometry

Although use of interior angles as the design variables has proven convenient in achieving the goal of guaranteeing a solution for all possible permutations of the design variables in the specified range, for purposes of solution it makes more sense to convert back to the more commonly used form where angles are defined relative to the +x axis. This is done using the following transformations

$$\theta_{3,i} = \alpha_i - \frac{\pi}{2} \tag{3.16}$$

$$\theta_{3,f} = \alpha_f - \frac{\pi}{2} \tag{3.17}$$

$$\theta_{4,i} = \alpha_i + \gamma_i - \frac{\pi}{2} \tag{3.18}$$

$$\theta_{4,f} = \alpha_f + \gamma_f - \frac{\pi}{2} \tag{3.19}$$

which together with considering lengths instead of complex vectors results in Figure 3.9

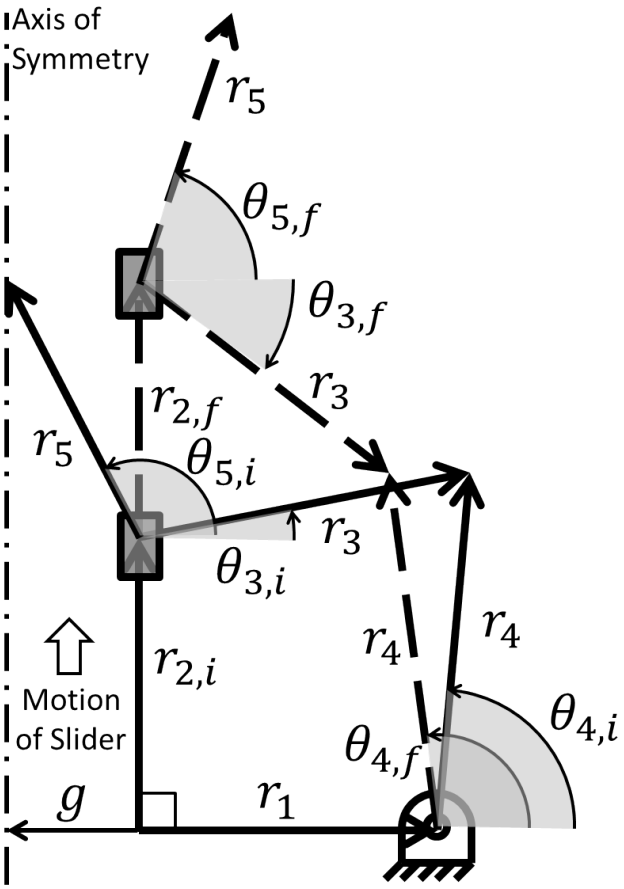


Figure 3.9: Linkage geometry after transformation of angles

There are seven lengths and six angles after this variable conversion. The six quantities $\{g, r_1, \theta_{3,i}, \theta_{3,f}, \theta_{4,i}, \theta_{4,f}\}$ are known. The seven unknowns $\{r_{2,i}, r_{2,f}, r_3, r_4, r_5, \theta_{5,i}, \theta_{5,f}\}$, must be solved for. They need not all be solved for simultaneously, as the system decouples

into two parts. The four link lengths $\{r_{2,i}, r_{2,f}, r_3, r_4\}$ can be found first by solving the system of four equations that results from forming complex vector loops in the initial and final positions. The vector loop equations are

$$r_{2,i}e^{j\frac{\pi}{2}} + r_3e^{j\theta_{3,i}} - r_4e^{j\theta_{4,i}} - r_1e^{j0} = 0 \tag{3.20}$$

and

$$r_{2,f}e^{j\frac{\pi}{2}} + r_3e^{j\theta_{3,f}} - r_4e^{j\theta_{4,f}} - r_1e^{j0} = 0 \tag{3.21}$$

where j is the imaginary unit to avoid confusion with i for initial.

Separating into real and imaginary parts (and dividing out the imaginary unit in the latter),

$$r_3cos(\theta_{3,i}) - r_4cos(\theta_{4,i}) - r_1 = 0 \tag{3.22}$$

$$r_{2,i} + r_3sin(\theta_{3,i}) - r_4sin(\theta_{4,i}) = 0 \tag{3.23}$$

$$r_3cos(\theta_{3,f}) - r_4cos(\theta_{4,f}) - r_1 = 0 \tag{3.24}$$

$$r_{2,f} + r_3sin(\theta_{3,f}) - r_4sin(\theta_{4,f}) = 0 \tag{3.25}$$

Note that as all angles are known, all the sine and cosine terms reduce to simple numerical values, resulting in a linear system of four equations in four unknowns, which is trivial to solve for $\{r_{2i}, r_{2f}, r_3, r_4\}$.

With these four lengths solved for, it only remains to determine the coupler (jaw) geometry, comprising the remaining unknowns $\{r5, \theta_{5,i}, \theta_{5,f}\}$. The coupler problem is as shown in Figure 3.10. The coupler geometry is defined by $\vec{R_5}$. The base of $\vec{R_5}$ translates by $\vec{\Delta}$ and

the tip of $\vec{R_5}$ translates by $\vec{J}$ from the initial to the final position. Additionally, the rotation of $\vec{R_5}$ is equal to the specified jaw rotation θ_{rot}.

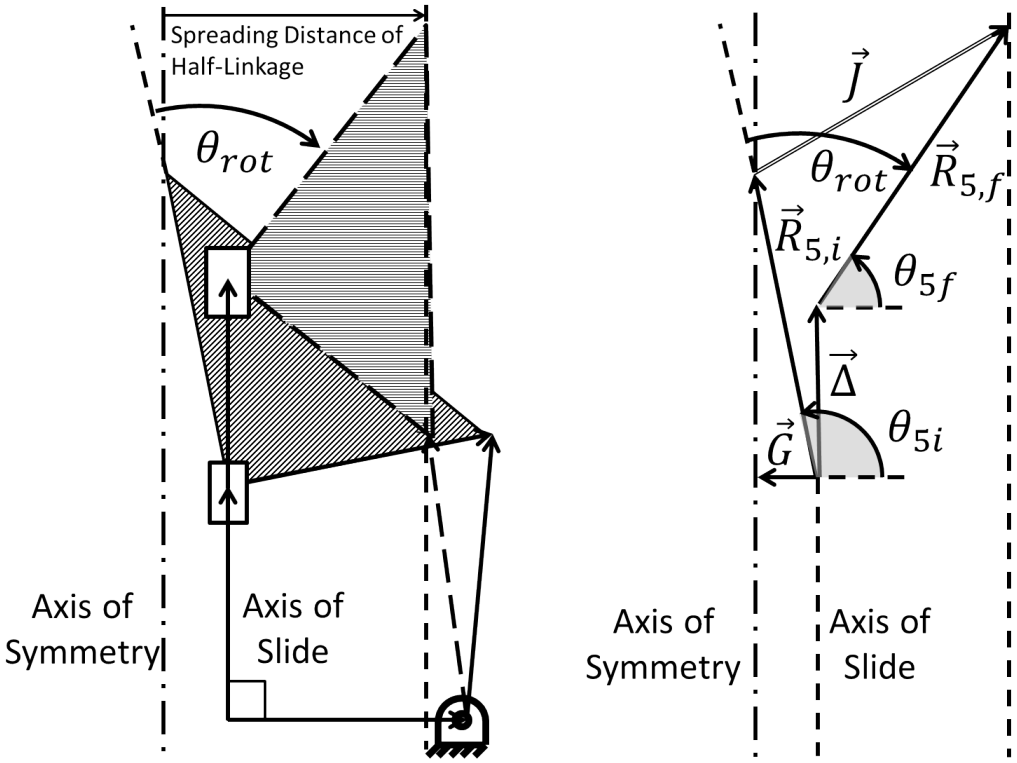


Figure 3.10: Coupler geometry problem

The problem amounts to solving the following four equations, which correspond respectively to the following conditions: real vector loop, imaginary vector loop, coupler tip on symmetry line in initial position, and specified rotation. As usual, lower case letters represent the lengths of vectors represented by the same uppercase letter. The formulation of the complex vector loop is omitted and the real and imaginary equations are immediately given.

$$r_5 cos(\theta_{5,i}) + J_x - r_5 cos(\theta_{5,f}) = 0 \tag{3.26}$$

$$r_5 sin(\theta_{5,i}) + J_y - r_5 sin(\theta_{5,f}) - \delta = 0 \tag{3.27}$$

$$g = r_5 cos(\theta_{5,i}) \tag{3.28}$$

$$\theta_{5,f} = \theta_{5,i} - \theta_{rot} \tag{3.29}$$

These four equations in the four unknowns $\{r5, \theta_{5,i}, \theta_{5,f}, J_y\}$ can be algebraically reduced to two equations in the two unknowns $\{\theta_{5,i}, J_y\}$:

$$g\Big(1 - \frac{cos(\theta_{5,i} - \theta_{rot})}{cos(\theta_{5,i})}\Big) + J_x = 0 \tag{3.30}$$

$$g\Big(tan(\theta_{5,i}) - \frac{sin(\theta_{5,i} - \theta_{rot})}{sin(\theta_{5,i})}\Big) + J_y = 0 \tag{3.31}$$

which are however nonlinear and nonreducible and must be solved numerically. After this is accomplished, the unique coupler geometry that will both touch the axis of symmetry in the initial position of the four-bar and provide the required x-translation when rotated by θ_{rot} is determined, and the linkage dimensions are now completely determined.

3.2.4 Validity of Solutions

For a solution to be considered valid, all links must lie in the first quadrant and correspond to a single branch of the linkage, and both transmission angles must be acceptable throughout the motion. The method of specifying the design variables deals with branch and transmission angle difficulties. There is nothing in the system of equations that requires the solution to lie above the x-axis, however, and in the first stage of solution, for the four-bar links without the coupler, the solution is occasionally outside the first quadrant. Often the solution can be reflected over the x-axis to recover a valid solution, as in the example shown in Figure 3.11 (note that the initial and final positions must switch identities when this happens).

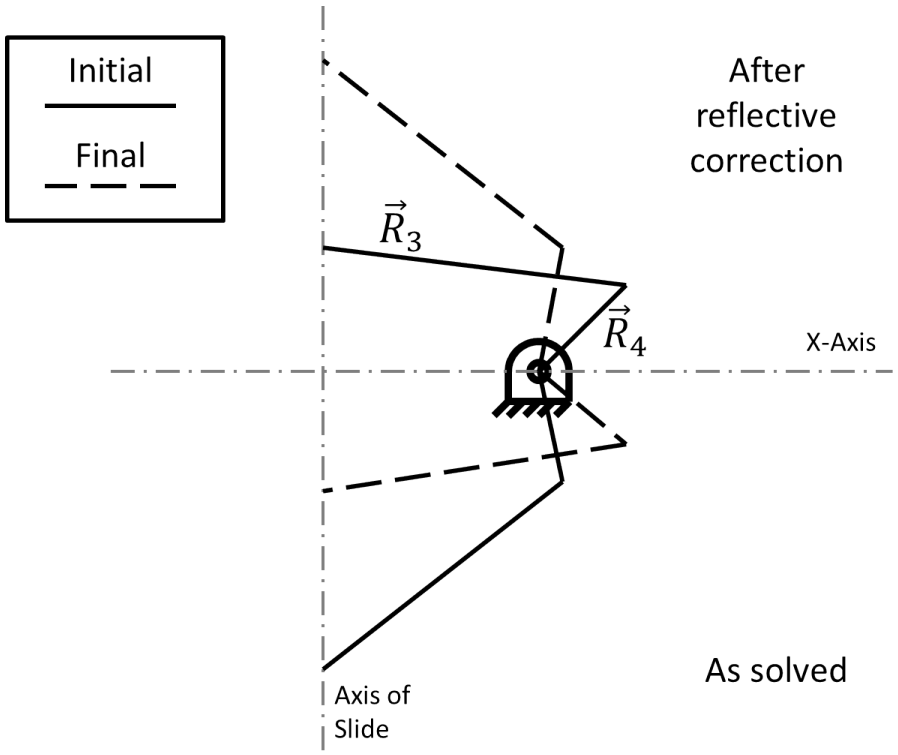


Figure 3.11: Reflection of solution over the x-axis

However, a more troublesome case can occur where one position is below the x-axis and one is above, as in Figure 3.12. The reflection trick will not work, of course, as this will merely switch which position is above and below. Uniform rotation of the solution about the ground pivot at $(r_1, 0)$ will result in altering the axis of slide, thus changing the solution. Uniform vertical translation will move the ground pivot off the x-axis. Therefore it must be concluded that no transformation will result in a valid solution, and the only recourse is to throw out the solution and try again. However, the majority of randomly generated solutions will be either initially valid or valid after reflection over the x-axis, so use of this formulation of the design variables is still beneficial.

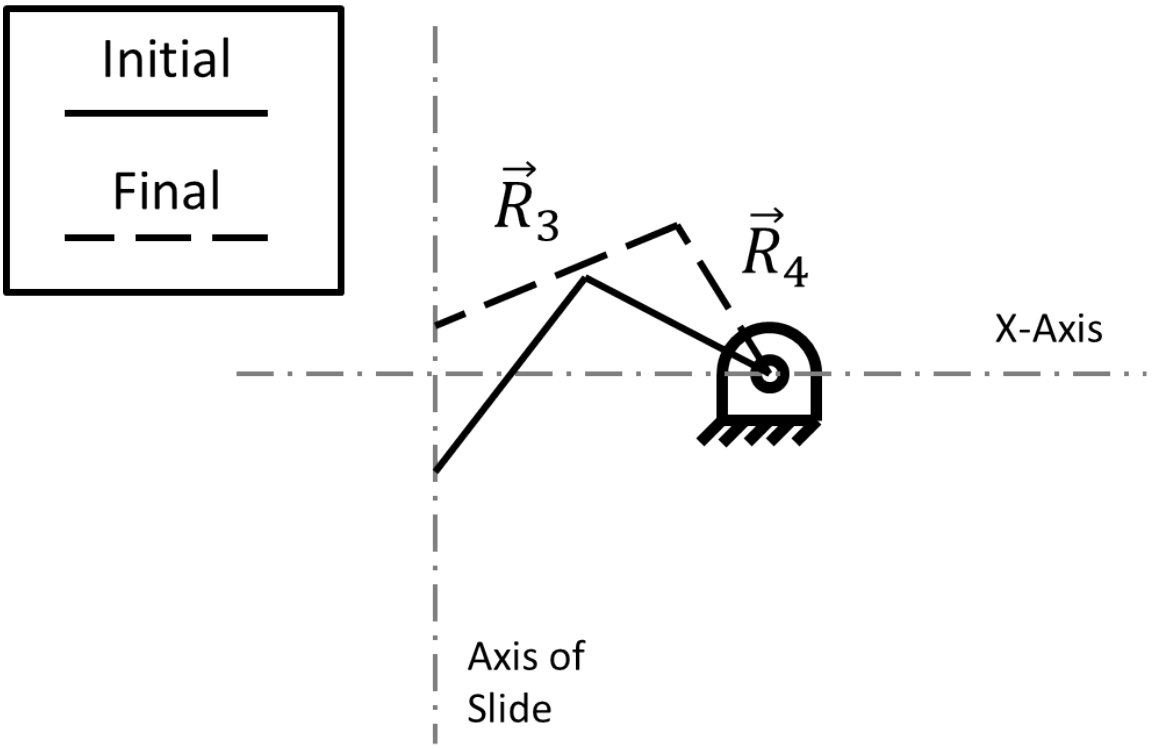


Figure 3.12: Invalid solution

3.2.5 Summary of Preliminary Kinematic Analysis

The preliminary kinematic analysis is now complete. The five kinematic design variables $\{g, r_1, \alpha_i, \gamma_i, \Gamma_f\}$ have been identified, and a procedure for fully determining a candidate linkage geometry corresponding to a given combination of design variable values, based on solving the spreader motion generation problem, has been given. The next section looks at the remaining work that must be done before evaluating the kinematic objectives and proceeding to the secondary structural optimization problem.

3.3 Primary Objective Function

Even after completing determination of the required linkage dimensions, it is still the case that only one of the three kinematic objectives can be immediately evaluated, namely

the jaw rotation, which since it is also a design variable is simply passed from input to output. The other two, mechanism length and spreading force variation, require some additional analysis.

3.3.1 Determination of Actuator, Joint, and Spreading forces

Specification of the linkage dimensions determines the kinematics and thus how much the spreading force varies as the mechanism moves from the jaws closed to jaws open position. The requirement that the spreading force be 10,000 lbf in the closed position scales this relative variation and allows determination of an absolute variation. As in many kinematics problems, quantities that are continuous over the motion were calculated at a finite number of discrete points. Five positions were evaluated for this problem, as determined by evenly spaced piston extensions between the closed and open positions. The force balance was solved at each of these five positions for a unit force applied by the hydraulic actuator. The resulting force at the jaw tips was then compared to the desired value of 10,000 lbf in order to determine a scale factor by which the results were multiplied to determine the actual values.

The first step is to assume that in any position the linkage is in force and moment equilibrium, which as the motion is very slow is an excellent approximation. The goal is to calculate the reaction force at the tips needed to maintain this equilibrium, as well as the joint forces involved in transmitting the force between the actuator and the tips. Note that when considering the half-linkage there is no requirement that the x-components of the forces on the piston and cylinder/frame balance, as there will be equal and opposite forces acting from the other side of the symmetry line. The forces acting on the half-linkage are shown in Figure 3.13, where F_{ab} is defined as the force exerted by a on b. The subscript definitions are:

- j = jaw

- p = piston
- 4 = connecting link ($\vec{R}_4$)
- g = ground (cylinder + frame)
- L = Load (at jaw tip)

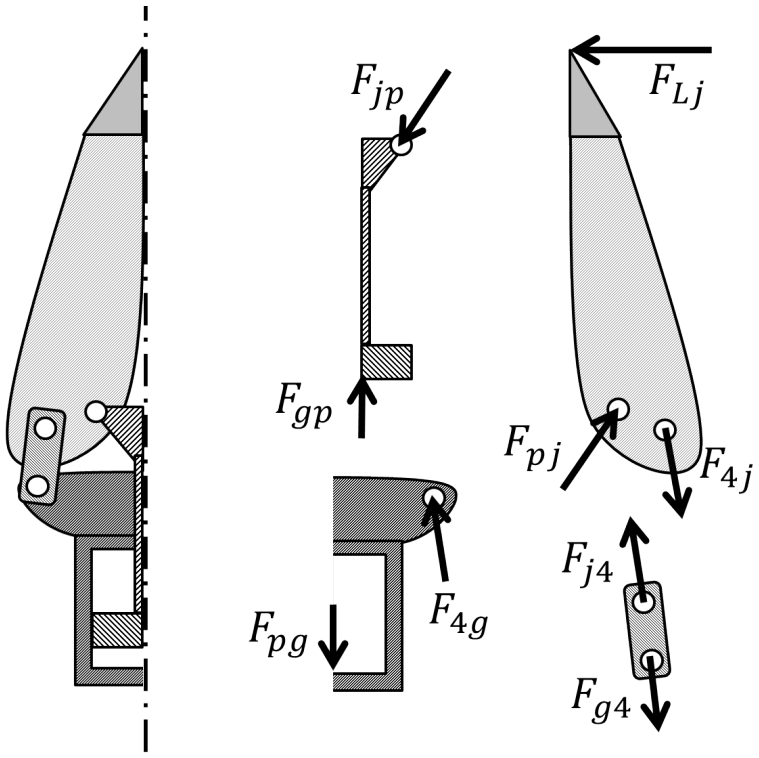


Figure 3.13: Semi-exploded view of mechanism showing free-body diagrams on links

A matrix formulation is often given for this type of problem, but as it is possible to solve the equations sequentially rather than simultaneously this approach is taken for reasons of computational efficiency.

One can begin with the pressure forces in the hydraulic actuator and work upwards through the linkage in a sequential manner, until the force at the jaw tip is determined. As the forces scale linearly, all that is needed is to multiply the results for the unit actuator

force by the correct factor to make $-F_{Ljx}$ = 10,000 lbf in the closed position. Once scaling is accomplished, pin diameters can be found by sizing them for a double shear failure mode with an appropriate safety factor, where in each case the diameter D is given by

$$D = \sqrt{\frac{2\sqrt{3}(SF)F}{\pi\sigma_y}} \tag{3.32}$$

where (SF) is the safety factor, F is the maximum joint force over the five positions evaluated, and σ_y is the yield stress of the material. The factor of $\sqrt{3}$ reflects the fact that shear failure occurs at $\sqrt{3}$ times lower stress than axial failure, and the factor of 2 reflects the double shear assumption.

The minimization of force variation objective is implemented as minimizing the amount by which the spreading force in the open position is greater than that in the closed position, with the caveat that zero is the lowest allowed difference to prevent the optimization from driving the force in the open position below the 10,000 lbf target. While it would be easy to add three additional objectives for the forces at the intermediate positions, it is always best to use as few objectives as will get the job done so as to simplify the optimization, and these are not really necessary. In order to address the last kinematic objective, total mechanism length, it is first necessary to digress into a discussion of how structural aspects of the problem will be modeled.

3.3.2 Link Model Categorization

Since the force balance of the previous section has already determined the forces acting on the individual links, it would be foolish to try to come up with a unified structural model for the entire mechanism, when the opportunity for a divide and conquer approach that looks at each link independently exists. With half symmetry, there are four links that must be modeled: the jaw, the connecting link, the piston, and the cylinder/frame. As always, a decision must be taken as to what type of model will give acceptable accuracy while being

as simple as possible. The essential choice is between a simple lumped analytical model in which the optimal design can be found by inspection (e.g. for a rod in uniaxial tension, the optimal design is just enough cross-section to carry the load), or if the situation is too complex, a finite element model with parameters that must be optimized in the inner loop.

The hydraulic actuator falls into the first category. It consists of a cylinder, two end-caps, tie rods, a piston hub and a piston rod. Due to the modularity of the construction, the uniformity of the applied pressure load, and the simplicity of the geometry, this lends itself well to a lumped analytical model. The connecting links are also quite simple, and can be modeled to good approximation by a uniaxial loading of a rod if buckling is taken into account. As no optimization is required, these structures are determined only by the lengths and forces that arise from the kinematic analysis, and are thus somewhat counterintuitively part of the kinematic outer loop rather than the structural inner loop. In constrast, the crossbar at the end of the piston that supports the two revolute joints, as well as the jaws, have complex geometry and loading and are thus treated with a finite element-based optimization.

Finally, a simplifying assumption is introduced regarding the flange mounted to the top of the cylinder that supports the connecting link ground pivots (see Figure 3.1). In general, there is a great deal of design freedom as to the location of these ground pivots, which need not be above the top of the cylinder, and can in fact be in any of three quadrants relative to the top corner, as shown in the right half of Figure 3.14. The necessary support geometry would difficult to parameterize in a way that allows continuous variation between the three regions, so the support location is arbitrarily fixed at the edge of the top endcap as shown in the left half of Figure 3.14. It is likely that the optimization would drive towards a similar case as this minimizes the size of the required support structure.

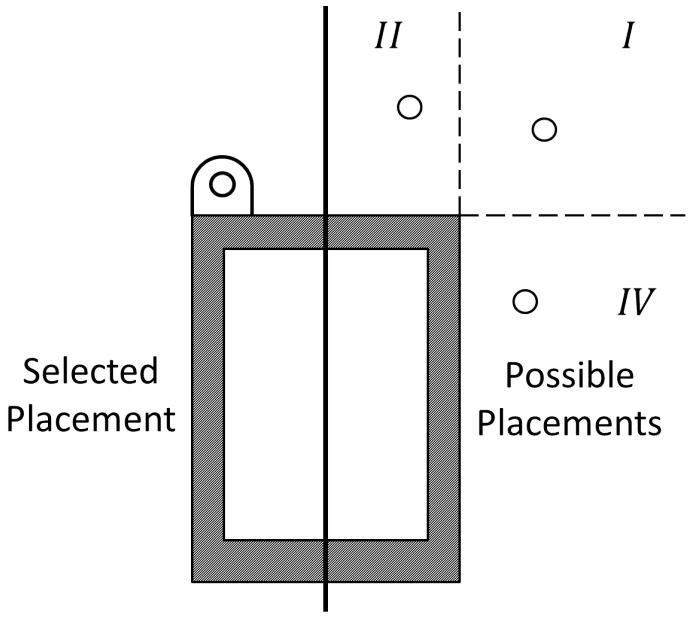


Figure 3.14: Possible locations of ground pivots

As only enough material is required to surround the pins, these supports are not modeled apart from assuming a fixed mass requirement of 1 lb of material. The surrounding radius of material is conservatively assumed to have a thickness equal to the pin diameter, that is, the pin center is 1.5 pin diameters above the surface of the endcap.

3.3.3 Hydraulic Actuator

This requirement serves to fix the relative vertical locations of the hydraulic actuator and the linkage, as the ground pivot just described is the same as that at the tip of $\vec{R}_1$. The automatic design of the actuator can now proceed. The required actuator force is known from the force balance, and the minimum required outer diameter (OD) of the cylinder is known from the requirement that the ground pivot be above the endcap. Note that the distance from the line of symmetry to the ground pivot is equal to $|g| + |r_1|$, both of which are design variables and have been specified by this point, that is, the positions of the ground pivots are fixed. There are five different components that must be designed: the

cylinder, the piston hub, the piston rod, the endcaps, and the tie rods.

Cylinder

The constraints on the cylinder design are a minimum OD, a required pressure force, and a maximum allowable pressure of 10,000 psi. The design equations are as follows. To ensure that the area, pressure and force are consistent,

$$ID = \sqrt{\frac{4F}{\pi P}} \tag{3.33}$$

where ID is the internal diameter, F is the actuator force, and P is the pressure. To ensure that the cylinder can withstand the stress of pressurization (using the thin-wall assumption),

$$t = \frac{(SF)\ P\ ID}{2\sigma_y} \tag{3.34}$$

where t is the wall thickness, (SF) is the safety factor, and σ_y is the yield stress. Finally, for dimensional consistency,

$$OD = ID + 2t \tag{3.35}$$

In general, for a given actuator force requirement a narrow, high-pressure cylinder will be lighter than a wide, low-pressure one. Thus the first attempted cylinder design strategy is:

1. Solve equation 3.33 for ID assuming the maximum pressure of 10,000 psi
2. Solve equation 3.34 for t
3. Solve equation 3.35 for OD

This may however result in an OD that is too small to reach the position of the ground pivots, as required by the support assumption. It is then necessary to expand the diameter

and reduce the pressure. Setting the OD to the value needed to reach the ground pivots and also still prescribing the actuator force F to match that demanded by the kinematic analysis results in the need for a simultaneous solution of equations 3.33, 3.34, and 3.35, leading to the condition

$$t = \frac{OD}{4} - \sqrt{\left(\frac{OD}{4}\right)^2 - \frac{(SF)F}{\pi\sigma_y}} \tag{3.36}$$

which allows subsequent solution of equation 3.35 for ID and of equation 3.33 for the required pressure, which will be less than 10,000 psi. However, if too great a force is demanded for the size of the cylinder, equation 3.36 may fail to have a real solution, in which case the third case must be resorted to, where the smallest value of OD that will yield a real solution is taken, that is,

$$OD = 4\sqrt{\frac{(SF)F}{(\pi\sigma_y)}} \tag{3.37}$$

and thus

$$t = OD/4 \tag{3.38}$$

This last case clearly violates the thin-wall assumption, but this is not a concern as there is essentially no chance that any solution of this character will be selected. This is due to the fact that mechanisms with large cylinder diameters quickly become extremely massive, primarily because the piston hub mass is proportional to the third power of diameter, so all the optimization needs is a somewhat realistic model to tell it to avoid these solutions.

In any case, one way or another the annular cross-section of the cylinder is now determined, and once the required length is known after some additional piston calculations the mass is easily evaluated.

Piston Hub

The piston hub is an easy case as it is assumed to not be in danger of structural failure, and is sized only on the basis of making sure enough material is present to install the requisite sealing and/or bearing hardware. The height of the piston is simply set to half the diameter as a conservative assumption, and the design task is complete. The required cylinder length can now be found, as the piston stroke (determined by solution of the kinematics) added to the piston hub height, with an extra half inch to allow for some clearance at the ends of travel.

Piston Rod

Due to its generally short length to diameter ratio as well as the support condition where it passes throught the top endcap, the piston rod is assumed to not be in danger of buckling failure, so the design reduces to a simple axial stress calculation. The mass m is given by

$$m = \rho \frac{(SF)F}{\sigma_y} L \tag{3.39}$$

where the length L is determined by adding the piston stroke to the extra distance the rod reaches above the top endcap when fully retracted, which is taken to be enough to be level with the centers of the ground pivots.

Endcaps

A simple model for the endcaps is used where they are of flat, of constant thickness, and are assumed to fail in shear at the ID of the cylinder. Since a material will fail in shear under a force $\sqrt{3}$ times less than that required for it to fail in tension, the magnitude of the shear stress at failure τ_y is

$$\tau_s = \frac{\sigma_y}{\sqrt{3}} = \frac{(SF)F}{\sqrt{3}\pi D t} \tag{3.40}$$

where F is the actuator force, D is the diameter of the cap and t is the endcap thickness. Solving for t,

$$t = \frac{\sqrt{3}\,(SF)\,F}{\pi D\,\sigma_y} \tag{3.41}$$

Thus the mass of a single endcap is found to be

$$m = \frac{\sqrt{3}\,(SF)\,\rho\,F}{4\,\sigma_y} D \tag{3.42}$$

and the total mass is twice this minus the volume of the top endcap removed to allow the piston to pass through.

Tie Rods

As the load is assumed to be evenly distributed, there is no reason to prevent determination of the total tie rod cross-sectional area needed as a unified quanitity, that is, the set of tie rods is treated as a single larger rod for analysis purposes. Note that this means the number of tie rods need not be specified. The load supported is the same as for the piston rod case, the load in question being the actuator force, so the result is the same as equation 3.39 with the replacement of the piston rod length with the cylinder length.

3.3.4 Connecting Link Design

With the design of the hydraulic actuator complete, the only remaining lumped analysis to perform is that concerning the connecting links (represented by the $\vec{R}_4$ vector kinematically). The requisite geometry is approximated by the model shown in Figure 3.15.

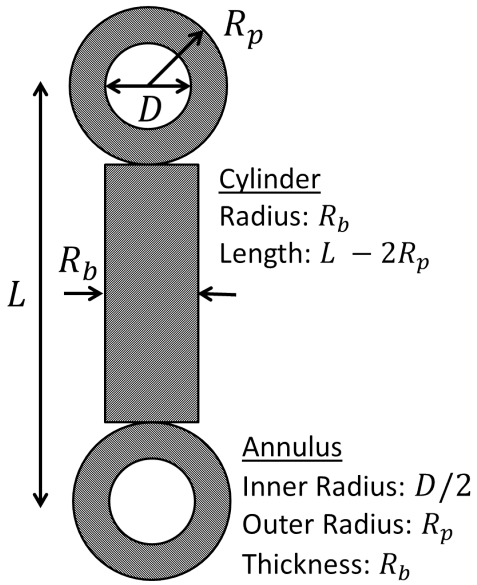


Figure 3.15: Model of Connecting Link Geometry

The pin diameter D is known from equation 3.32. The required value of R_p is estimated by simply setting it equal to D. The required radius of the middle portion is based on the buckling analysis described below. Once all geometric parameters are known, determining the mass of the link is a matter of a simple volume calculation and multiplication by the material density.

Column Buckling for Two-Force Members

When a two-force member is loaded in tension, a simple tensile stress calculation suffices to determine the required radius (and hence cross-sectional area). However, when the load is reversed as the linkage moves in the opposite direction, it is possible that a compressive load will induce buckling even if the axial stress is below the yield point. Thus a buckling analysis is necessary.

The maximum load that a column can withstand without buckling is known as the critical load, here denoted by F_{cr}. Two different buckling models are used in this analysis: the classic Euler model is used for long columns, which transitions to the Johnson model

for short columns (the exact definitions of "long" and "short" will be seen shortly). F_{cr} is given by Euler as

$$F_{cr} = \frac{\pi^2 EI}{L_e^2} \tag{3.43}$$

where E is the modulus of elasticity, I is the smallest second moment of area of the cross section about the buckling axis, and L_e is the equivalent length of the column. The Johnson model is

$$F_{cr} = \sigma_y A - \frac{(L_e \sigma_y A)^2}{4\pi^2 EI} \tag{3.44}$$

where σ_y is the yield stress of the material, and A is the cross sectional area of the column. Both models are as given in Juvinall and Marshek [80] with some slight algebraic rearrangement. The quantity L_e is the actual length of the column with a modifying factor to account for the effect of different end conditions. When both ends are pin joints, as is the case here, this factor is 1, so $L_e = L$. Equations 3.43 and 3.44 can be put in a form that allows calculation of the required radius from the column length by making two substitutions valid for a circular cross section,

$$I = \frac{\pi r^4}{4} \tag{3.45}$$

and

$$A = \pi r^2 \tag{3.46}$$

Additionally, we set $L_e = L$, and since we want the minimum safe radius, $F_{cr} = (SF)\ F$, where (SF) is the desired safety factor and F is the actual force acting on the member. All this results in

$$r = \left(\frac{4(SF)FL^2}{\pi^3 E}\right)^{1/4} \tag{3.47}$$

for Euler and

$$r = \sqrt{\frac{(SF)F}{\sigma_y \pi} + \frac{\sigma_y L^2}{\pi^2 E}} \tag{3.48}$$

for Johnson. The minimum safe radius from a simple compressive stress consideration is given by

$$r = \sqrt{\frac{(SF)F}{\sigma_y \pi}} \tag{3.49}$$

Equations 3.47, 3.48, and 3.49 are plotted in Figure 3.16 for the case of a high-strength steel link with $\sigma_y = 1.7 \times 10^5$ psi and $E = 3 \times 10^7$ psi under a compressive load of 10^5 lbf. The minimum safe radius (with $(SF) = 1$) as a function of the length of the link between pin centers according to Euler, Johnson, and compression considerations is shown by the solid curve, dashed curve, and dotted horizontal line respectively. Both quantities are in units of inches. It will be noted that the two buckling curves are in general tangent at the point

$$L = \sqrt{\frac{2\pi^2 EI}{\sigma_y A}}, \qquad r = \sqrt{\frac{2(SF)F}{\sigma_y \pi}} \tag{3.50}$$

which defines the cutoff point between "long" columns of larger L where Euler is valid and "short" columns of smaller L where Johnson is valid.

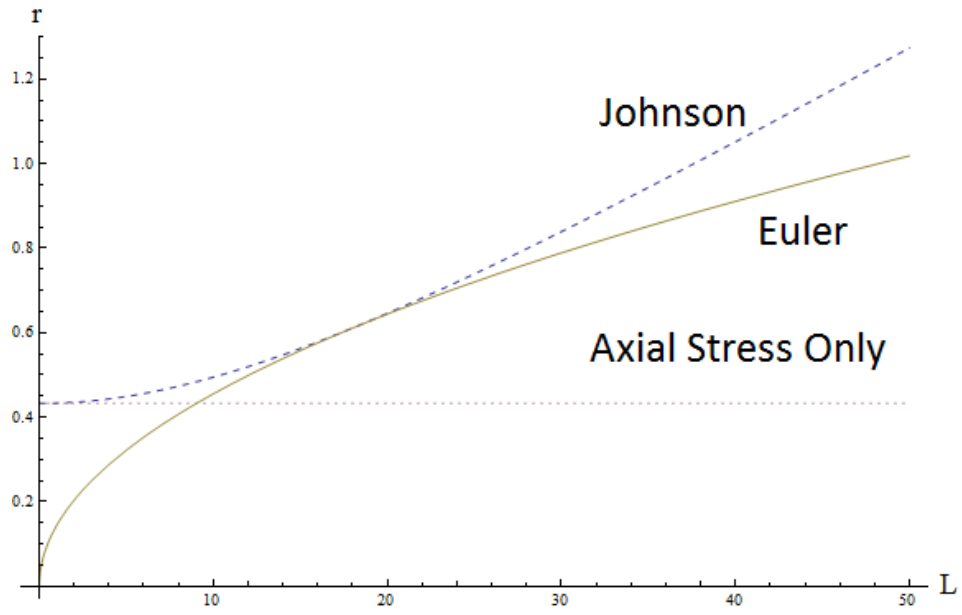


Figure 3.16: Typical Buckling Case for Two-Force Member

Therefore the buckling radius R_b was determined by comparing the length of the link to the tangency point, choosing the appropriate model, and evaluating r using that model given the values of the relevant parameters. As can be seen in Figure 3.16, the radius needed according to buckling considerations is always more conservative than that needed by simple axial loading considerations.

3.3.5 Summary of Primary Objective Function

The primary objective function call is complete after going through the calculations corresponding to the analysis in this section. With the knowledge of the length of the hydraulic actuator added to knowledge of the dimensions of the linkage, the total length of the mechanism can be determined, which along with the jaw rotation and force variation completes evaluation of the kinematic objectives. Additionally, the masses of the links modeled by lumped analyses are known at this point. However, evaluation of the non-kinematic objective (total mass of the mechanism) cannot be completed without determining the masses of

the jaws and the piston crossbar, which require more complex structural analysis. Referring to the nested optimization flowchart in Figure 2.12 if necessary, it is now recognized that the time has come to move on to the inner optimization.

The problem at hand is how to find the lowest possible mass that the jaws and crossbar can have, given the current values of the kinematic design variables. In other words, what is the most efficient possible structural design for these links given the candidate linkage dimensions? There are two main challenges associated with answering this question: how to parameterize the geometry in a general yet efficient way, and how to evaluate the stresses in the part for any given geometry. The literature relevant to these questions was surveyed in Section 1.2.3. Based on the information therein, it was decided to use a B-spline shape parameterization and a finite element-based inner objective function. Before proceeding to the details of the implementation used for this problem, a quick overview of B-splines and FEA is in order.

3.4 B-Splines

First, the properties of B-splines that are relevant to the current situation are discussed. For a much more detailed conceptual introduction, see Appendix A. If an in-depth mathematical understanding of B-splines is desired, the reader is referred to [81], or for the more mathematically inclined [82] might be more suitable.

3.4.1 What is a B-Spline?

A B-spline is a planar curve consisting of a number of smoothly connected polynomial sections, whose shape and position are determined by a separate entity known as the curve's *control polygon*. The control polygon consists of a number of individual points, known as *control points* or CPs, which are connected in a specific order to form the polygon. The control points influence the nearby portions of the B-spline, and can be moved around to

re-shape the curve as needed. Figure 3.17 shows an example of a B-spline, with the control points shown and connected by dashed lines to form the control polygon. Note that the last side of the polygon, connecting the first and last points in the ordered set of CPs, is not shown.

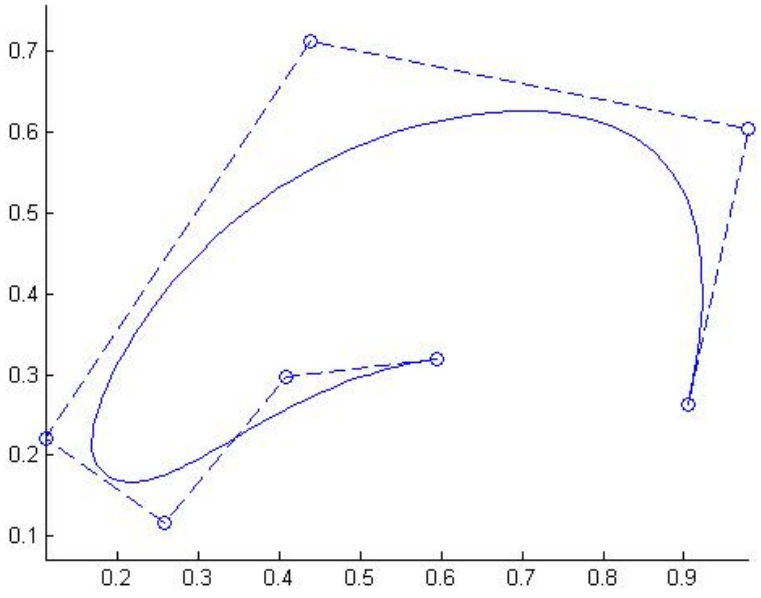


Figure 3.17: More complex B-spline example

B-splines are an excellent choice for representing link geometries in the inner optimization problem for a number of reasons. Taking the locations of the control points (CPs) as the design variables allows the representation of quite complex curves using relatively few variables, allowing a very general yet efficient formulation. Since changing the location of a CP will only affect the nearby portion of the curve, B-splines have the so-called local control property. This means that changes to a design variable (i.e. a control point) have only local effects, in contrast to the troublesome global perturbation to a simple high-order polynomial. Local control is illustrated in Figure 3.19, where the effect of moving the second to last CP defining the initial curve (seen again in Figure 3.18) is shown. Comparison

of the two figures will show how the majority of the curve is unaffected by the change.

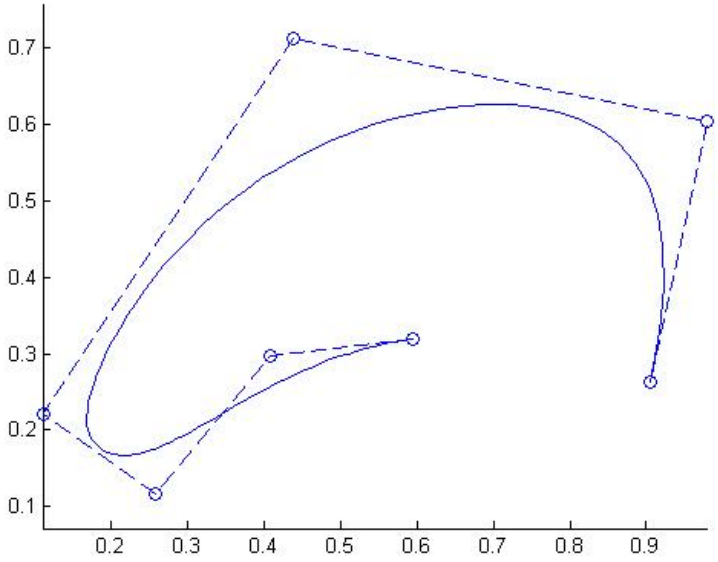


Figure 3.18: Initial curve

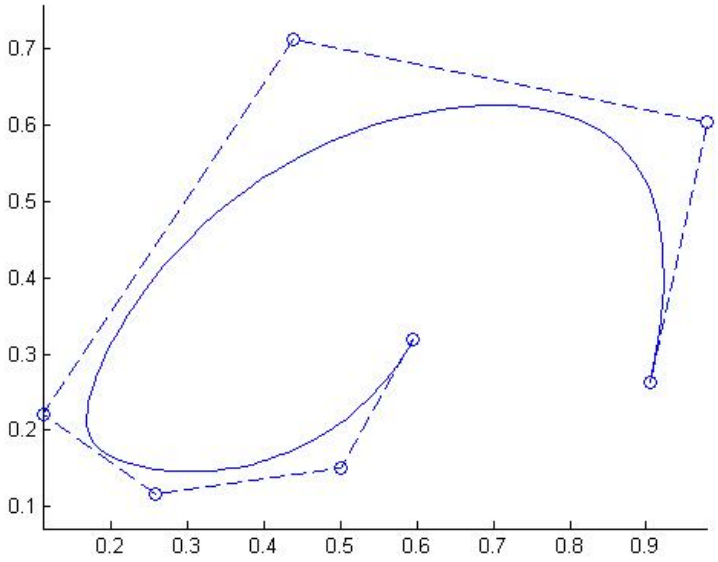


Figure 3.19: Locally modified curve

The local control property can greatly improve the ability of an optimization to isolate

objective dependence on an individual design variable and converge quickly. Additionally, as the spline consists of polynomial segments of only cubic order, computing the location of a point on the spline is much cheaper than in the high-order polynomial case. Finally, the spline always lies within the convex hull of the control points; in fact, a stronger version of this property holds wherein a given curve segment lies within the convex hull of the subset of control points that define it. This is very convenient for easily and efficiently applying constraints. For example, to ensure that the curve does not cross a certain line as the control points are moved, it suffices to check that the control points do not cross the line, which is far superior from a computational standpoint to evaluating and checking a large number of points approximating the curve itself.

3.4.2 Key Points

While the reader should consult the references if a true mathematical understanding of B-splines is desired, the qualitative points relevant to their use in the current problem are as follows:

- B-splines are a type of polynomial spline curve
- They can represent arbitrary planar curves accurately and efficiently
- The shape of a B-spline is determined by its control points
- Moving a control point re-shapes the curve *locally*
- B-splines can be evaluated and constrained with high computational efficiency

3.5 Overview of Finite Element Method

During the course of an inner objective function call, the shape of the part will be defined by the specification of control point locations as the design variable values and the

construction of a B-spline from them. The loads acting on the part are known from solution of the force balance in the outer objective function. To determine the stresses in the part as result of this geometry and loading, a finite element model is constructed and solved. The finite element method transforms a problem requiring the continuous solution of a partial differential equation over some domain into one that yields an approximate result by solution of a system of merely algebraic equations on a discretized domain. The construct that specifies the exact subdivision of the original domain is known as the mesh, and consists of two types of entities, elements and nodes. An element contains a small portion of the full solution domain, and usually takes the form of a Euclidean shape such as a triangle or quadrilateral. Each element has a number of nodes at predefined local positions such as vertices or centroids, each of which also has a global location at a coordinate point in the solution domain.

There are a large number of different element types available for different situations. For this case perhaps the simplest 2D element, the Constant Strain Triangle (CST), was selected. The CST consists of a triangle with straight sides and a node at each vertex, as shown in Figure 3.20. The name comes from the triangular shape of the element and the fact that the strain is approximated by a single constant value everywhere inside the element, in contrast to more sophisticated element types where the strain can take on a linear or higher-order distribution. The CST was chosen as it is easy to implement and computationally inexpensive, although it has less accuracy than higher-order elements. For the purposes of optimization this loss of accuracy is perhaps less troublesome than in a simple analysis problem, since as long as the optimization correctly identifies the trends in the relationship of objectives to design variables accuracy in an absolute sense is not as important. Once the optimal set of design variables is located a full-complexity manual design can be undertaken using commercial analysis software.

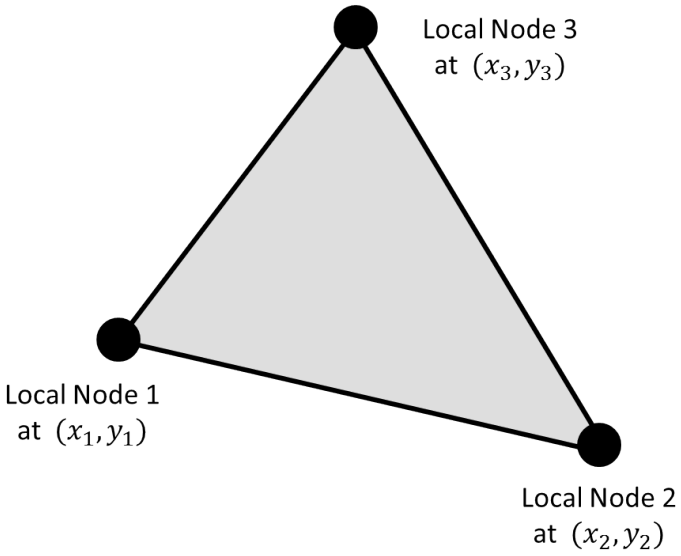


Figure 3.20: Constant Strain Triangle finite element

A qualitative examination of the steps needed to formulate and solve an FEA problem in practice is now presented. The implementations of these steps will be described in detail separately for the jaw and piston crossbar. A detailed development of the mathematics behind the finite element method for planar elasticity problems using the CST element is given in Appendix B.

Geometry

The first step in setting up an FEA problem is specification of the solution domain, which in the case of a structural problem means defining the part geometry. In the rescue spreader problem the geometry is restricted to a 2D shape with constant thickness to simplify the analysis, which is a natural extension of the fact that the optimization as a whole concerns a planar mechanism. The 2D outline of the part is specified by B-splines.

Meshing

The next step, once the solution domain has been specified, is the discretization of that domain, i.e. the creation of a mesh. Broadly speaking, there are two different tasks that are required to create a mesh: the specification of the global node locations, and the assignment of those nodes to elements as local nodes. Graphically, these correspond respectively to placing points throughout the domain and connecting those points with edges. The automatic creation of high-quality meshes without any user input beyond specification of the gometry is an entire field of research in its own right, particularly for 3D domains. The main concerns in creating a high quality mesh are maintaining good aspect ratios of the element shapes, in order to minimize numerical error, and putting smaller elements in regions where high field gradients are expected (e.g. stress concentrations) in order to increase the resolution of the approximate solution in that area. Any implementation of a finite element mesh must also take care to use an efficient data structure to remember which global nodes correspond to which elements as this data must be accessed many times during the FEA procedure.

Assembly

Once the mesh is defined, it is necessary to assemble the elemental stiffness matrices $\underline{\boldsymbol{k}}$ into the global stiffness matrix $\underline{\boldsymbol{K}}$. Qualitatively, this step involves integrating the indiviual pieces of the discretized domain into a single entity in such a way that the behavior of the original, un-discretized domain is approximated; the reader should consult the Appendix for the details of the process as well as the definition of the notation used. Since any real problem will have almost all zero-valued entries in $\underline{\boldsymbol{K}}$, in practice it is usually the case that a sparse array is used instead of a dense array, meaning that a 1D list of nonzero entries and their locations in the array is stored instead of the full 2D array of mostly zeros. For the spreader problem the meshes were small enough (about 1000 nodes) that the overhead involved in using the sparse array methods in Matlab made it more efficient to stick with

the dense representation (experimentation indicated that the break-even point was about 2000 nodes).

Boundary Conditions

At this point it is time to apply the boundary conditions for the problem, which consist of either a force or a displacement being prescribed for each node in the mesh. The details depend on the specifics of the problem as will be discussed later.

Solution

With the boundary conditions defined and the global stiffness matrix assembled, solution of the FEA problem is now possible. This solution entails solving the equation

$$\underline{\boldsymbol{F}} = \underline{\boldsymbol{K}}\,\underline{\boldsymbol{D}} \tag{3.51}$$

where $\underline{\boldsymbol{F}}$ is a column vector of nodal forces, $\underline{\boldsymbol{D}}$ is a column vector of nodal displacements, and $\underline{\boldsymbol{K}}$ is the square global stiffness matrix. In this type of problem $\underline{\boldsymbol{F}}$ and $\underline{\boldsymbol{K}}$ are known at this point and $\underline{\boldsymbol{D}}$ must be solved for. The naive approach is to find $\underline{\boldsymbol{K}}^{-1}$ and then left-multiply both sides of equation 3.51 by the result, giving

$$\underline{\boldsymbol{K}}^{-1}\underline{\boldsymbol{F}} = \underline{\boldsymbol{D}} \tag{3.52}$$

It turns out that this is just about the least efficient possible way to solve the problem, and a number of much more efficient and sophisticated techniques are available [83] that do not explicitly solve for $\underline{\boldsymbol{K}}^{-1}$ but use various methods to find $\underline{\boldsymbol{D}}$ directly. However, for kinematics problems where it is necessary to solve a problem on a mesh that is identical up to a rigid-body displacement (corresponding to the motion of a link from one position to another) multiple times, the best strategy is to solve the problem in a local coordinate system that is stationary with respect to the mesh and transform forces and displacements

to and from the global coordinate system as needed. This means that $\underline{\boldsymbol{K}}^{-1}$ can be found just once, and then right-multiplied by different $\underline{\boldsymbol{F}}$ vectors at each position that must be analyzed to find the various values of $\underline{\boldsymbol{D}}$. In other words, the FEA solution can be "recycled" without the need to remesh, reassemble, or re-solve the system, instead only performing the simple operations of coordinate transformation and matrix multiplication, which are many orders of magnitude faster. Direct solution for $\underline{\boldsymbol{K}}^{-1}$ takes 2-2.5 times longer than the preferred linear system solver in Matlab, but as in the rescue spreader problem 5 positions are evaluated and the time needed to "recycle" the known value of $\underline{\boldsymbol{K}}^{-1}$ is negligible, the overall cost decreases by over 2 times.

Post-Processing

After $\underline{\boldsymbol{D}}$ has been determined, by whatever means, the post-processing phase is entered in which this displacement information is converted into whatever else might be of interest in the problem. In the present case the elemental stresses are found from the nodal displacements contained in $\underline{\boldsymbol{D}}$. The FEA has now done its job and it is up to the inner optimization algorithm to make intelligent use of the information thus obtained.

3.6 Inner Optimization Implementation - Jaw

The necessary digression into the mathematical techniques underlying the method of analysis used is now concluded. The reader is reminded that before beginning this digression the description of the overall optimization had reached the point where the primary objective function evaluation was concluded, and the problem at hand was how to find the lowest possible mass of the jaws and crossbar given the current values of the kinematic design variables. Having outlined the underlying theory behind B-splines and FEA, the specific implementation of these concepts used to contruct the actual computer algorithms for execution of the inner optimization is now discussed. A detailed account is given of

the conversion of the inner level design variables to explicit geometry, as well as the FEA procedure used, for both the jaw and piston crossbar.

As the jaw is a somewhat easier case than the crossbar due to simpler topology and boundary conditions, it is treated first. The section begins with a high-level discussion of the modeling decisions made, after which the conversion of the inner level design variables to explicit geometry is explained. The finite element procedure is then laid out in the same order in which it is treated algorithmically.

3.6.1 Modeling Assumptions

The basic geometry that must be modeled is shown in Figure 3.21. The jaw consists of a single piece with two pins at the base and a serrated triangular tip at the top. The entire jaw is assumed to be of a constant thickness, in order avoid the need for a much more expensive 3D FEA solution. The topology of the part was chosen to exclude any internal cutouts beyond those needed for the pin holes. An internal cutout would roughly double the number of design variables needed to parameterize the geometry, as well as introduce greater opportunity for topological degeneration of the model as the boundaries shift during optimization, and it was felt the increase in modeling accuracy thus obtained did not warrant these sacrifices. The 2D assumption and simple topology together will actually produce a conservative (high) estimate of the required mass, as the actual jaw geometry will almost certainly have a webbed cross-section, which virtually all existing jaw designs do as well.

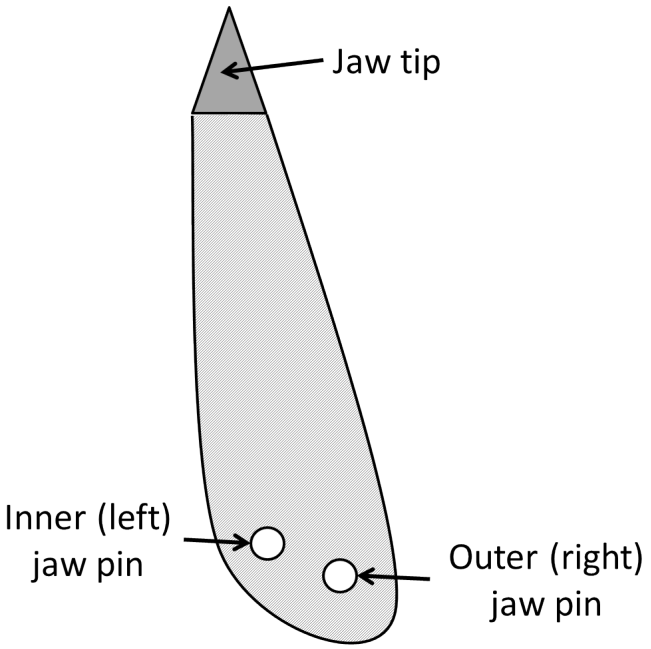


Figure 3.21: Jaw Geometry

The geometry of the tip is taken to be an isoceles triangle. The length is a balance between ensuring enough gripping area on the face and keeping it small relative to the body of the jaw. The angle spanned by the tip should be narrow enough to be inserted into small gaps but wide enough for strength. Based on examination of existing tip designs a 6 inch side length and a 30 degree span angle were chosen. For example, these characteristics are true of replacement tips for the Hurst SP-310 spreader obtained by the author (Figure 3.22). Since the boundary condition on the gripping faces of the tip will experience a wide local variation depending on the exact way that the tool is being used, the stress field in the tip itself is not modeled.

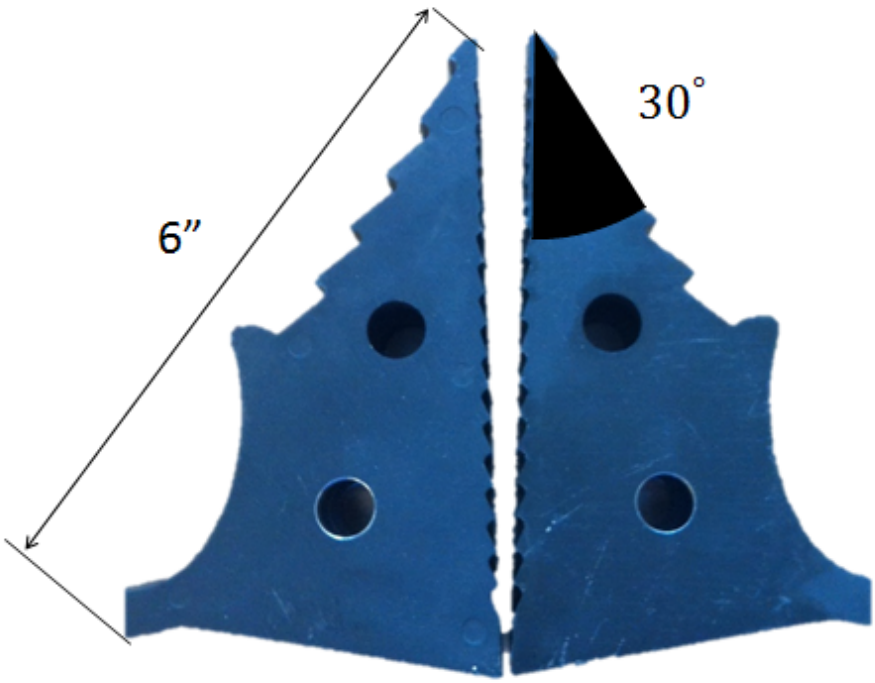


Figure 3.22: Jaw Tip Geometry

Similarly, it was decided to ignore the stress field in the neighborhood of the pin holes as well. In the author's experience with both the commercial FEA software ANSYS and the custom FEA procedure used for this problem, the stress field in the vicinity of pin bearing surfaces is artificially high due to simplified implementation of the boundary condition. A locally accurate model would need to include a nonlinear contact problem which would greatly increase the overall computational expense. Instead, the region around the pins is still meshed and solved in order to maintain proper behavior of the overall solution domain, but the stresses near the pins are simply ignored. It is expected that in practice this region will not be in danger of failure; if need be, a local thickening around the pins with minimal mass impact can be added. The boundary of the ignored region is given by a roughly horizontal line across the part, the location of which is determined by an iterative algorithm that will be explained shortly.

In reality, the boundary conditions on the jaw during operation will consist of bearing loads applied by the two pins as well as a force applied somewhere on the tip by the resistance of the target structure to deformation. In Section 3.3.1, the rigid-body force balance on the linkage was solved by assuming a point load applied at the geometric tip of the jaw tip itself, yielding among other things the required forces at the two pins. These pin forces are applied in the finite element model, but the tip force is replaced by the assumption that the jaw tip is a rigid, fixed body that produces a statically equivalent reaction force along the boundary between the jaw tip and the jaw body. This change will have some slight effects in the immediate vicinity of the boundary but will not significantly impact the part as a whole. This change is necessitated by the fact that at least one displacement boundary condition must be specified in order to preclude rigid body motion and obtain a solution using the finite element procedure. The high-level description of the jaw model is illustrated in Figure 3.23

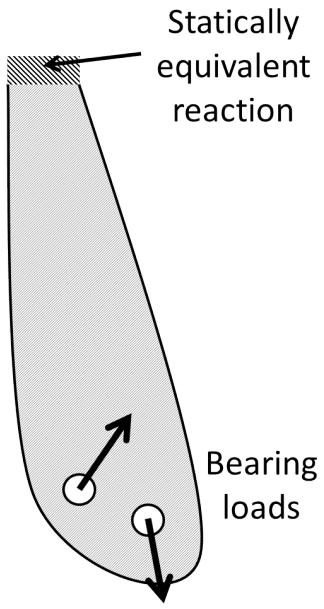


Figure 3.23: Jaw Model

3.6.2 Geometry Construction

The geometry corresponding to this model is specified by two types of quantities. The first are the design variables, i.e. the locations of B-spline control points, which are changed to optimize the part mass by the inner optimization algorithm. The second are fixed parameters, which as the name suggests do not change at the inner optimization level but are different for each candidate linkage (i.e. change at the outer level). These are the diameters of the pin holes, which are calculated from the assumption that the pins will fail in double shear, and the position of the base of the jaw tip relative to the locations of the pin holes. This last is determined from the coupler geometry calculated during the kinematic analysis, combined with the requirement that the inner face of the jaw tip be vertical in the closed position so as to produce surface contact with its counterpart on the other jaw.

An algorithm for creating an explicit geometry from these values must now be specified. The first step is to read in the values of $\vec{R_3}$, the vector from the left pin center to the right pin center, and of $\vec{R_5}$, the vector from the left pin center to the point of the jaw tip. The reader may refer to Figure 3.3 if a review of the vector formulation of the linkage is desired. These vectors are shown as solid lines in Figure 3.24. To simplify the mathematics, the jaw geometry is constructed in a local coordinate system in which the origin is the center of the left pin and both pin centers lie on the x-axis.

The jaw tip is contructed next. The tip of the triangle is coincident with the end of $\vec{R_5}$ by definition, and the angular position of the triangle about that point is defined by the requirement to have the inner face of the triangle vertical when the jaw is in the closed position. The position of the tip is needed only to specify the position of the base of the triangle, which is the fixed support surface in the jaw model.

At this point the inner level design variables exert their influence. A hypothetical line segment is contructed from the left pin center to the left end of the tip base, and similarly from the right pin center to the right end of the tip base. These are the long dashed lines in

Figure 3.24. Six B-spline control points, represented by black dots, are then placed at the ends of outward-oriented perpendiculars to the baselines thus constructed.

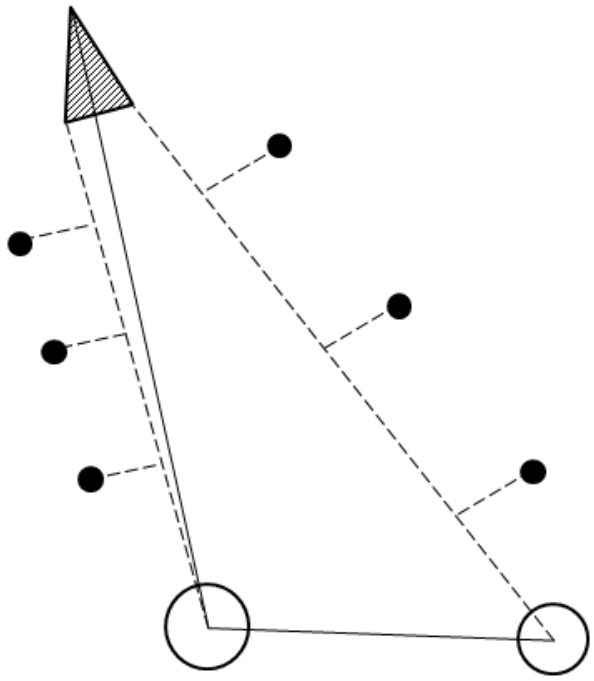


Figure 3.24: Construction of jaw geometry

A pause for a brief discussion of the way the control points (CPs) are located is in order at this point. The obvious way to do it would be to simply let the x and y coordinates of each point be the design variables. The drawback to this approach is that perturbation of the CP locations during optimization will result in just as much motion parallel to the part boundary as perpendicular to it. Motion of the CPs parallel to the boundary have little impact on its shape, and can in fact become a liability if all the CPs bunch up at one end of the curve, making control of the other end unstable. In essence, only one of the two degrees of freedom each CP has in the plane is actually relevant to the optimization, and varying both would waste computing power on attempting to optimize with respect to inert design variables. Therefore it is far more efficient to restrict the CPs to 1 DOF each, in a direction

perpendicular to the part boundary. Strictly speaking it would be necessary to compute the normal vector to the curve at the desired point in order to achieve this, but as the shape of the curve itself depends on the locations of the CPs this would have to be recalculated for every change. The method used here of allowing the CPs to move only perpendicular to the dashed baselines has essentially the same result but is much easier to implement and cheaper to calculate.

Thus there is only one design variable per CP, which specifies the perpendicular distance from the baseline to the CP location as a fraction of the baseline length. This has the 1 DOF advantage discussed in the previous paragraph, and also allows the design variables to be dimensionless quantities describing the curvature of the part boundary without reference to the absolute size of the part. Thus the CP locations are essentially specified in another local coordinate system nested inside the local coordinate system in which the jaw geometry is constructed, in which one dimension is given parallel to the baseline and the other perpendicular to it. The perpendicular coordinate is the design variable, as was just said. The CPs are given a uniform fixed spacing in the parallel direction, i.e. one, two, and three quarters of the way along the baseline. Mathematically, the location is easily calculated with basic vector geometry as follows. First, the appropriate length along the baseline is defined as $R_{||}$

$$R_{||} = \alpha R_5 \tag{3.53}$$

where $\alpha \in \{\frac{1}{4}, \frac{2}{4}, \frac{3}{4}\}$. Next, $R_{\perp}$ is defined by a CCW rotation of $\pi/2$ and a scaling by k, where k is the actual design variable corresponding to each CP and specifies the ratio of the length of the perpendicular to that of the baseline.

$$R_{\perp} = k(Im(R_{||}) - j\ Re(R_{||})) \tag{3.54}$$

Finally, the location of the CP R_{CP} is simply the sum of these two vectors. This calcu-

lation is shown in Figure 3.25.

$$R_{CP} = R_{||} + R_{\perp} \tag{3.55}$$

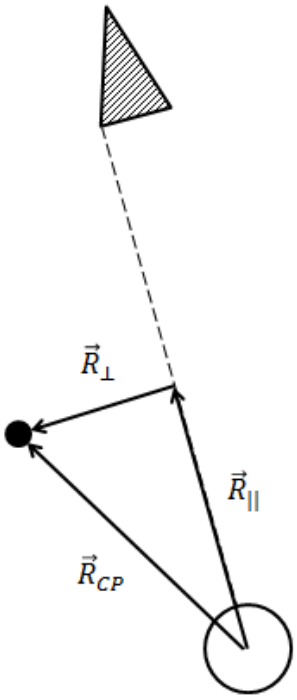


Figure 3.25: Location of control points

Using this procedure for the left side and a similar one for the right side, all six edge CPs are located from their design variable values, k_1 through k_6. An additional fixed CP is placed at either end of the jaw tip base. This of course requires no additional design variables as no DOFs are involved.

Thus the top and sides of the jaw are defined. The bottom portion is done as follows. First, two radii are specified, corresponding to the dashed circles in Figure 3.26. These radii are fixed at 3 times the radii of the corresponding pins. This is meant as a conservative estimate of the necessary radius of material to prevent pin tearout. As discussed, simple FEA is unreliable in the vicinity of the pins, and an attempt at analytical calculation

produced unrealistic results. This is a good example of a general principle of this type of complex modeling, which is that if a minor detail can be estimated to reasonable accuracy (say 90%), it's not worth potentially derailing the entire optimization by coming up with an unreliable model of greater complexity in an attempt to get to 95%.

Five additional fixed CPs are then placed at equal angular spacing along a portion of each circle (four are shown for graphical clarity). The start point of this arc is the point of tangency with a line from the circle edge to the fixed CP at the edge of the tip base. The end point is the bottom of the circle. A final fixed CP is placed at the midpoint of a line segment connecting the innermost CPs on each of the two circles.

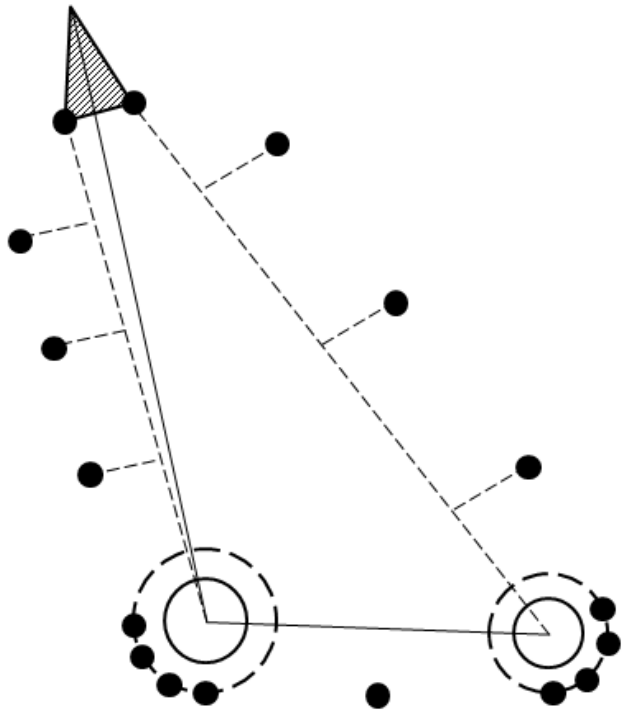


Figure 3.26: Construction of jaw geometry

With all control points placed, a single B-spline is constructed that begins at the left edge of the jaw base, proceeds CCW around the perimeter of the shape, and ends at the right edge of the jaw base. Placement of the fixed CPs at the base edges and at the bottom

of the shape do not contribute to the optimization, but greatly simplify construction of the geometry by allowing the entire boundary (with the exception of the line segment representing the base of the tip) to be represented by a single curve. An example of the B-spline thus generated is given in Figure 3.27.

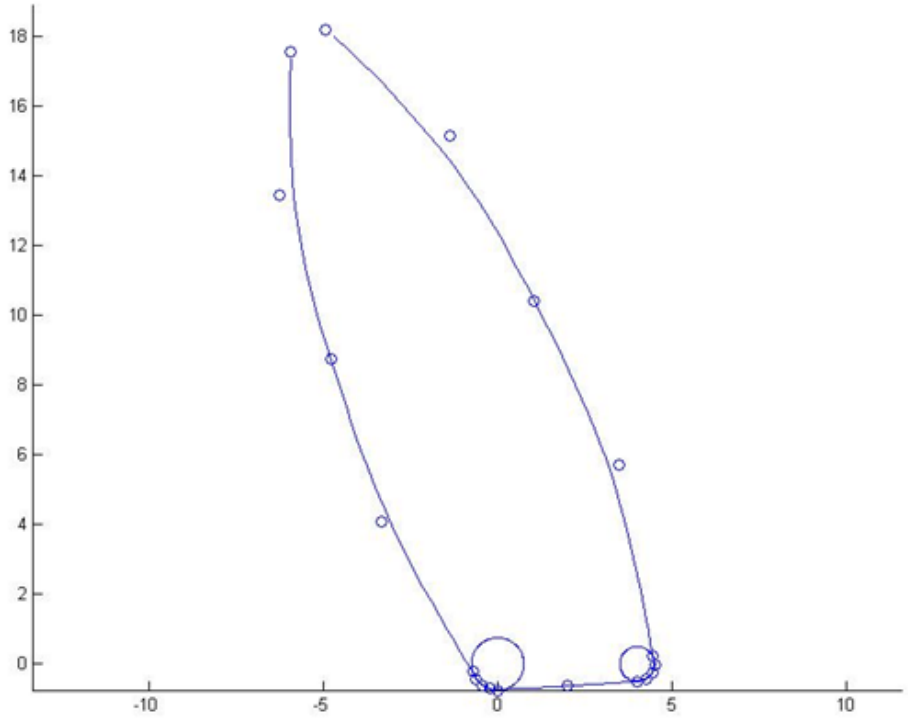


Figure 3.27: Example jaw boundary B-spline

Another example with the control points hidden but the tip shown is given in Figure 3.28. Also note that the pin holes are shown instead of the larger CP circles.

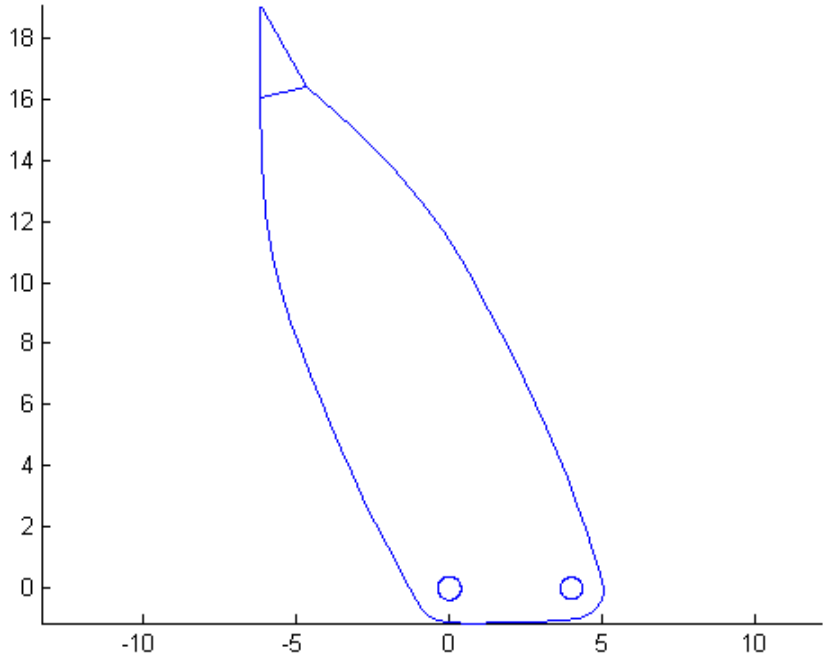


Figure 3.28: Example of full jaw boundary

As a final note on the geometry, the part is initially assumed to be of unit thickness, and a scaling to determine the true thickness is performed during post-processing of the FEA solution information.

3.6.3 Mesh Generation

With the geometry defined, the next step is to create a mesh of finite elements. As the mesh generation algorithm can be customized to suit the specific geometry, there is no need to use the sophisticated and expensive generalized methods used by commercial FEA software that must be able to handle any geometry the user chooses to input.

Location of Dividing Line

The algorithm starts by finding the cutoff point between the main body of the jaw and the bottom region around the pins where the stresses will be ignored. This cutoff point also allows much more efficient mesh generation once found, as will be seen shortly. Before describing the means by which this cutoff point is located, it is necessary to explain a minor obstacle to working with B-spline boundaries. A B-spline is defined parametrically; since it is one-dimensional a single parameter is used to define different locations along the curve. If this parameter is called u, one can input a u value and then evaluate the corresponding x,y point on the curve, but the algorithm cannot reverse the process and determine the u value that corresponds to a particular point on the curve. Therefore iterative "guess and check" type techniques must be used to accomplish this task. Due to the extremely low cost of u to x,y evaluations this is not much of an issue from the standpoint of computational expense, but does complicate algorithm design slightly.

The goal is to determine two u-values u_L and u_R that correspond to points on the curve that, when connected by a line, create the desired division. This is accomplished by sweeping a line down the shape until it hits one of the radii surrounding the pin holes (corresponding to the dashed circles in Figure 3.26). This condition is shown in Figure 3.29, where the line has happened to hit the left circle first.

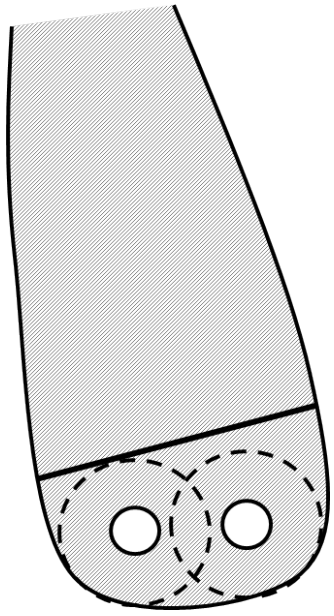


Figure 3.29: Desired position of dividing line

The convergence history of one example is shown in Figure 3.30, where the bottom line is the final position and the other lines are the positions calculated by the "sweep" algorithm during the solution process. The first few iterations are numbered to help illustrate the convergence trend.

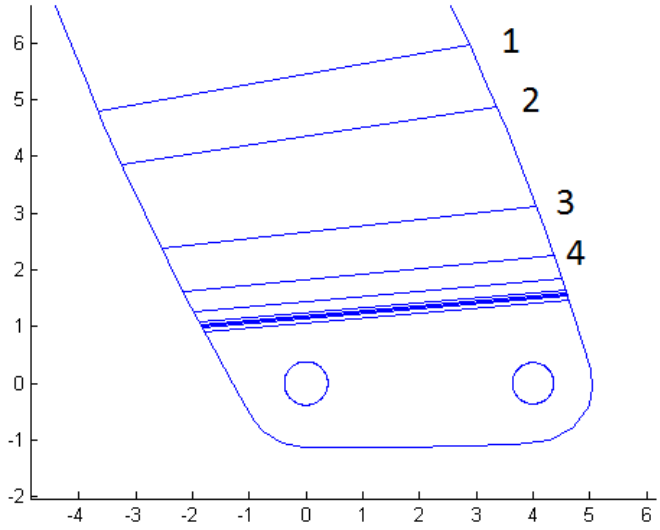


Figure 3.30: Convergence of dividing line

Upper Region Mesh

With the dividing line in place, it will be noted that the majority of the jaw area is now contained in the upper region. This region is now bounded by four curves: the top and bottom bounds are line segments, and the left and right bounds are B-spline segments. If the region is considered as a distorted rectangle, it is evident that a high-quality mesh can be produced extremely efficiently by mapping a rectangular grid onto it, as illustrated in Figure 3.31.

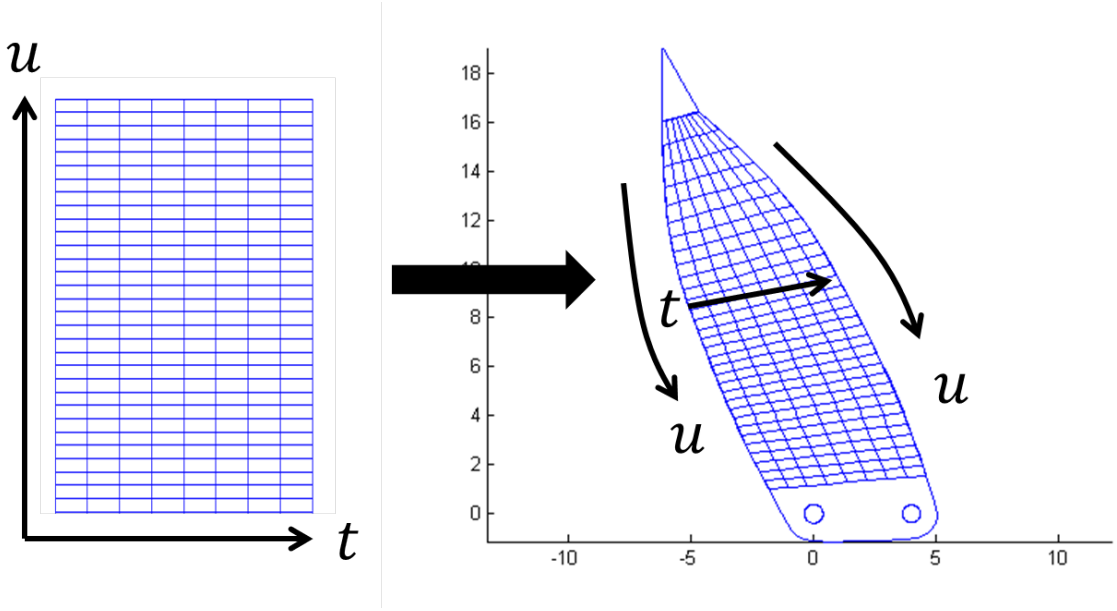


Figure 3.31: Main jaw region meshed by distortion of rectangular grid

This is accomplished by the following procedure. Two parametric coordinates are defined, u and t, where u corresponds to travel along the B-spline boundary and t corresponds to travel roughly parallel to the dividing line. In order to determine the optimal grid spacing, it is necessary to find the average size of the domain in the u and t directions. This is done by evaluating a number of discrete points along the sides of the domain between the known positions of the tip base and the dividing line, summing the chord lengths to determine the u-length, and averaging the t-distances at each evaluation to determine the t-length. See Figure 3.32 for a graphical representation of this process.

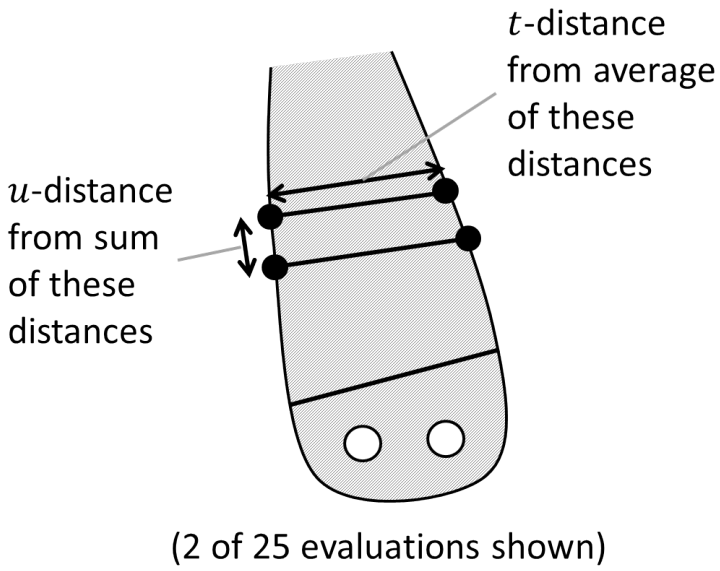


Figure 3.32: Determination of approximate u-t size of domain

Once the approximate u-t size of domain is known, the correct mesh grid spacing in both directions is determined based on a desired approximate number of elements using

$$\begin{aligned} D_u &= round\Big(\sqrt{\frac{L_u N_e}{2L_t}}\Big) \\ D_t &= round\Big(\sqrt{\frac{L_t N_e}{2L_u}}\Big) \end{aligned} \tag{3.56}$$

where D is the calculated number of divisions, L is the u or t length of the domain, and N_e is the desired number of elements.

With this calculation complete, a similar 2D sweep method is used to iterate through both dimensions using the computed grid spacings and place a node at each iteration forming a rectangular grid in the curvilinear u-t coordinate space, forming elements so as to make the corresponding quadrilateral elements. As triangular elements are used, the final step is to divide each quadrilateral element into two triangular elements across the short diagonal, so as to produce the best aspect ratios of the resulting triangles. This is shown in

Figure 3.33.

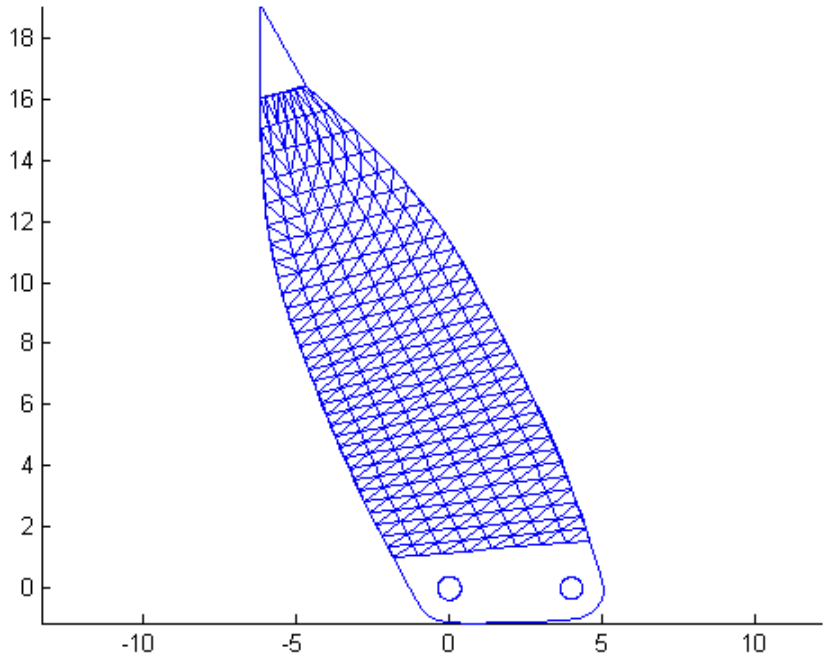


Figure 3.33: Division of quadrilaterals into triangles

The upper region mesh in now complete.

Lower Region Mesh

It still remains to mesh the lower region, which will be more difficult as the topological complexity introduced by the two pin holes prevents using the same strategy as for the upper region. The pin hole boundaries are represented by regular polygons with the nodes at the vertices; as the entire mesh ultimately must consist of nothing more than nodes connected by line segments this is the closest approximation to a circle that can be obtained. A number of nodes are placed along the bottom portion of the B-spline part boundary, and more are placed in the interior of the bottom region. The problem is then how to connect all these nodes so as to produce the highest quality triangular elements. This is accomplished

by means of a Delaunay triangulation, a well-known algorithm that solves exactly this problem [84]. This is a much more general method than the rectangular grid method, and is popular in many meshing applications. However, it is more computationally expensive than the rectangular grid method, so is limited to the relatively small lower region where it is needed to deal with topological complexity. By including the lowest row of nodes in the rectangular grid (i.e. those on the dividing line between the upper and lower regions) in the list of nodes to be triangulated on, it is ensured that the two mesh halves will match up properly. Additionally, there is the problem that the Delaunay triangulation only works on convex sets; that is, it will connect the pin bearing nodes to each other, thereby meshing the pin circles. Since the algorithm has no way of knowing which connections are undesirable, it must assume that all connections are valid. Exclusion regions corresponding to concave portions of the mesh must be externally specified, in this case by an IF statement skipping the formation of any elements that contain all three nodes on the bearing surface. The resulting lower region mesh is shown in Figure 3.34.

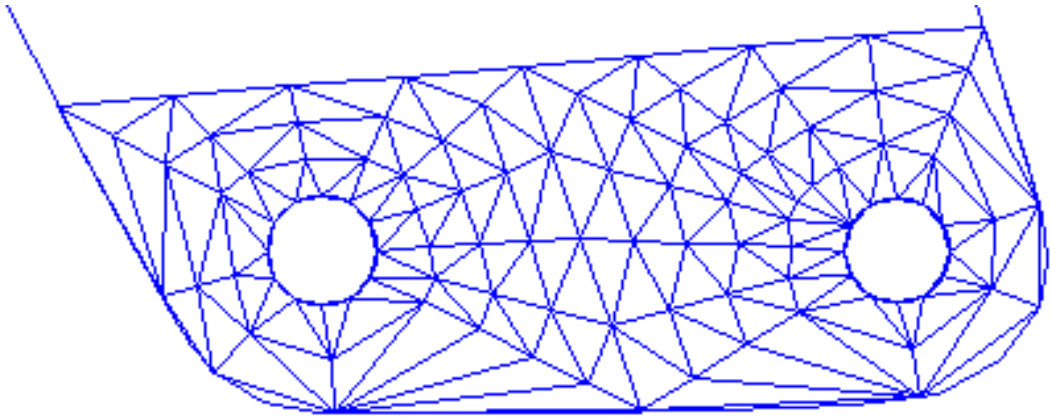


Figure 3.34: Lower region meshed by Delaunay triangulation

The entire mesh is shown in Figure 3.35

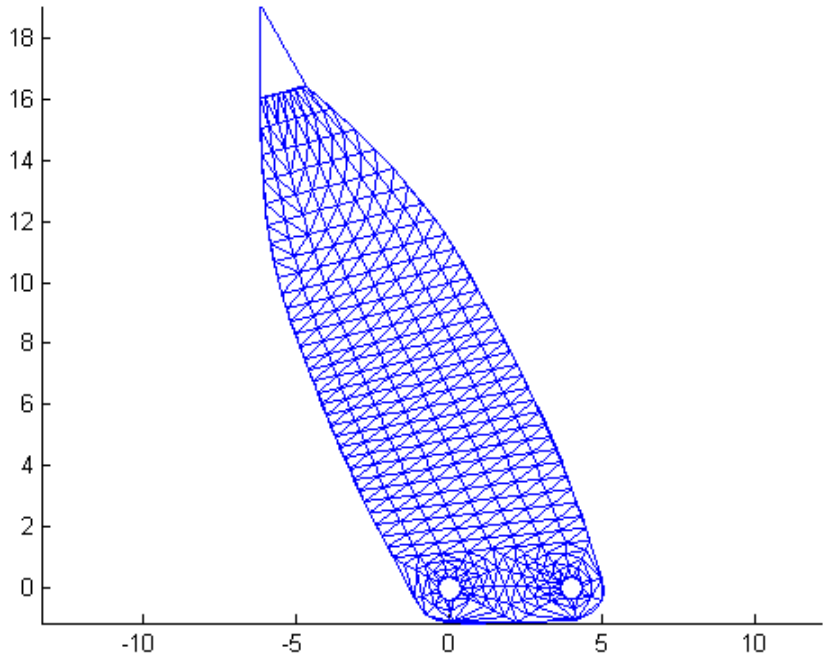


Figure 3.35: Completed jaw mesh

3.6.4 Application of Boundary Conditions

After meshing is completed, the next step in the FEA process is to specify the boundary conditions for the problem. For reasons discussed earlier, all the nodes on the top row of the upper region rectangular mesh are considered to be a fixed support, that is, are constrained to zero displacement in both the x and y degrees of freedom. The other boundary conditions are of course the bearing loads at the pins. The magnitude and direction of the applied forces are known from the rigid body force balance performed earlier in the evaluation of the current candidate linkage. However, only forces corresponding to x and y components on particular nodes can be specified in the FEA formulation, so the total bearing load must be split up into smaller loads on the nodes that form the bearing polygon. This is done using three assumptions:

1. The sum of the nodal forces must equal the total force on the bearing

2. Only the half of the bearing surface closest to the direction of the application of the load carries the load. That is, the back of the bearing sees zero applied load.

3. The distribution of nodal forces should approximate a cosine distribution as closely as possible.

This last condition is meant to approximate the true stress distribution in the bearing, as it satisfies the conditions of symmetry about the force vector and a smooth transition to zero value at angles of $\pm\pi/2$ relative to the force vector (see Figure 3.36). The cosine distribution is simply an intuitively developed second-approximation (where constant stress would be a first-approximation), since as stresses in the immediate region around the pins are ignored, exact accuracy is not important. The bearing model is shown in Figure 3.37.

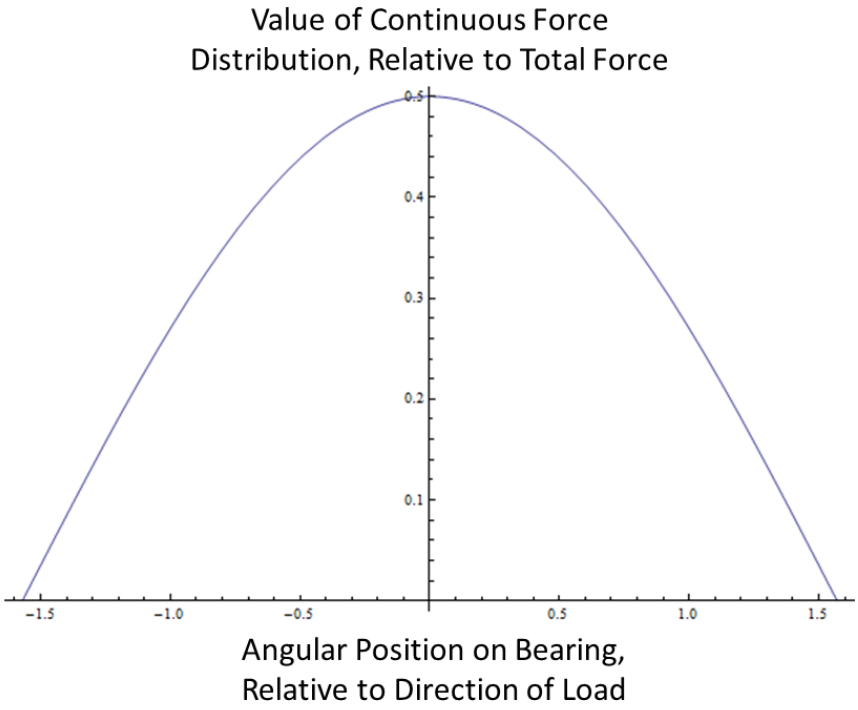


Figure 3.36: Force distribution over bearing surface

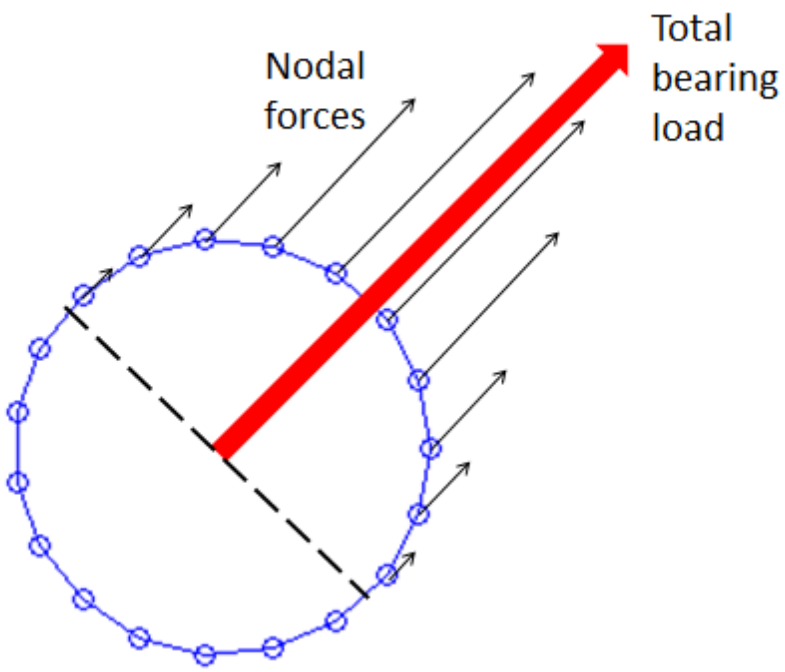


Figure 3.37: Model of the boundary condition at the pin bearing surface

3.6.5 Solution and Post-Processing

With the boundary conditions specified, equation 3.51 is now solved for the system. As previously discussed, the method with the best overall efficiency for a multi-position kinematic FEA problem is direct inversion of the global stiffness matrix and storage of the inverse for re-use at each position. Right-multiplication of the inverse by the nodal force vector solves for the nodal displacements. The stress value in each CST element is then computed from these displacements. For graphical output purposes, the continuous color scaling of Figure 3.38 is used to indicate relative stress magnitudes.

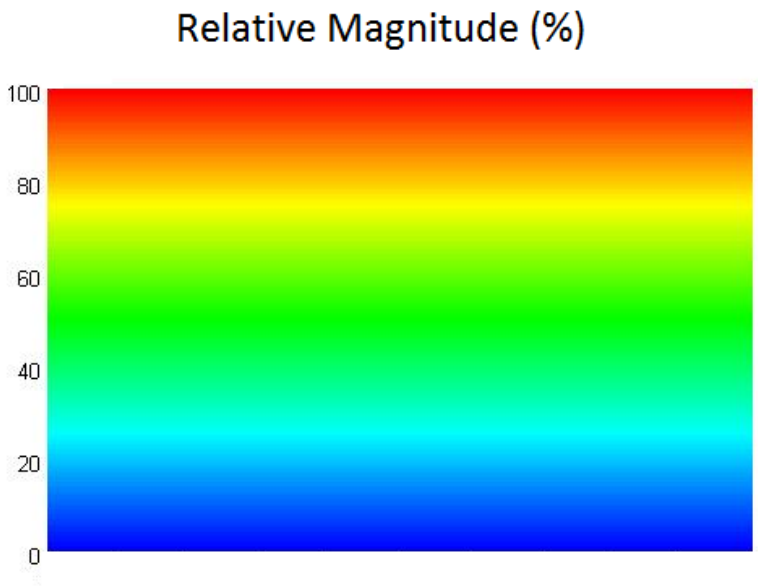


Figure 3.38: Continuous color scale for stress plots

A plot of von Mises equivalent stress for a typical case is shown in Figure 3.39. Note that as the spreading force is applied horizontally at the tip (insofar as the equivalent support boundary condition reproduces the effect correctly) the jaw is essentially a cantilever beam in bending, so the minimum stress at the neutral axis and maximum at the extreme fibers is exactly as one would expect. Also note that the bottom portion of the mesh is left blank as the stresses are ignored in that region, and the jaw tip itself is not modeled.

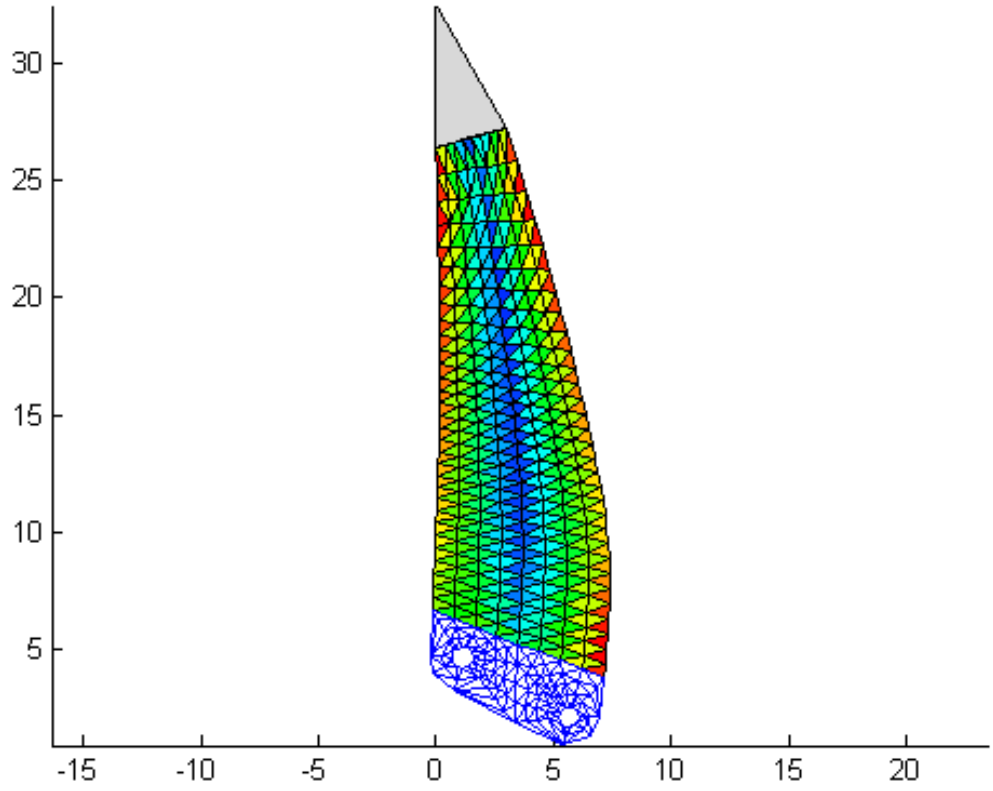


Figure 3.39: Typical von Mises stress distribution for jaw

It should be recalled that the solution corresponds to a part of unit thickness. The maximum stress in the part is found and compared to the material yield stress and safety factor, and the thickness scaled accordingly to satisfy

$$\sigma_{max} = \sigma_y/(SF) \tag{3.57}$$

The mass of the part can then be evaluated given the material density, since the thickness is known from the scaling just performed and the planar area is easily found by summing the areas of the mesh elements. With the determination of the part mass, the inner objective function call can now return.

3.6.6 Inner Optimization Algorithm

Now having the ability to evaluate the mass of the part corresponding to a given set of B-spline control point locations, it still remains to specify the method by which the optimal locations are determined. The choice of inner optimization algorithm takes advantage of two aspects of the nested optimization approach. First, any suitable algorithm can be chosen regardless of what is used at the outer level. Second, as only the structural design variables are being optimized, the nonlinearity of the problem is drastically reduced, and a traditional gradient descent method can be used instead of a more powerful but expensive metaheuristic algorithm with global optimization capability. Specifically, Matlab's built-in fmincon function was used, with the interior-point method selected. The design variables were the $k_1 - k_6$ specifying the lengths of the outward perpendiculars to the control points as discussed in Section 3.6.2.

Convexity Constraint

A nonlinear constraint function was also specified in order to impose convexity on the set of control points. As was discussed in the literature review, a common problem that has receieved much study in finite element-based shape optimization is the degeneration of the part boundary into jagged or otherwise unrealistic shapes. For example, given the initial geometry in Figure 3.40, an unconstrained optimization converged to the final geometry shown in Figure 3.41, which is clearly unsuitable (note that unlike Figure 3.39 the stresses are shown as discrete contours rather than a continuous variation).

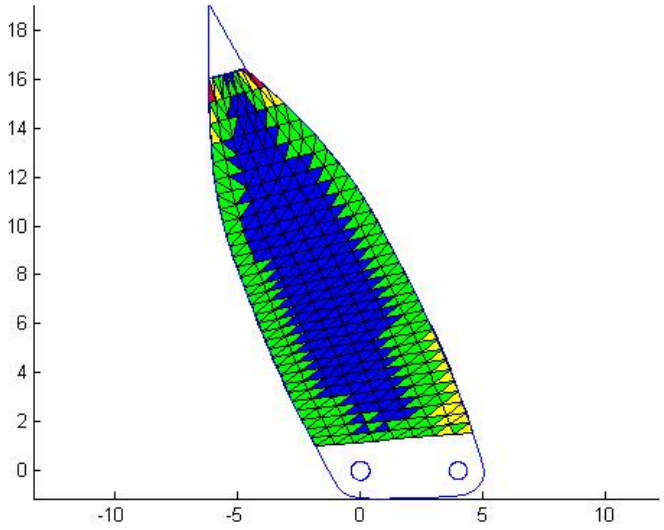


Figure 3.40: Initial jaw geometry

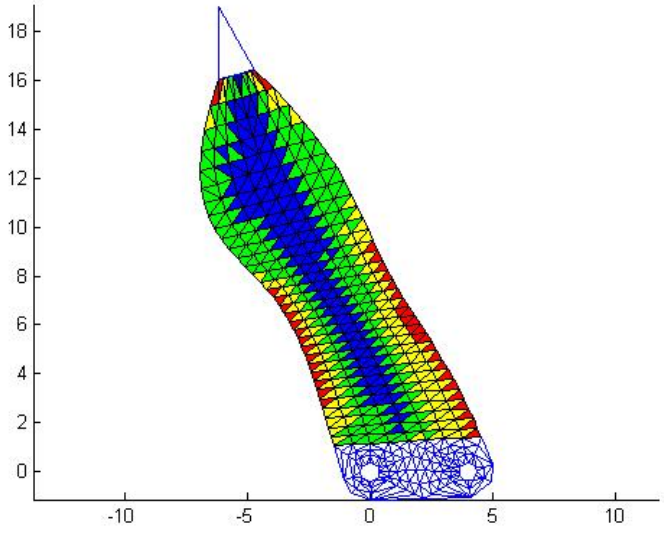


Figure 3.41: Unconstrained solution

The convexity constraint was introduced in order to prevent this type of shape degeneration. A convex set (in Euclidean space) is one in which a line segment connecting any

two points in the set remains entirely within the set. For example, this is true of a circle, but not of a crescent (Figure 3.42).

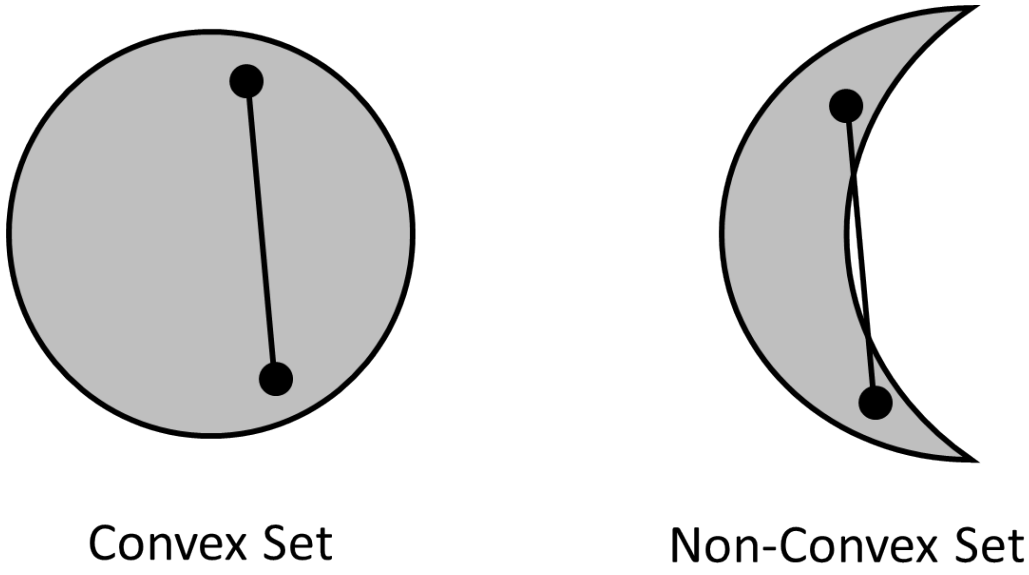


Figure 3.42: Convex and non-convex set examples

The constraint applied to the control points required that the control polygon formed by connecting the ordered set of control points with line segments be convex. This was implemented by computing the signed area of each set of three consecutive control points. A CCW orientation of the three-point set produces a positive signed area, whereas a CW orientation produces a negative one. Given the orientation of the B-spline as a whole, CCW orientation of each subset of three CPs is equivalent to convexity of the control polgyon. This fact is demonstrated in Figure 3.43.

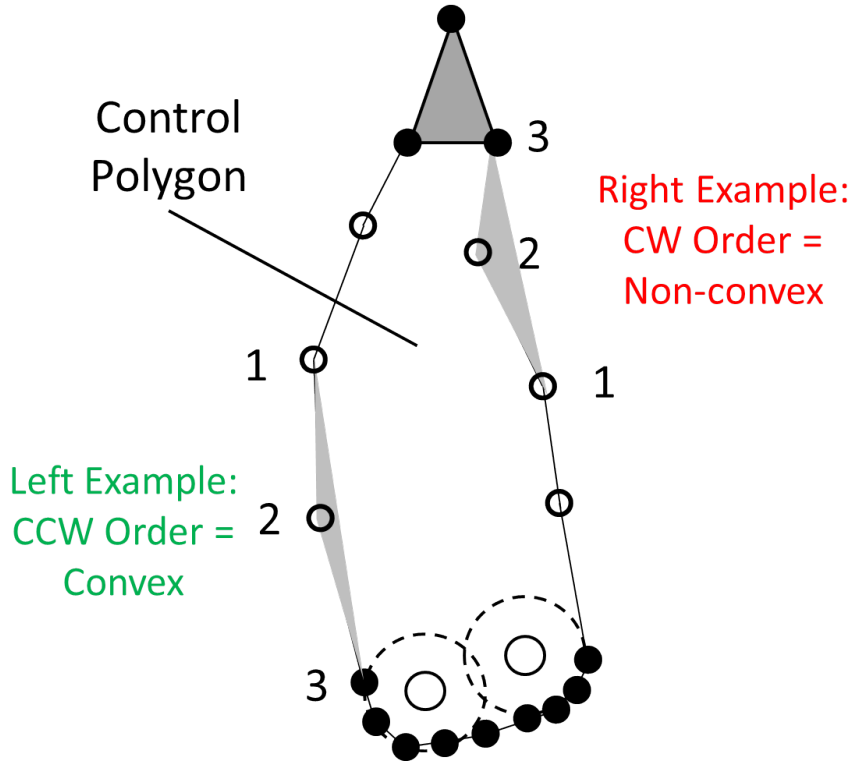


Figure 3.43: Equivalence of convex control polgyon and CCW-oriented CP subsets

Note that only the six "mobile" CPs corresponding to the six design variables are capable of violating this contraint; the fixed CPs on the circles are located on the boundary of a convex shape and thus are guaranteed to produce a convex polygon. Additionally, a dummy CP was added to the jaw tip, which was not used in B-spline construction but was included in the convexity determination so as to keep the entire jaw shape within the angle spanned by the sides of the jaw tip.

The constraint function functioned by summing the absolute values of the areas of the violating triangles while ignoring the complying ones, and the constraint was defined as requiring this sum to be less than or equal to zero, that is, the constraint would be satisfied only if no violating triangles existed. After inclusion of this constraint, the initial geometry of Figure 3.40 converged instead to the solution shown in Figure 3.44.

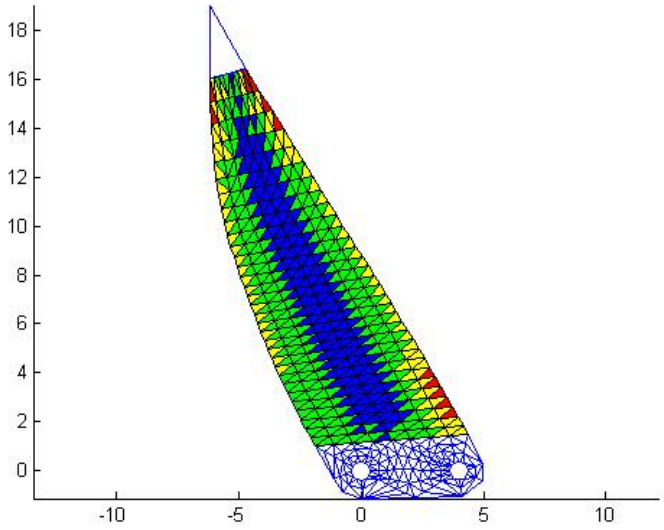
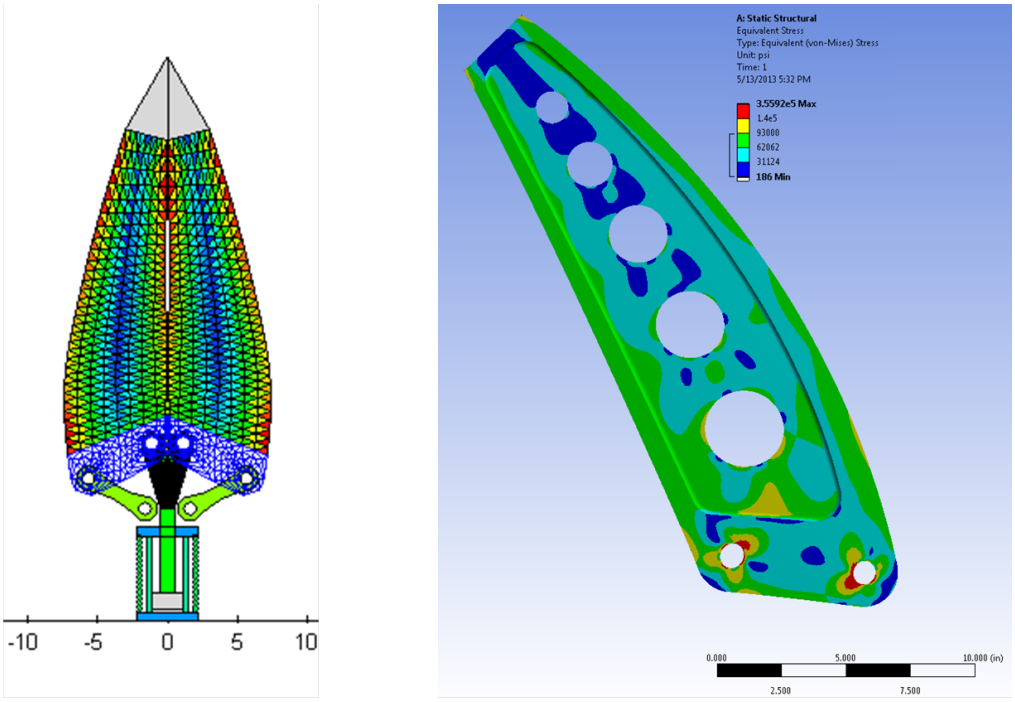


Figure 3.44: Constrained solution

With the use of the constrained gradient-descent inner optimization, the inner optimization was able to reliably determine optimal part designs and return the corresponding jaw masses to the outer objective function.

3.6.7 Validation of Jaw Model

The accuracy of the jaw model was examined by choosing a candidate design and performing a manual design iteration using commercial CAD and FEA software for those values of the kinematic design variables. As shown in Figure 3.45 the jaw masses were quite similar. The full 3D design freedom given by the use of commercial software allowed for a more efficient design, the mass savings of which were largely offset by the fact that the rather coarse CST mesh in the optimization tends to underpredict the peak stresses at the edge of the part as the CST element effectively averages the stress field over its area. However, it seems reasonable to be confident in the results of the optimization to within a few pounds.



Jaw Mass: 13.91 lbm

Jaw Mass: 14.52 lbm

Figure 3.45: Jaw mass comparison

3.7 Inner Optimization Implementation - Piston Crossbar

A similar optimization was required for the last remaining part of the mechanism, the piston crossbar. The crossbar is defined as the top portion of the piston rod that is not a simple cylinder and serves to support the pins joining the piston to the jaws.

3.7.1 Modeling Assumptions

While in reality the crossbar cannot be of strictly constant thickness in order to produce a smooth transition to the cylindrical piston rod geometry, a fair approximation may be obtained by the use of a 2D analysis, treating the piston rod as a horizontal 2D line support at the base of the crossbar. Additionally, as the part is symmetric only the right half is

modeled, with a symmetry condition imposed on the left side of the geometry. The basic geometry is shown in figure 3.46.

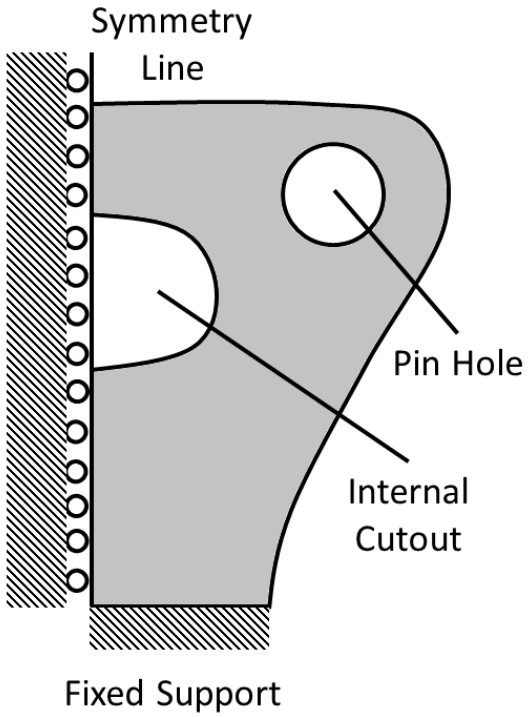


Figure 3.46: Geometry of piston crossbar

Note that unlike the jaw case, the topology of the part has been chosen to include an internal cutout. This is due to the fact that using an open contour that must terminate at both ends on the symmetry line uses significantly fewer design variables than a closed contour that is free to "float" in the interior of the part.

3.7.2 Geometry Construction

The parameters that define the basic shape of the geometry are shown in Figure 3.47. Two separate B-splines are now necessary due to the more complex part topology. The outer B-spline begins on the horizontal fixed support at location X, which is a fixed parameter as this must correspond to the piston radius for a smooth transition. The outer B-spline

ends on the vertical symmetry line at location Y, which is a design variable as there is no required location. The inner spline is defined using a local polar coordinate system as will be seen shortly, with the local origin a distance Y' above the overall origin, this distance also being a design variable. The pin hole center is located by the fixed vector $\vec{R}_p$ which is fully determined by specification of the kinematic design variables at the outer level. A circle of radius r is also defined, concentric with the pin circle but of twice the radius. r is also a design variable and the optimization is allowed to vary the initial value of twice the pin radius. It should be realized that although the B-splines are shown in Figure 3.47, this is only to orient the reader, since as the control point locations have not yet been defined the B-splines have not yet been constructed.

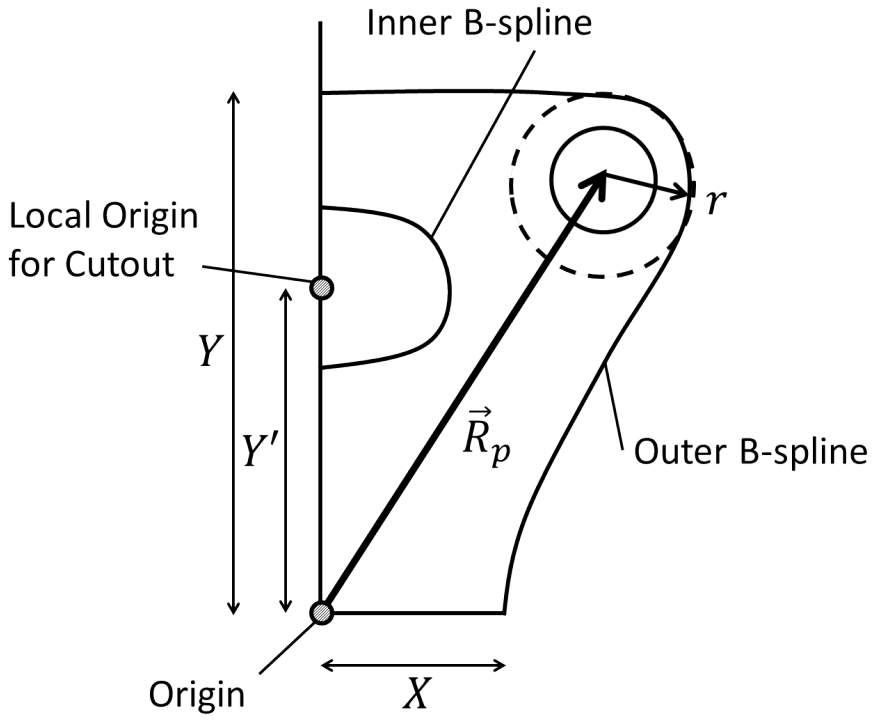


Figure 3.47: Geometric definition of piston crossbar

Placement of the control points proceeds as follows. There are five CPs on the inner spline, defined in a polar coordinate system relative to the cutout origin. The five CPs are

at equal angular intervals from $-\pi/2$ to $\pi/2$, with the radii being the design variables (the origin of the cutout can also slide up and down the symmetry line).

Placement of the outer CPs is considerably more complicated. As in the case of the jaw, a number of fixed CPs are placed on the dashed circle surrounding the pin hole. This reflects the fact that any rational design will minimize material usage by having the outer curve take a path that "rounds the corner" around the pin location as tightly as possible on its way from the fixed support to the symmetry line, leaving just enough thickness to retain the pin. Thus the CPs in that location are not granted computationally expensive individual freedom, but all move in or out based on a single design variable, r. 12 CPs are placed on the circle, but fewer are shown in Figure 3.48 for graphical clarity. These CPs are placed at equal angular spacing, with the angular limits defined by the design variables θ_1 and θ_2.

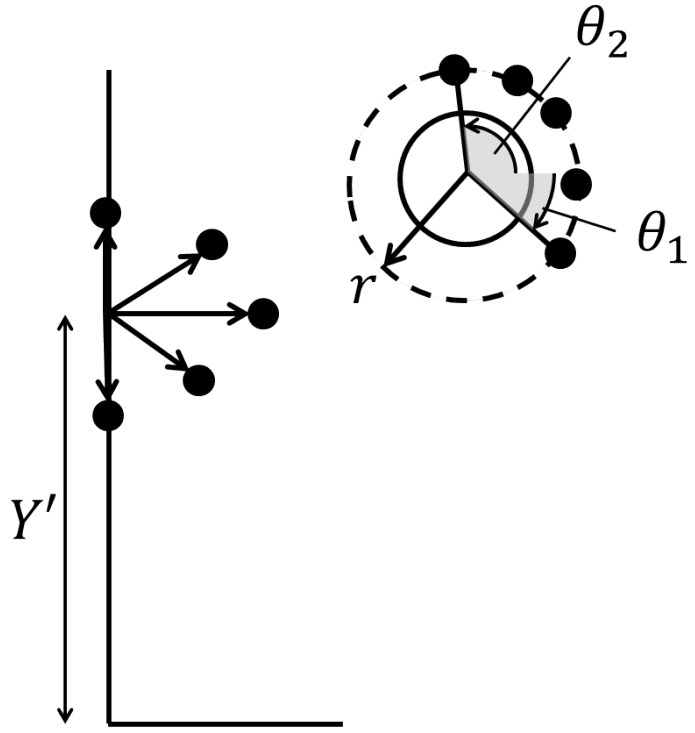


Figure 3.48: Placement of inner CPs and fixed outer CPs

At this point a difficulty arises from the fact that just as in the case of the jaw, some

constraint must be applied to the part boundary to keep it well-behaved, but unlike the jaw the crossbar may need to use a non-convex outer boundary, so the same constraint strategy cannot be used. For example, if the start point of the outer B-spline on the fixed support is at a smaller x-coordinate than the pin center, the curve will inevitably need to contain at least one inflection point (thus ruling out convexity) in order to both leave the fixed support at right angles (to maintain continuity with the piston rod) and make it around the pin. A similar condition obtains on the other half of the outer curve near the symmetry support.

The solution adopted is to only use a single CP between the fixed CP on the end of the support and the fixed CPs on the dashed circle surrounding the pin hole. Furthermore, this CP is placed using the same orthogonal vector method that was used in the case of the jaw. This strategy permits only one inflection point in the curve at each end, and reduces the number of design variables needed. The exact placement computation is as follows. A baseline is constructed from the support end fixed CP to the nearest circle fixed CP. A perpendicular is then placed halfway along the baseline, with length determined by the single design variable, and the CP is placed at the end of the perpendicular. This is illustrated in Figure 3.49.

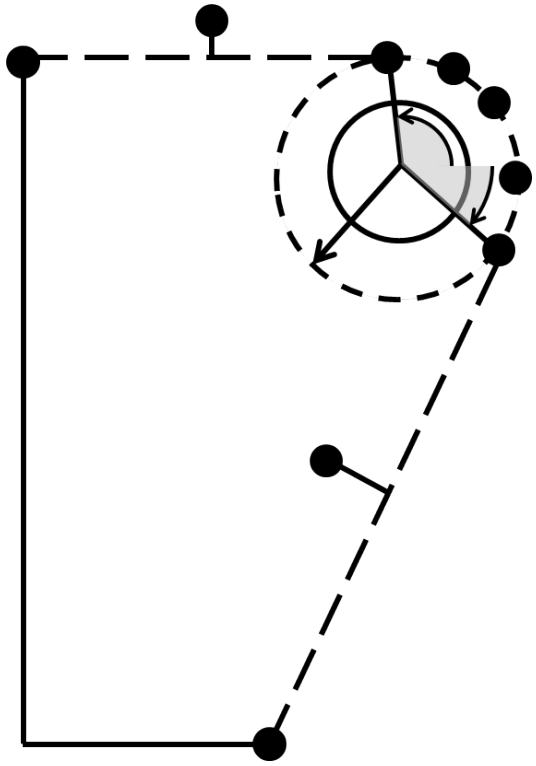


Figure 3.49: Placement of remaining outer CPs

With all the CPs placed, the B-splines can be constructed, as shown in Figure 3.50.

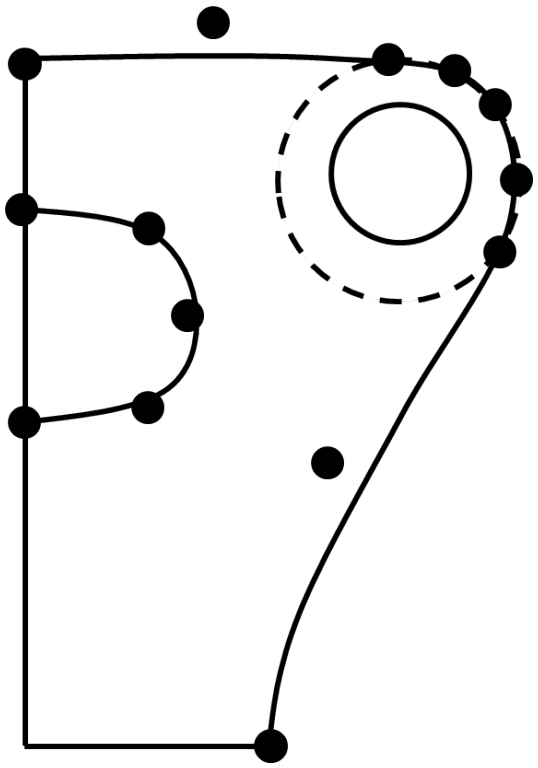


Figure 3.50: Completion of B-spline construction

3.7.3 Mesh Generation

Geometry construction is now complete, and the part next needs to be meshed. A similar strategy as in the jaw case is employed, applying a rectangular grid in a curvilinear coordinate system. Note that the vertical line segment between the origin and the bottom of the cutout was treated as part of the inner B-spline for meshing purposes, in order to maintain a rectangular boundary topology (two opposite B-splines, two opposite line segments). The fact that the pin hole is in the center of the part makes partitioning the domain more difficult than in the jaw case. The first step is to determine the u, t coordinates of the pin center. As previously discussed, B-spline evaluation can only go from parametric space to real space so iterative methods must be used to determine the location of a point in real space in parametric space. A custom algorithm was not needed for this iterative solution, unlike

the case of the jaw dividing line, so the built-in 1D system solver fminbnd in Matlab was employed to determine u_c value such that evaluating each B-spline at u_c and connecting the resulting points would produce a line that passed through the pin center (to within a finite error tolerance), as shown in Figure 3.51. The t location of the pin center was then found by analytical means, using the derivative of the Cartesian distance metric to determine the closest point of approach of the line thus created to the center.

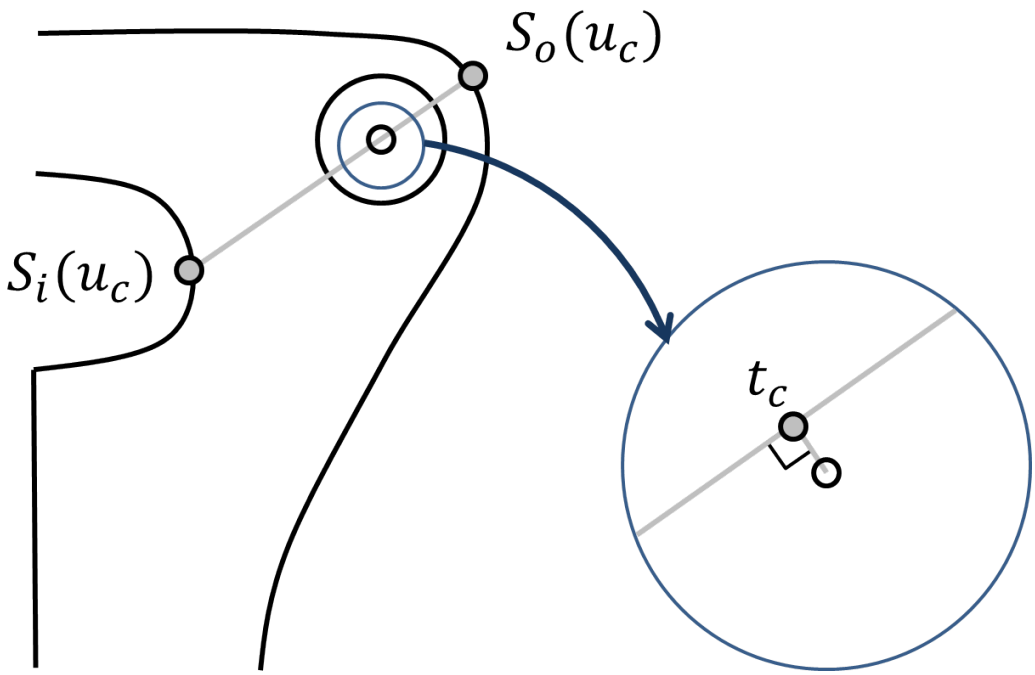


Figure 3.51: Location of pin center in u, t space

With the location of the center known, a similar line sweep method using fminbnd was used to search in either direction to determine the u bounds on the circle. The t bounds could be found directly using the radius of the circle. A rectangle in u, t space bounding the pin circle was now known, as illustrated in Figure 3.52

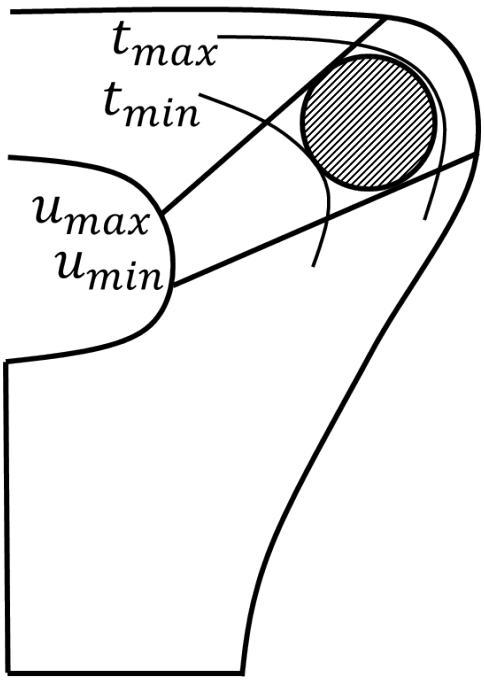


Figure 3.52: Bounds on pin circle in u, t space

Using this information, it was now possible to mesh the majority of the domain using a rectangular grid, bypassing the formation of any elements that would fall within the bounding rectangle on the pin. Due to the greater curvature of the B-splines that the mesh gridlines are mapped onto, element quality is a bit more variable than in the case of the jaw, but is still acceptable. A typical main section mesh is shown in Figure 3.53.

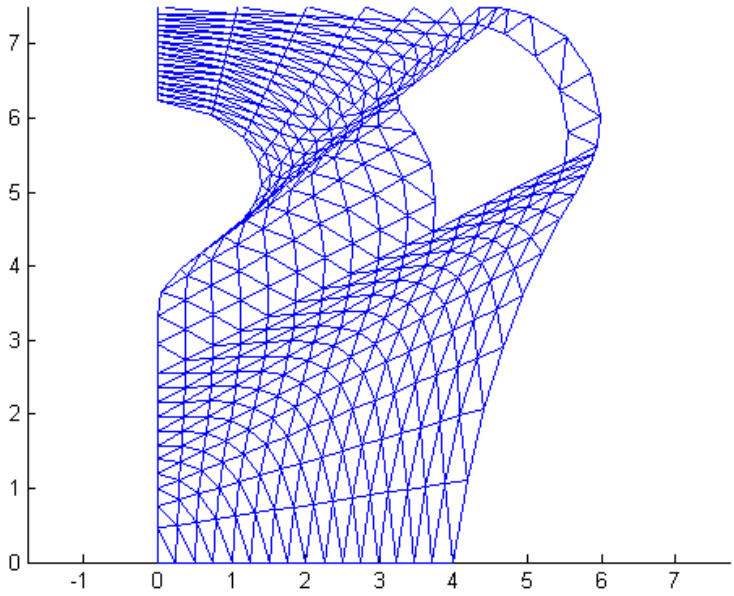


Figure 3.53: Rectangular grid mesh of crossbar

The area between the pin circle and the bounding rectangle was then meshed. In order to ensure that the two mesh regions matched properly, the nodes on the rectangle had to be consistent with the main rectangular grid mesh. The polygon approximating the pin circle was then chosen to have the same number of nodes, so that a 1-1 correspondence between the nodes on the circle and on the rectangle could be formed. A line segment was then computed joining each of these node pairs, and additional nodes placed along it as necessary based on aspect ratio considerations. The final result is shown in Figure 3.54.

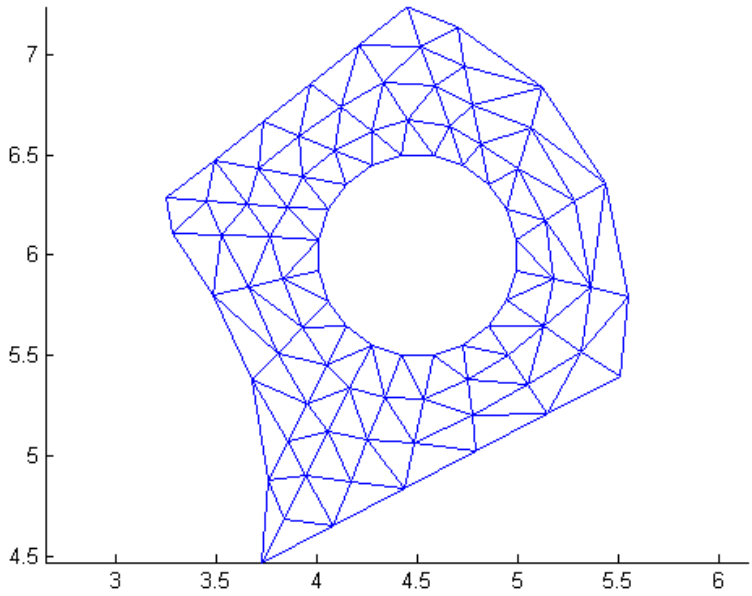


Figure 3.54: Mesh in pin region of crossbar

The entire part has now been meshed, as shown in Figure 3.55.

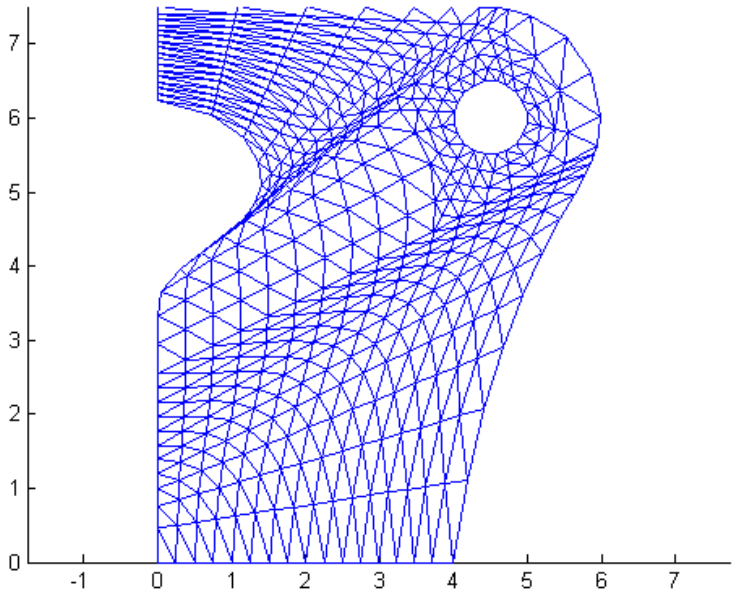


Figure 3.55: Full crossbar mesh

3.7.4 Application of Boundary Conditions

The boundary conditions for the crossbar are fairly straightforward. The bottom surface is a fixed support, meaning that both the x and y displacements of the nodes on that surface are set to zero. The left surface is a symmetry condition, which means that x displacements of those nodes are set to zero, as any nonzero displacement would violate symmetry. The y displacements are unconstrained however. Finally, exactly the same bearing load application method as in the case of the jaw was used to apply the pin force to the appropriate nodes on the pin polygon.

3.7.5 Solution and Post-Processing

With the boundary conditions specified, equation 3.51 is now solved for the system. The same comments on choice of solution method apply as in the case of the jaw. Again, the

elemental stresses are computed from the nodal displacements during post-processing.

3.7.6 Inner Optimization Algorithm

The inner optimization for the crossbar is the same as that for the jaw, with some important differences in the constraint function. As was already discussed, the convexity constraint is not used for the crossbar. However, given the more complex topology of the part due to the presence of the cutout, there is a lot more that can go wrong and must be guarded against. If the perturbation of the CP locations during the optimization process results in either

1. The two B-spline boundaries intersect each other, or

2. One of the B-splines crosses through the pin circle

the topology of the solution domain will be altered. This will invariably result in at best failure of the FEA-based objective function and at worst execution of the objective function call that will produce nonsensical results. Figure 3.56 shows the effect on the mesh and stress solution of a typical case of topological degeneration.

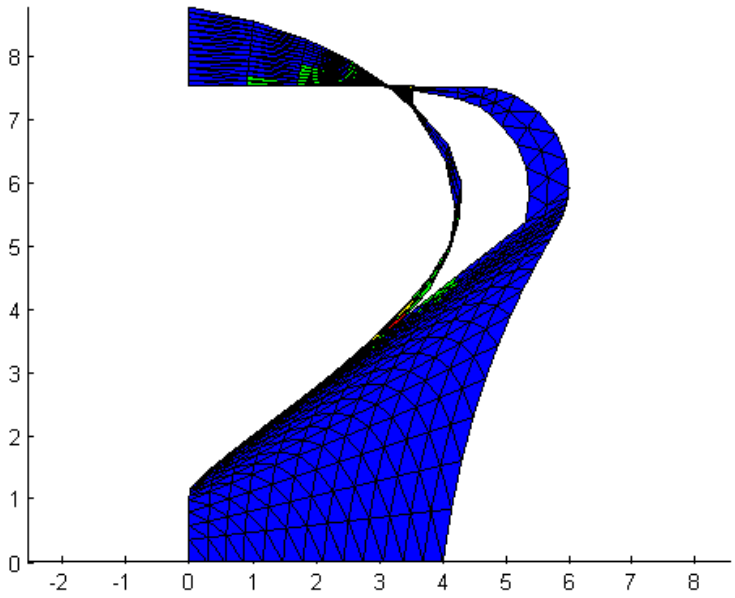


Figure 3.56: Failure due to topological degeneration

Inner/Outer Boundary Intersection Check

An efficient approximate check for the first condition can be performed by connecting each set of CPs in order with line segments, and seeing if these two line segment chains intersect. Checking for any intersection of two line segment chains can be done by checking for intersection of each possible combination of members of the two chains, which since there are only 4 segments in the inner chain and 15 in the outer chain is a quick calculation. The base operation of this algorithm, checking for the intersection of two line segments in the plane, can be done as follows.

If the first line segment is defined by a vector $\vec{R}$ whose base is located relative to the origin by $\vec{P}$, and if similarly the second line segment $\vec{S}$ is located by $\vec{Q}$ (see Figure 3.57), the cross-product c of $\vec{R}$ and $\vec{S}$ is calculated as

$$c = r_x s_y - r_y s_x \tag{3.58}$$

If $c = 0$, the two line segments are either parallel or collinear. These cases can be distinguished by computation of another cross product, $(\vec{Q} - \vec{P}) \times \vec{R}$. If this is also, zero, they are collinear, otherwise parallel.

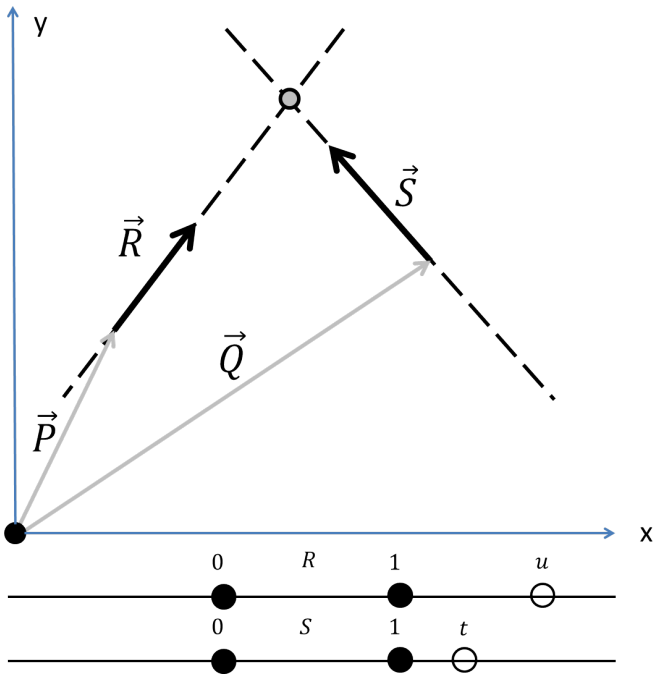


Figure 3.57: Line segment intersection check

If, as is very likely, $c \neq 0$, there must be some point in the plane where the infinite extensions of the line segments intersect. Letting u and t parameterize $\vec{R}$ and $\vec{S}$ in the usual way so as to be zero at the base and one at the tip of the line segment vector, the parametric coordinates of the intersection can be found as

$$\begin{aligned} t &= \frac{(q_x - p_x)s_y - (q_y - p_y)s_x}{c}; \\ u &= \frac{(q_x - p_x)r_y - (q_y - p_y)r_x}{c}; \end{aligned} \tag{3.59}$$

If both u and t are between zero and one, the line segments intersect, otherwise not. Thus the constraint is formulated by "folding" the real number line about the point 0.5 to

make the parameter value more negative as the point of intersection becomes further from the line segment. New variables are thus defined:

$$t_{prox} = \begin{cases} t, & t < 0.5 \\ 1 - t, & t \geq 0.5 \end{cases}$$
$$u_{prox} = \begin{cases} u, & u < 0.5 \\ 1 - u, & u \geq 0.5 \end{cases} \tag{3.60}$$

The value of the constraint for these two line segments is then set to be the smaller of these two values, and the overall value for the entire comparison of the two segment chains is the maximum of the individual values. A constraint violation occurring if this overall value is greater than zero, meaning that at least one pair of line segments intersected. The constraint is formulated in a continuously-valued way rather than a simple Boolean check in order to establish a gradient and drive the constraint towards the feasible region.

It must be noted that ensuring that the CP chains do not intersect is not quite a strong enough condition to guarantee that the B-splines do not intersect, although for well-behaved curve shapes it is a very good approximate check. A few pathological cases may occur where the check fails to detect a curve intersection or detects one where none exists. In the former case, the consequences of topological degeneration are catastrophic enough that they are easily detected after the fact, and the objective function evaluation can be aborted (as long as this is done sparingly it is not a problem). In the latter case, no real harm is done by constraining the curves to be a little farther apart than is absolutely necessary.

Pin Intersection Check

The other possible problem is if one of the B-splines intersects the pin circle. Since a B-spline segment always lies within the convex hull of the four CPs that define it, ensuring that the pin circle does not intersect the convex hull of any four consecutive CPs is sufficient

to ensure that the B-spline does not intersect the pin circle. Considering the quadrilateral formed by the convex hull of four points, there are four distinct modes of intersection, as shown in Figure 3.58.

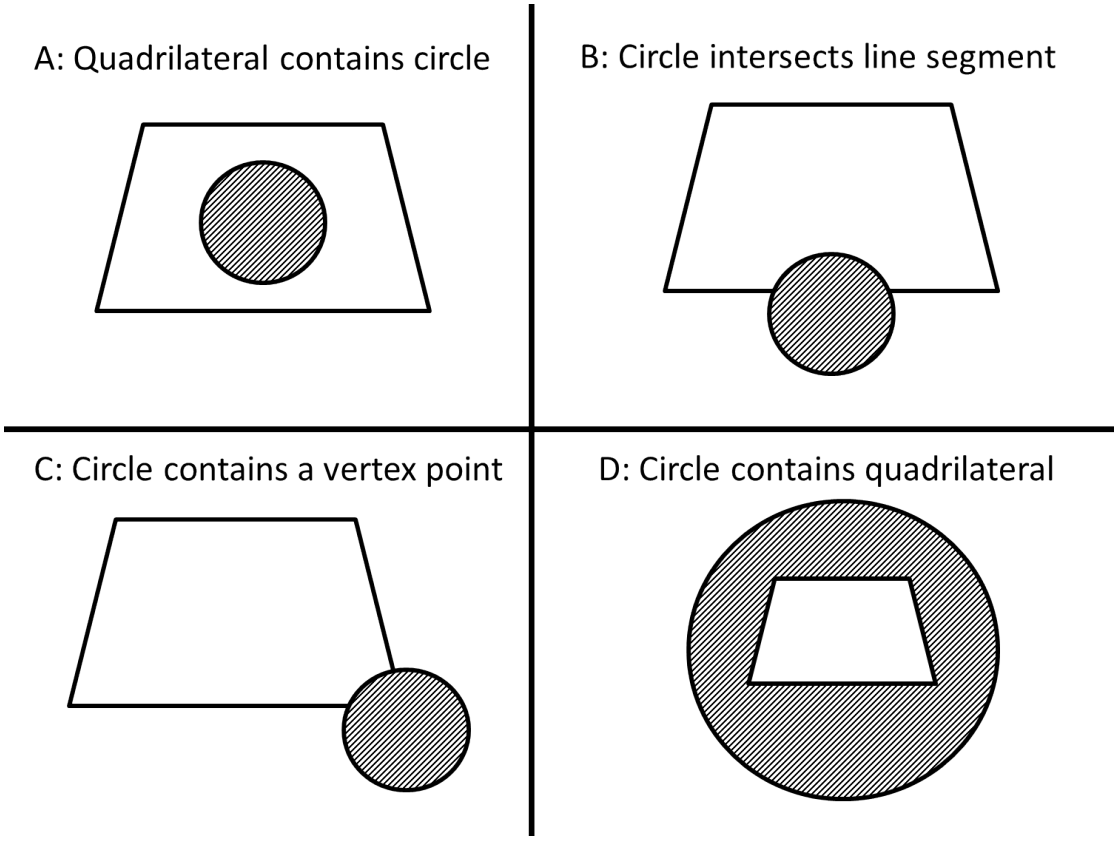


Figure 3.58: Circle/Quadrilateral intersection check

To ensure that none of these modes are present, the following three sub-checks must be made for each quadrilateral:

1. Ensure that the length of perpendicular line joining the circle center and the infinite extension of each of the four line segments defining the quadrilateral is greater than the radius of the circle, OR that the intersection of the perpendicular and the infinite line does not occur on the original segment (see Figure 3.59). Truth of this condition ensures falsity of mode B above.

2. Check that the distance from each vertex point to the circle center is greater than the circle radius. This precludes mode C. Note that sub-check 1 does not necessarily cover this as the case shown in Figure 3.60 is possible in which the perpendiculars fall outside the line segments but the circle contains the vertex point. This check also precludes mode D.

3. Finally, to check for mode A, it suffices to show that the circle center is not contained in the quadrilateral, which can be done with one's choice of well known point-in-polygon techniques [85].

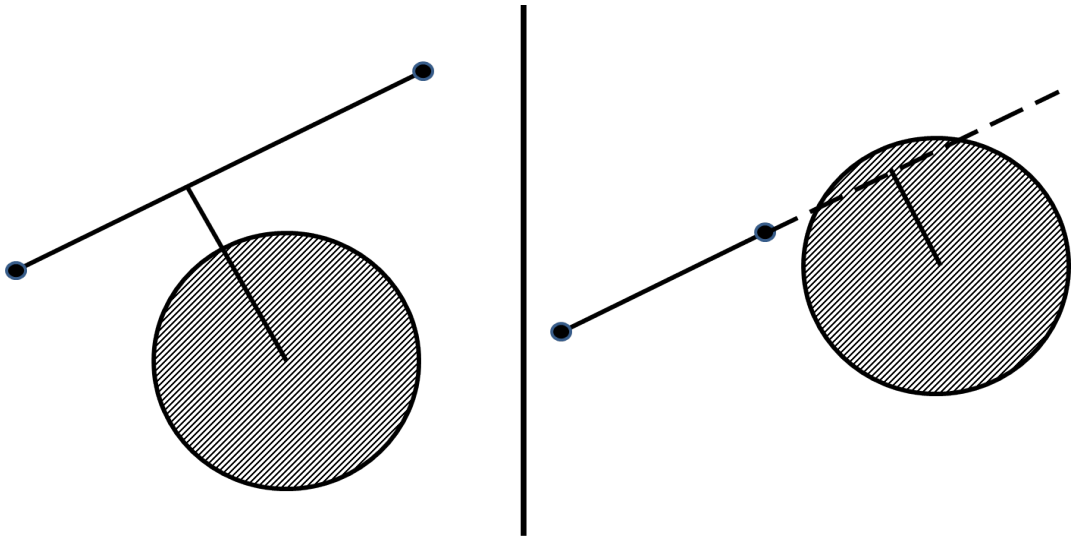


Figure 3.59: Sub-check 1; Either case shown passes

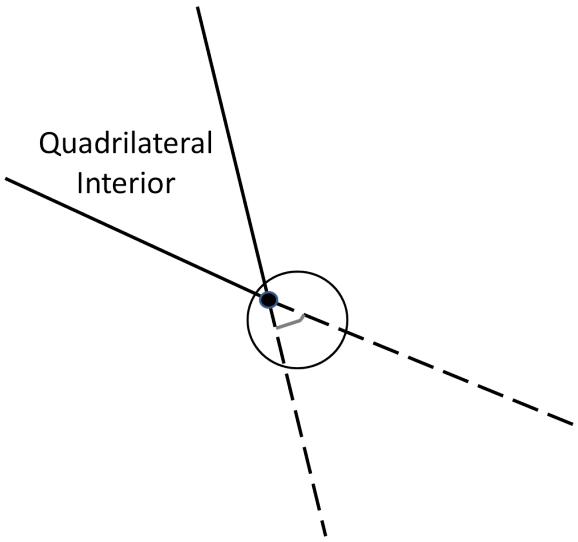


Figure 3.60: Sub-check 2; Case that fails 2 but passes 1 is shown

The constraint value is set to the minimum value out of the center-vertex distances and parametric coordinates (as defined in the B-spline intersection check) generated by any failed sub-checks, so as to create a gradient rather than a simple Boolean condition. The constraint is satisfied if the value is zero or less, which it will be if none of the sub-checks are failed.

3.7.7 Validation of Crossbar Model

As in the case of the jaw, commercial CAD and FEA software was used for comparitive validation. In this case the emphasis was on the accuracy of the FEA solution. In a test case of a slightly earlier crossbar model, the FEA solution for the stress field was almost exactly replicated by solution of the same problem in ANSYS. The problem setup is shown in Figure 3.61, and the stress field comparison is shown in Figure 3.62. Note that the stress contours are set to identical values in Figure 3.62.

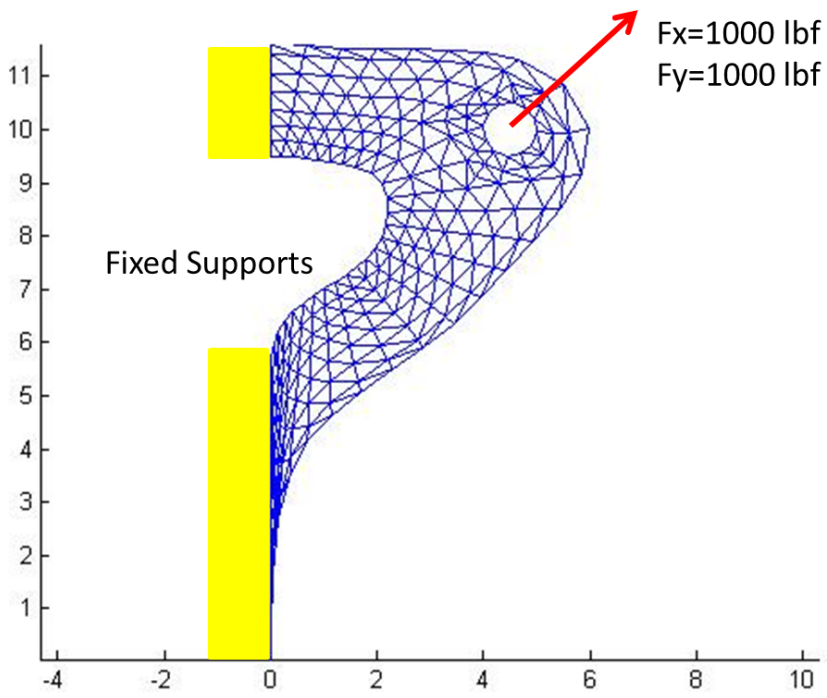


Figure 3.61: Test problem setup

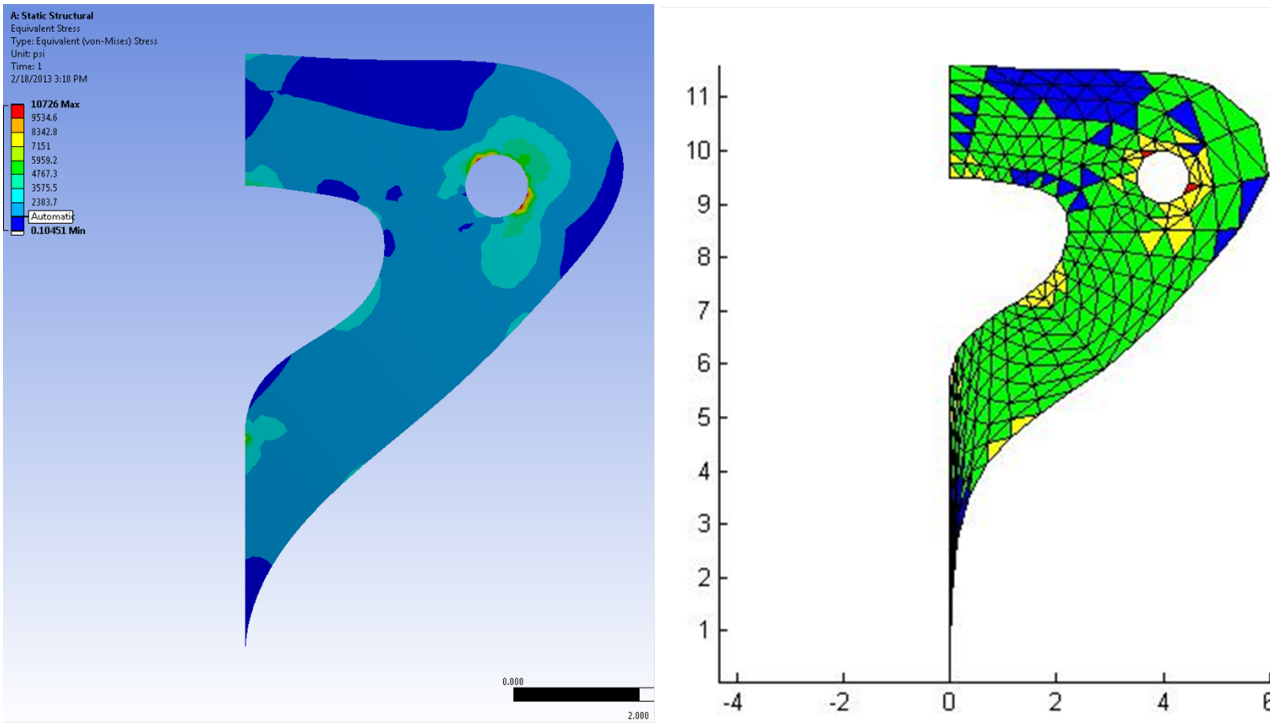


Figure 3.62: Test problem solution

3.8 Optimization Execution and Results

With both sub-optimization components complete, all the link masses can be found and the last objective, the total mass of the mechanism, evaluated. The design of the entire optimization is now complete, and all that remains is to select values of the necessary parameters and begin the computation.

3.8.1 Optimization Parameters

The outer-level optimization algorithm used was an implementation of the NSGA-II algorithm [12] written by the author in Matlab. 50 generations were computed with a population size of 90 individuals, having a 20-digit binary chromosome representation. There were four objectives, as has been previously discussed:

- Minimize mechanism mass
- Minimize mechanism length
- Minimize jaw rotation
- Minimize spreading force variation

The six design variables at the outer level, as defined in Section 3.2.1, were:

$$\{\alpha_i, \gamma_i, \Gamma_f, r_1, g, \theta_{rot}\}$$

where the first three are naturally bounded, and the next two were each given bounds of 0-6 inches, which as intended proved to be much greater than necessary to ensure inclusion of all optima. The final design variable (also an objective), jaw rotation, was bounded between 30 and 60 degrees.

The spreading force and spreading distance were required to be 10,000 lbf and 24 inches, respectively. These values were chosen in order to facilitate direct comparison to the new SP-300 spreader from market leading manufacturer Hurst. All components of the spreader were assigned one of two sets of material properties based on qualitative judgment as to suitability. Components are listed in Table 3.8.1 based on material, with the safety factor used also specified in each case. A base safety factor of 1.5 was used based on the testing specifications in the NFPA Standard on Powered Rescue Tools [86], and increased for components of the actuator as higher safety factors are advisable for pressurized systems.

Table 3.2: Material selections, material properties [87] and safety factors

Part	Material	Yield Stress (psi)	Density (lbm/in^3)	Safety Factor
Jaw Tips	4140 Steel Alloy	140,000	0.28	n/a
Piston Crossbar	4140 Steel Alloy	140,000	0.28	2
Piston Rod	4140 Steel Alloy	140,000	0.28	2
Cylinder	4140 Steel Alloy	140,000	0.28	3
Tie Rods	4140 Steel Alloy	140,000	0.28	2.5
Connecting Links	4140 Steel Alloy	140,000	0.28	1.5
Pins	4140 Steel Alloy	140,000	0.28	1.5
Jaws	7075-T6 Aluminum Alloy	63,000	0.10	1.5
Piston Hub	7075-T6 Aluminum Alloy	63,000	0.10	n/a
Endcaps	7075-T6 Aluminum Alloy	63,000	0.10	3

3.8.2 Visualization of Results

Before proceeding to presentation of the results, a minor digression into visualization methodology is needed. A major challenge when working with more than three objectives is visualization of the objective space, which greatly aids the decision maker in making a final selection from the Pareto optimal set returned by the optimization. A wide variety of methods have been proposed for dealing with this problem, and an adaption of a method proposed in [88] (illustrated in Figure 10 of this reference) is used here. The basic idea is to plot solutions in a regular 2D polygon where the number of vertices is the same as the number of objectives, and each dimension of the objective space is mapped to a line from the center of the polygon to one of the vertices. For example, if there are six objectives, one would take the six orthogonal axes of the (hypothetical, of course) six-dimensional plot of the objective space and put them between the center and vertices of a regular hexagon. Since the axes are no longer orthogonal, information is lost, as is inevitable when repre-

senting high-dimensional data in a 2D plot, but the result can still be illuminating.

The above description is only an approximate qualitative description of the process to aid intuitive understanding of the actual process, which is now described. First, the maximum and minimum value of each objective out of all the solutions returned by the optimization is identified, and all objective values are mapped onto [0,1] intervals where 0 is the minimum value and 1 is the maximum. This normalization process ensures that arbitrary differences between objective magnitude scales do not obscure the visualization result. These normalized objective values are then used as the barycentric coordinates of the solution in the polygonal space, and are plotted via convex combination of the vertex coordinates in the plane. A convex combination is a means of calculating the location of a point based on the weighted average of the locations of a number of other points, with the requirement that the sum of the weights equal one:

$$P = \sum_{i=1}^{N} W_i V_i, \qquad \sum_i W_i = 1 \tag{3.61}$$

where P is the output point, V_i are the positions of the N reference points, and W_i are the weights. If all weights are nonnegative, P will always land inside the convex hull of the reference points. Convex combination can be thought of as a generalization of linear interpolation between two points, which it in fact reduces to if $N = 2$.

In this case, the reference points are the vertices of a regular polygon, and the weights are obtained by dividing each normalized objective value by the sum of all normalized objective values for the given solution, to ensure that $\sum_i W_i = 1$ is satisfied. Nonnegativity is guaranteed by all the normalized objective values having been mapped to the interval [0,1] to begin with. After this process is completed, the location of each solution in the polygon will reflect the relative quality of its objective values. Thus, a solution that falls into the 37th percentile (for instance) in every objective, compared to the rest of the solution set, will end up in the exact middle of the polygon, whereas a solution that is best in objective 2 but worst in all the others will end up on top of the second vertex of the polygon. More

generally, the worse a solution does at objective j, the more it is "pulled" towards vertex j.

A problem with this is that the overall quality is obscured; a solution that is worst at all objectives and one that is best at all objectives will both end up in the middle of the polygon. Thus a color scaling is introduced, where "bluer" solutions have a smaller (i.e. better) sum of normalized objective values and "redder" solutions have a larger sum. Pure blue is a sum of zero (best at all objectives) and pure red is a sum of N (worst at all objectives). Note that this scaling does not preserve Pareto optimality, as it effectively maps solutions onto an additional dimension by collapsing all objectives to a single value using an unweighted sum. While imperfect, the combination of plotting by convex combination and color scaling is a reasonable approach to visualization.

3.8.3 Results

The optimization took about 96 hours on a desktop computer with an Intel i5 processor. Each generation took about 2 hours, each primary objective function call (evalutation of an individual candidate mechanism) slightly over a minute, and each secondary objective function call (FEA analysis on one of the parts in 5 positions) slightly over a second on average. FEA calculations were responsible for the bulk of the required solution time.

A polygon plot of the type just described is given in Figure 3.63 (the color scale is that of Figure 3.38) for the set of 90 solutions after the final generation of the algorithm completed. Final selection of a single solution was made by use of a supplementary code that returned subsets of the frontier that satisfied specified maxima of all four objectives. A simple manual iteration process of tightening the requirements and inspecting the resulting lists was employed using this tool.

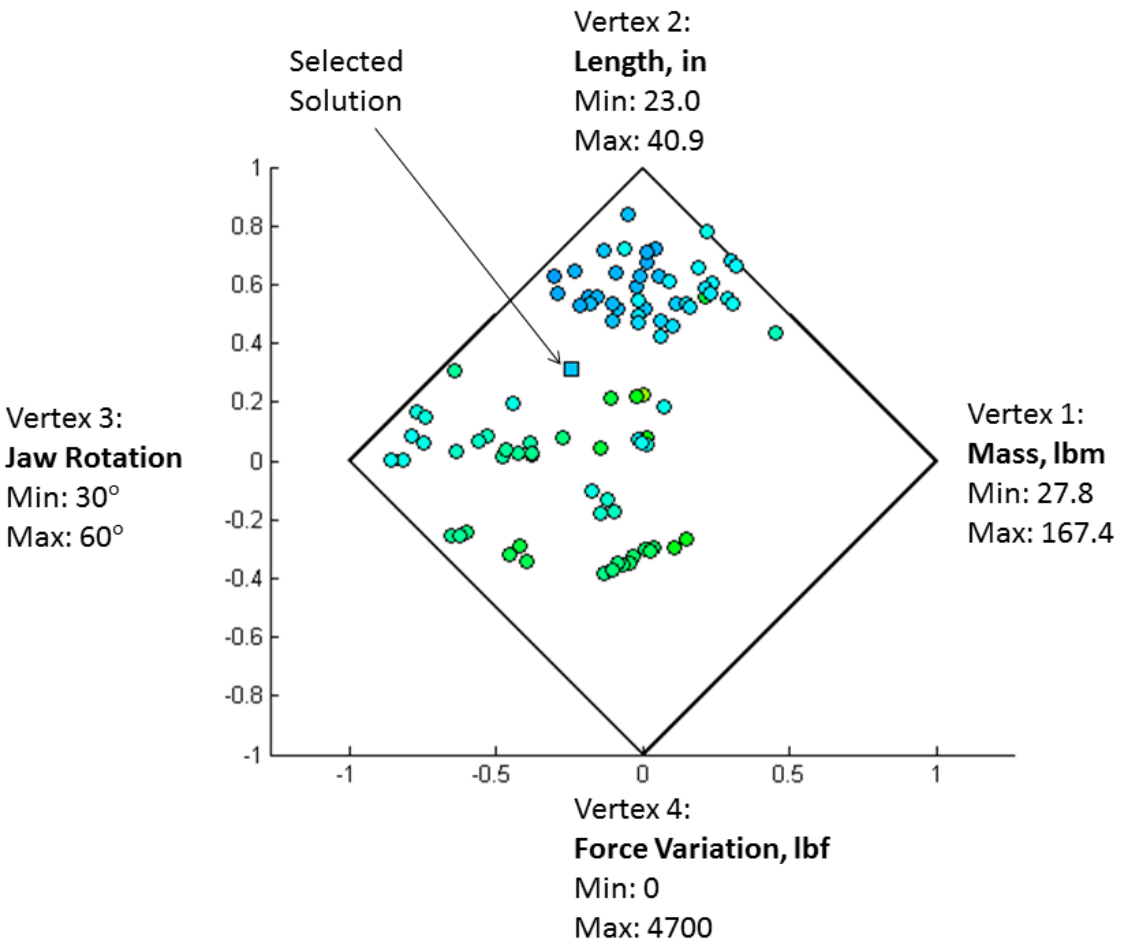


Figure 3.63: Polygon plot of spreader solutions

Also useful was another code to generate graphical depictions of a candidate linkage given the design variable values, by a combination of simple geometric objects for the lumped components and renderings of the stressed FEA meshes subjected to appropriate rotations and translations. Such a rendering of the final result is shown in Figure 3.64. All colors use the color mapping of Figure 3.38, and the renderings are to scale with the axes in units of inches.

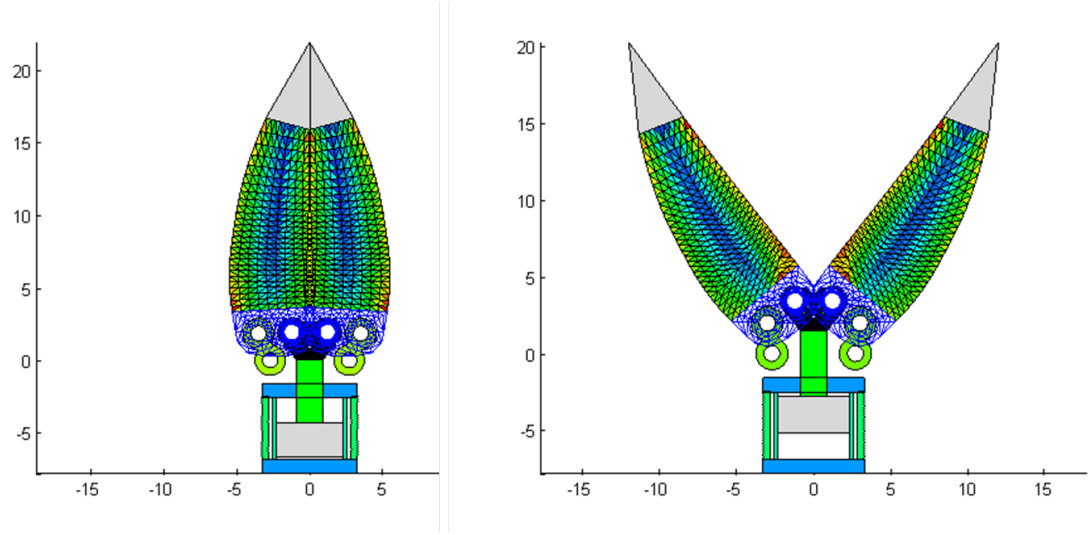


Figure 3.64: Optimized mechanism

The optimized values of the design variables as well as the bounds on the search space are listed below. The fact that all the values found are well in the interior of the search space supports the conclusion that the optimum does not lie outside of it.

Table 3.3: Bounds and optimized values of design variables

Variable	Min	Max	**Optimal**
α_i (degrees)	60	150	128.85
γ_i (degrees)	30	150	59.75
Γ_f (unitless)	0	1	0.57
r_1 (inches)	0	6	1.48
g (inches)	0	6	1.23
θ_{rot} (degrees)	30	60	36.25

This design compares favorably to the Hurst SP-300 that was selected as the competitor for the optimization parameter specification, as detailed in Table 3.1. LSF and HSF stand for “Lowest Spreading Force” and “Highest Spreading Force” respectively, and occur in the

closed and open positions respectively. These forces are measured in a manner regulated by the relevant NFPA standard [86] and are among the primary figures of merit in evaluating spreader performance.

Table 3.4: Comparison of objectives values between result and Hurst SP-300 model [89]

	Optimized	Hurst
Mass - lbs	31.08	37.70
Length - inches	29.96	29.53
Jaw Rotation - deg	36.25	54.00
LSF - lbf	10,000	7,419
HSF - lbf	10,704	8,992

A side-by-side graphical comparison is shown in Figure 3.65.

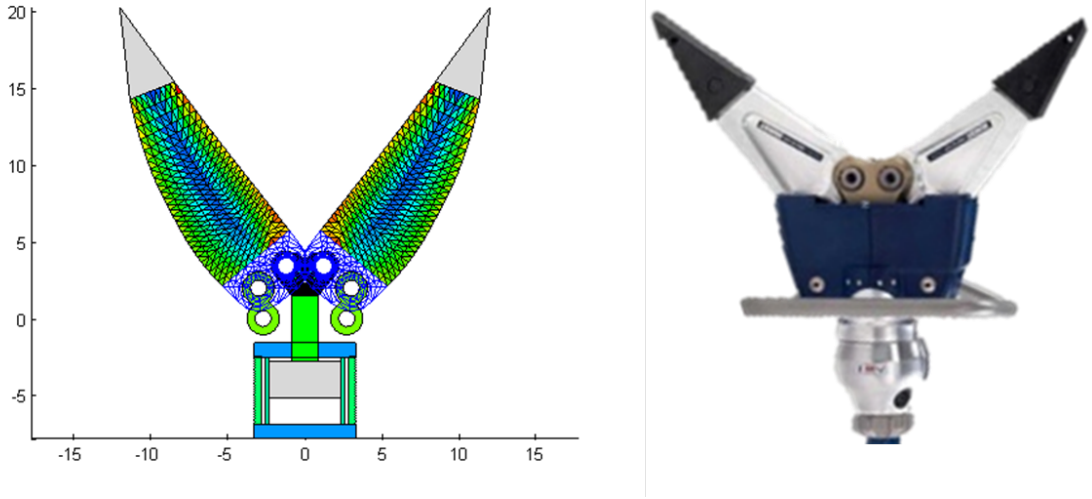


Figure 3.65: Graphical comparison of result to Hurst SP-300 model [89]

It should be noted that the mass estimate is likely over-optimistic, with the true mass of the design probably being about equal to that of the SP-300, due to some of the simplifying assumptions that were made. However, the force and jaw rotation objectives are exactly

correct as only vector geometry is involved, and the length objective is almost certainly very reliable as well.

There is a significant degree of qualitative resemblance between the optimized and Hurst spreaders, which is quite remarkable as the former is the product of a sophisticated numerical optimization written on general principles without significant prior experience in rescue spreader design, and the latter is the product of 40 years of manual design iteration by the industry-leading company. Note for example the fact that the "blind" process of application of a gradient-descent inner optimization to B-spline control points came up with almost exactly the same shape for the crossbar as Hurst's designers. This is a very strong indication that the optimization techniques presented in this thesis have merit. Furthermore, while the optimization more or less broke even with Hurst with respect to mechanism size and mass, the kinematic objective values are clearly superior. The optimized spreader accomplishes the same task with significantly less jaw rotation and increased spreading force magnitude and consistency.

Chapter 4:

Second Case Study: Internal Combustion Engine

One of the oldest problems in mechanical engineering is the conversion of linear to rotary motion in engines. This problem is of course long since solved in the modern internal combustion (IC) engine by the use of the simple slider-crank four-bar linkage. A fundamental limitation of this solution is that the relationship between crank angle and piston position is quite constrained, with piston position being restricted to an approximately sinusoidal function of crank angle. The progress of the combustion reaction depends in part on the cylinder volume as a function of time, and there is no reason to believe that a sinusoidal variation is optimal in terms of maximizing efficiency and minimizing emissions. Thus a need exists for an IC engine linkage design that will allow greater design freedom in specification of the piston position as a function of crank angle.

A suitable choice is found in the Stephenson-III six-bar linkage. The base four-bar is required to be a Grashof crank-rocker, with the crankshaft of the engine located at the crank ground pivot. The dyad is converted to a connecting rod and slider, with the slider being the piston. The linkage is illustrated in Figure 4.1. It should be noted that, as was the case in the rescue spreader problem, topological synthesis is not included in the optimization

process. This is again due to the fact that the problem is being approached as a case study of the method of Chapter 2, which does not deal with topological synthesis and assumes that this task was completed separately beforehand.

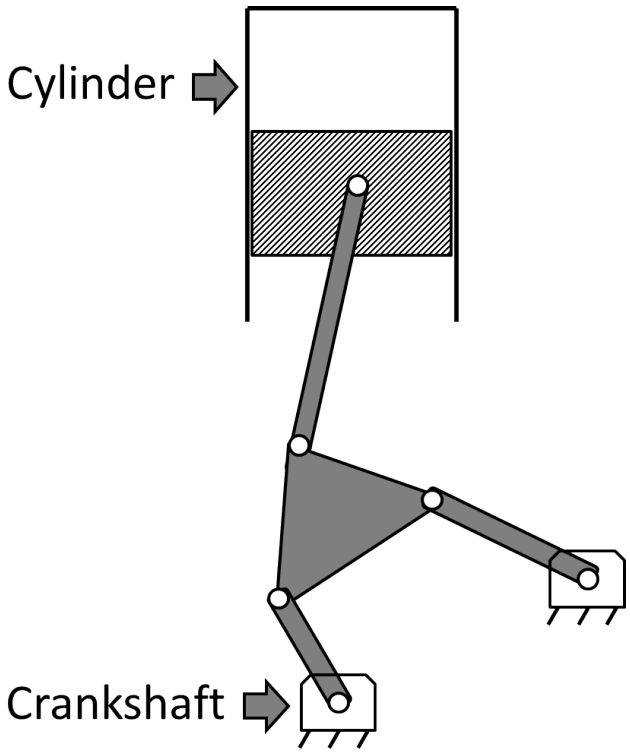


Figure 4.1: Stephenson-III linkage for an internal combustion engine

4.1 Design Considerations and Objectives

With the general goal and a suitable linkage topology identified, the task of determining the linkage dimensions and link designs is undertaken using a multi-domain optimization according to the general methodology of Chapter 2. Varying the linkage dimensions will result in different piston position functions, which can be converted to efficiency and emissions objective values by the use of a combustion simulation subroutine. In addition to the original goal of improving performance in these respects, it is also necessary to ensure that the engine design is workable from a mechanical point of view. The overall size and mass

of the engine must be within acceptable limits for inclusion in a vehicle, and the shaking force and moment must be kept reasonable. Finally, the side-loading on the piston must not be too great. In light of these considerations, the following objectives are defined:

Kinematic Objectives

- Maximize efficiency
- Minimize side-loading of piston

Non-kinematic Objectives

- Minimize RMS shaking force
- Minimize RMS shaking moment
- Minimize total mass
- Minimize planar footprint

4.2 Preliminary Kinematic Analysis

In keeping with the general procedure, a preliminary kinematic analysis is performed with two goals in mind:

1. Determine the best way to formulate design variables so as to reduce their number, or at least determine non-arbitrary bounds on as many as possible
2. Rectification of solutions: try to guarantee that any combination of design variables will result in a valid linkage

4.2.1 Naive Formulation

A naive formulation of the design variables would probably be:

$\{r_1, r_2, r_3, r_4, r_5, r_6, \theta_1, \theta_{35}, X\}$, where X is the distance from the origin (R_2 ground pivot) to the vertical axis of slide, and the remaining quantities are as shown in Figure 4.2.

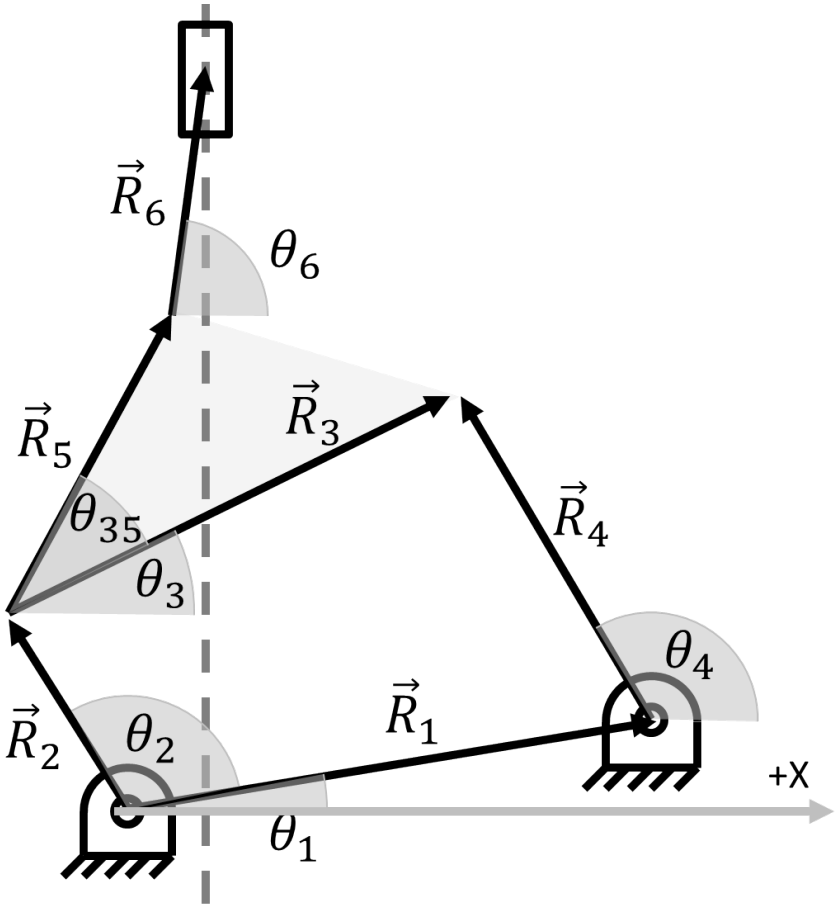


Figure 4.2: Linkage Diagram

Of these 9 variables, the 6 r_i have natural lower bounds of zero, but no upper bounds. The two angles are naturally bounded above and below due to the periodicity of the definition of an angle. X is completely unbounded. Any solution formulated this way will have to be checked for three things:

1. Make sure that the base four-bar is a Grashof crank-rocker (rather improbable for

random link lengths)

2. Make sure that the R_3/R_4 transmission angle is acceptable

3. Make sure the connecting rod R_6 can reach the axis of slide at an acceptable angle from all points on the coupler curve

4.2.2 Better Formulation

A better formulation of the design variables is:
$\{t_1, n_3, n_4, n_5, m_6, \theta_1, \theta_{35}, t_s\}$, where in general, t is an index of interpolation, m is a scaling, and n is a normalized length. The best way to explain the meaning of these variables is by explaining how the actual linkage dimensions in the naive formulation are recovered.

The first thing to notice is that the strategy of normalizing all lengths by r_2 has been adopted, resulting in the elimination of r_2 from the list of design variables and the replacement of r_i by $n_i = r_i/r_2$. Next the base four-bar is examined, starting with the determination of the minimum and maximum values of n_1 that will result in a Grashof crank-slider with acceptable transmission angles of at least θ_{min}. As illustrated in Figure 4.3, the conditions are

$$\begin{aligned} n_{1,min} &= \sqrt{n_3^2 + n_4^2 - 2n_3n_4\ cos(\theta_{min})} + n_2 \\ n_{1,max} &= \sqrt{n_3^2 + n_4^2 - 2n_3n_4\ cos(\pi - \theta_{min})} - n_2 \end{aligned} \tag{4.1}$$

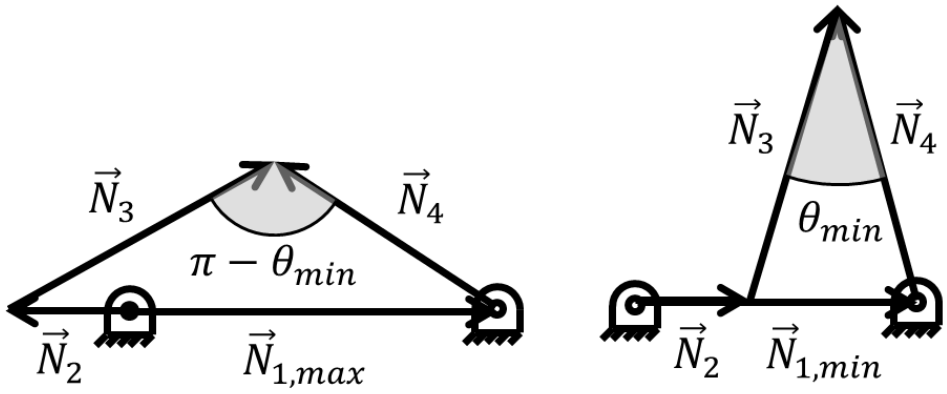


Figure 4.3: Min/Max R1

In this way, given a specified θ_{min}, selected values of n_3 and n_4 greater than 1, and $n_2 = 1$ by definition, bounds on n_1 are obtained. The interpolating index $t_1 \in [0, 1]$ is then used to select a specific value for n_1 from this range, using

$$n_1 = (1 - t_1)n_{1,min} + (t_1)n_{1,max} \tag{4.2}$$

The coupler dimensions n_5 and θ_{35} are then chosen directly.

Next, the four-bar motion is analyzed at a number of discrete positions and the positions, velocities, and accelerations of each joint are computed. Additionally, the position data provides an approximation of the coupler curve, and the min and max x-position reached by the coupler point are determined. It is now time to look at the connecting rod and slider, which so far have been ignored. There exists an implicit lower bound on the possible length of the connecting rod; with the coupler curve having a known, finite width

in the x direction, it is evident that for a small enough connecting rod length there will not exist any possible axis of slide that can be reached from all points on the coupler curve. If some minimum angle θ_s is required between the connecting rod and horizontal for transmission purposes, this constraint becomes even more stringent.

As shown in Figure 4.4, putting the axis of slide halfway between the x-extremes of the coupler motion enables the smallest possible value of connecting rod length, which will be given by

$$n_{6,min} = \frac{X_2 - X_1}{2\ cos\ \theta_s} \tag{4.3}$$

where the variables are as represented in Figure 4.4.

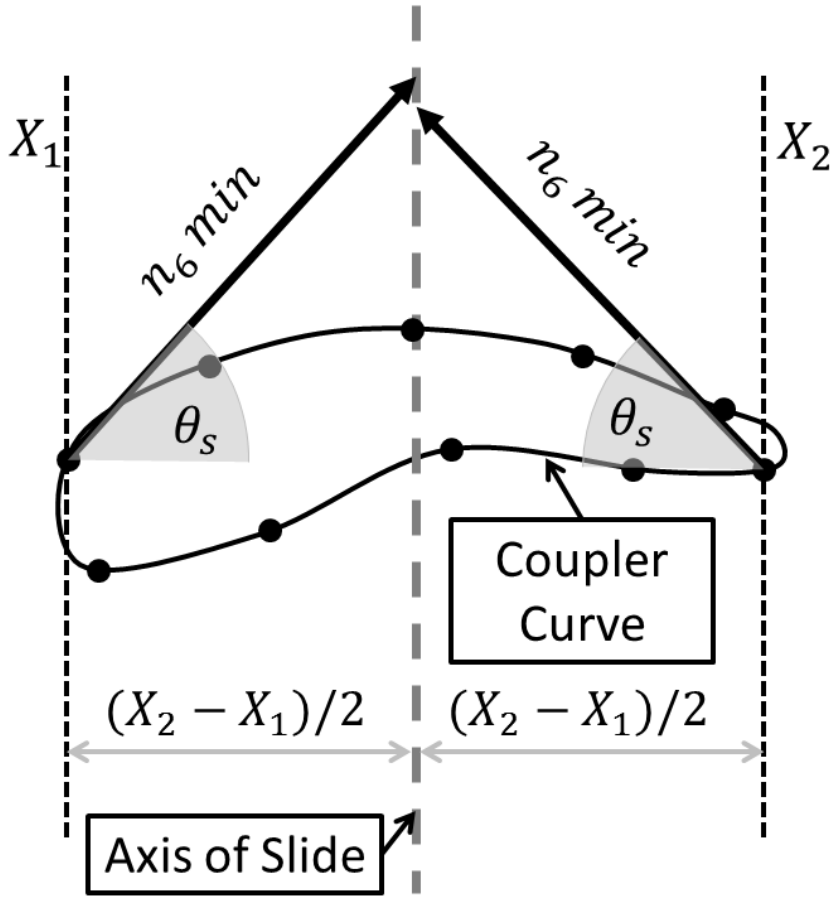


Figure 4.4: Determination of minimum connecting rod length

The actual value of n_6 is then chosen based on the value of the design variable $m_6 \geq 1$, using

$$n_6 = m_6(n_{6,min}) \tag{4.4}$$

or in other words the value of n_6 is found by multiplying the minimum allowable value by 1 or more.

If $m_6 > 1$, the axis of slide has some finite range of possible locations, and can be moved left or right from the center position as long as the connecting rod can still reach from the side of the coupler curve that the axis of slide has been moved away from, as illustrated in Figure 4.5. Specifically, if X_0 is the central position of the axis of slide as shown in Figure 4.4, the left and right limits X_{min} and X_{max} are given by

$$X_{min} = X_0 - (n_6 - n_{6,min})(\ cos\ \theta_s) \tag{4.5}$$

$$X_{max} = X_0 + (n_6 - n_{6,min})(\ cos\ \theta_s) \tag{4.6}$$

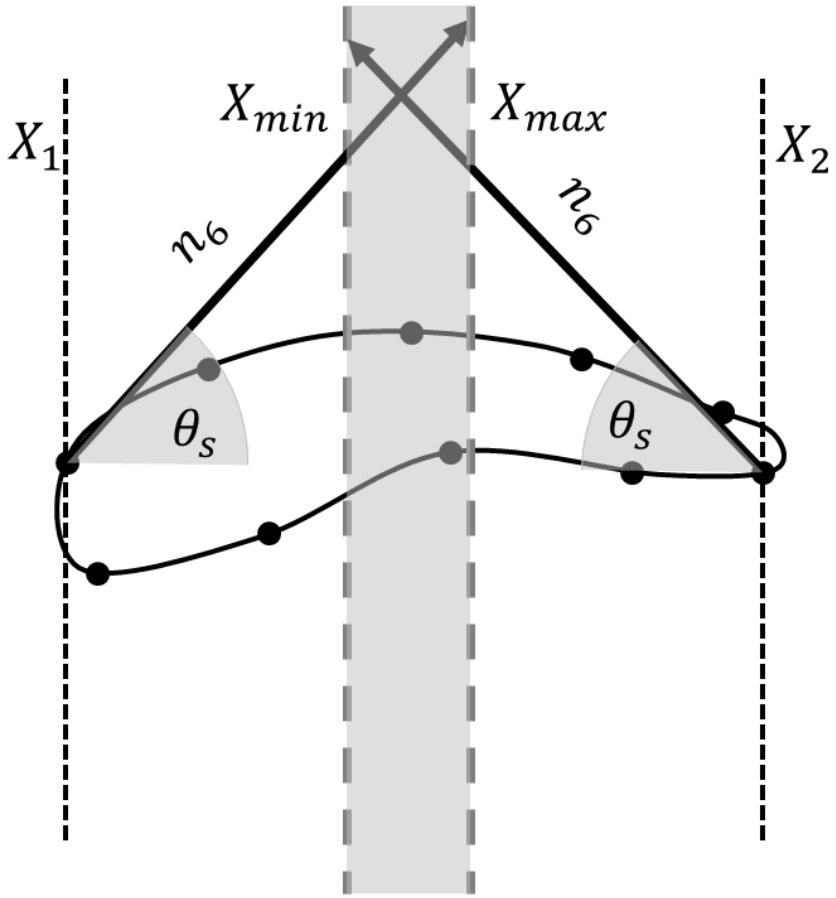


Figure 4.5: Allowable axis of slide locations

Once the limits are known, the interpolation index $t_s \in [0, 1]$ is applied to locate the axis of slide somewhere between them, using

$$X = (1 - t_s)X_{min} + (t_s)X_{max} \tag{4.7}$$

Next, a position, velocity, and acceleration analysis is done on the connecting rod and slider, using the previously computed four-bar data to avoid repeating the four-bar calculations. The position analysis includes the limits of motion reached by the slider along the axis of slide. This distance is then compared to the desired distance, and the linkage scaled accordingly. Recovery of the explicit design variables $\{r_1, r_2, r_3, r_4, r_5, r_6, \theta_1, \theta_{35}, X\}$ is

now complete. What has been gained? One dimension of the space has been eliminated by normalizing the linkage at the beginning, and instead of the unbounded variables r_1 and X the fully bounded interpolation indices t_1 and t_s are used. Furthermore, the three solution rectification conditions

1. Grashof crank-rocker
2. R_3/R_4 transmission angle
3. Connecting rod reaches axis of slide at acceptable angle

are all automatically satisfied for any combination of design variables. Thus it should never be necessary to discard a candidate linkage as physically unworkable, greatly improving optimization stability and convergence. Also note that only a few trivial algebraic computations are added, as the computation of the coupler curve and axis of slide limits are piggy-backed on the position/velocity/acceleration analysis, which had to be done anyway.

4.3 Primary Objective Function

A primary objective function call begins with the recovery of the actual linkage dimensions from the design variables in the manner just described. The next step is a dynamic analysis. There are two major simplifying assumptions used. The first is that the rotational inertia of the mechanism is large enough that the crank can be assumed to have constant angular velocity, even given a variable applied gas force. This allows a kinetostatic analysis rather than a full time response treatment. Second, the dynamic forces are assumed to be neglible compared to the structural forces due to the gas loading, which allows direct solution of the structural properties of the linkage rather than an expensive iterative solution. The validity of this assumption is discussed in Section 4.5.2.

4.3.1 Position, Velocity, and Acceleration Analysis

The analysis begins by solving for the linkage positions as a function of the crank angle θ_2. The linkage can be characterized by three vector loop equations, of which two are linearly independent.

$$\begin{aligned} r_2 e^{i\theta_2} + r_3 e^{i\theta_3} - r_4 e^{i\theta_4} - r_1 e^{i\theta_1} &= 0 \\ r_2 e^{i\theta_2} + r_5 e^{i\theta_5} + r_6 e^{i\theta_6} - R_{sx} - iR_{sy} &= 0 \\ r_2 e^{i\theta_2} + r_5 e^{i\theta_5} - R_{px} - iR_{py} &= 0 \end{aligned} \tag{4.8}$$

where R_s is the position of the slider and R_p is the position of the coupler point, i.e. the coupler/connecting rod joint, relative to the origin. This notation was shown in Figure 4.2; additionally, the tip of R_p traces the coupler curve in Figures 4.4 and 4.5. The position analysis is performed using the well-known Freudenstein equations [90] for the base four-bar, with the slider then solved using

$$\theta_6 = cos^{-1}\Big(\frac{R_{sx} - r_2 cos(\theta_2) - r_5 cos(\theta_5)}{r_6}\Big); \tag{4.9}$$

followed by application of the second vector loop equation to find R_{sy}.

With the position analysis complete, the three loop equations are differentiated with respect to time in order to allow a velocity analysis, yielding

$$\begin{aligned} i\dot{\theta}_2 r_2 e^{i\theta_2} + i\dot{\theta}_3 r_3 e^{i\theta_3} - i\dot{\theta}_4 r_4 e^{i\theta_4} &= 0 \\ i\dot{\theta}_2 r_2 e^{i\theta_2} + i\dot{\theta}_5 r_5 e^{i\theta_5} + i\dot{\theta}_6 r_6 e^{i\theta_6} - i\dot{R_{sy}} &= 0 \\ i\dot{\theta}_2 r_2 e^{i\theta_2} + i\dot{\theta}_5 r_5 e^{i\theta_5} - \dot{R_{px}} - i\dot{R_{py}} &= 0 \end{aligned} \tag{4.10}$$

Splitting into real and complex parts gives six scalar equations; these are then solved for

the six unknowns: $\{\dot{\theta}_2, \dot{\theta}_4, \dot{\theta}_6, \dot{R_{sxy}}, \dot{R_{spx}}, \dot{R_{py}}\}$ using the results from the position analysis.

Similarly, once this is completed the loop equations are again differentiated for an acceleration analysis.

$$\begin{aligned} -\dot{\theta_2}^2 r_2 e^{i\theta_2} + i\ddot{\theta_3} r_3 e^{i\theta_3} - \dot{\theta_3}^2 r_3 e^{i\theta_3} - i\ddot{\theta_4} r_4 e^{i\theta_4} + \dot{\theta_4} r_4 e^{i\theta_4} = 0 \\ -\dot{\theta_2}^2 r_2 e^{i\theta_2} + i\ddot{\theta_3} r_5 e^{i\theta_3 5+\theta_5} - \dot{\theta_3}^2 r_5 e^{i\theta_3 5+\theta_5} + i\ddot{\theta_6} r_6 e^{i\theta_6} - \dot{\theta_6}^2 r_6 e^{i\theta_6} - i\ddot{R_{sy}} = 0 \\ -\dot{\theta_2}^2 r_2 e^{i\theta_2} + i\ddot{\theta_3} r_5 e^{i\theta_3 5+\theta_5} - \dot{\theta_3}^2 r_5 e^{i\theta_3 5+\theta_5} - \ddot{R_{px}} - i\ddot{R_{py}} = 0 \end{aligned} \tag{4.11}$$

which are split into scalar equations and solved using the results from the velocity analysis. In this way the positions, velocities, and accelerations of each link are solved; the analysis is repeated at 20 discrete positions at evenly spaced crank angles.

The linkage is then scaled to recover the actual dimensions from the normalized dimensions. Lengths as well as linear velocities and accelerations are multiplied by a scale factor; angular quantities are the same for all scalings. The scale factor is determined by comparing the normalized piston stroke to the prescribed actual piston stroke.

The position, velocity, and acceleration of each joint in the linkage over a complete cycle is now fully determined, and simulation of the combustion process is now possible. Under the kinetostatic assumption, it is as though an infinitely large flywheel were attached to the crankshaft and spun at a constant angular velocity, driving the piston through a corresponding motion. Thus the true interdependence of the piston motion and combustion process is replaced by a simplification in which there is a one-way dependence of the combustion on the motion, in order to simplify the analysis.

4.3.2 Combustion Simulation

The combustion simulation subroutine uses a simple zero-dimensional or lumped model with respect to spatial treatment of the cylinder volume, but does include the time dimension via parameterization by crank angle. The subroutine was not written by the author

but was provided by a group working on the combustion-related aspects of the problem. A qualitative description of the model is now given.

The combustion model accepts four inputs from the primary objective function: the cylinder bore diameter, the length of the piston stroke, the spark timing, and a vector of 20 piston positions above BDC at equally spaced crank angles (and hence equally spaced times given constant crank rotation speed). Additionally, the following fixed parameters are specified:

- Fuel: Iso-octane
- Equivalence Ratio: 1
- Initial Temperature: 298 K
- Initial Pressure: 101 kPa
- Compression Ratio: 9
- RPM: 2000
- Coolant Temperature: 358 K

The bore and stroke were both fixed at 3.5 inches.

This information is used to simulate the compression and power strokes. A closed system thermodynamic analysis determines the pressure and temperature at every crank angle given a volume profile (determined by the position profile, bore, and stroke), and given the initial temperature and pressure. The amount of air initially present in the cylinder is found from the ideal gas law, and the amount of fuel then follows from the equivalence ratio. With the initial conditions determined, the simulation then steps forward in time at a rate such that the crank moves one degree between time steps. Since this entails analysis of 360 positions, but only 20 kinematic positions are solved for, a PCHIP (Piecewise Cubic

Hermite Interpolating Polynomial) spline is fit to the 20 supplied positions and used to find the piston positions by interpolation.

At each time step, the temperature is found from the first law of thermodynamics, taking into account heat lost and expansion/compression work done. This allows the pressure to be determined using the ideal gas law. The heat added from combustion is found using the Wiebe function, and the heat lost using the Woschni correlation. After the simulation is complete and the pressure and volume histories are known, the work done and hence the thermal efficiency can be post-processed (mechanical friction is neglected). As the cylinder bore is known, the pressure history can be converted to a gas force at each of the 20 kinematic positions. This gas force profile as well as the thermal efficiency are the outputs from the subroutine. Figure 4.6 shows a plot of piston position and gas force for a test mechanism with near-sinusoidal piston motion used to help verify correct functioning of the model, resulting in curves familiar from traditional slider-crank engines. Note that as a result of the low resolution of the kinematics relative to the combustion simulation, the peak gas force is "clipped" and ignition occurs between kinematic positions.

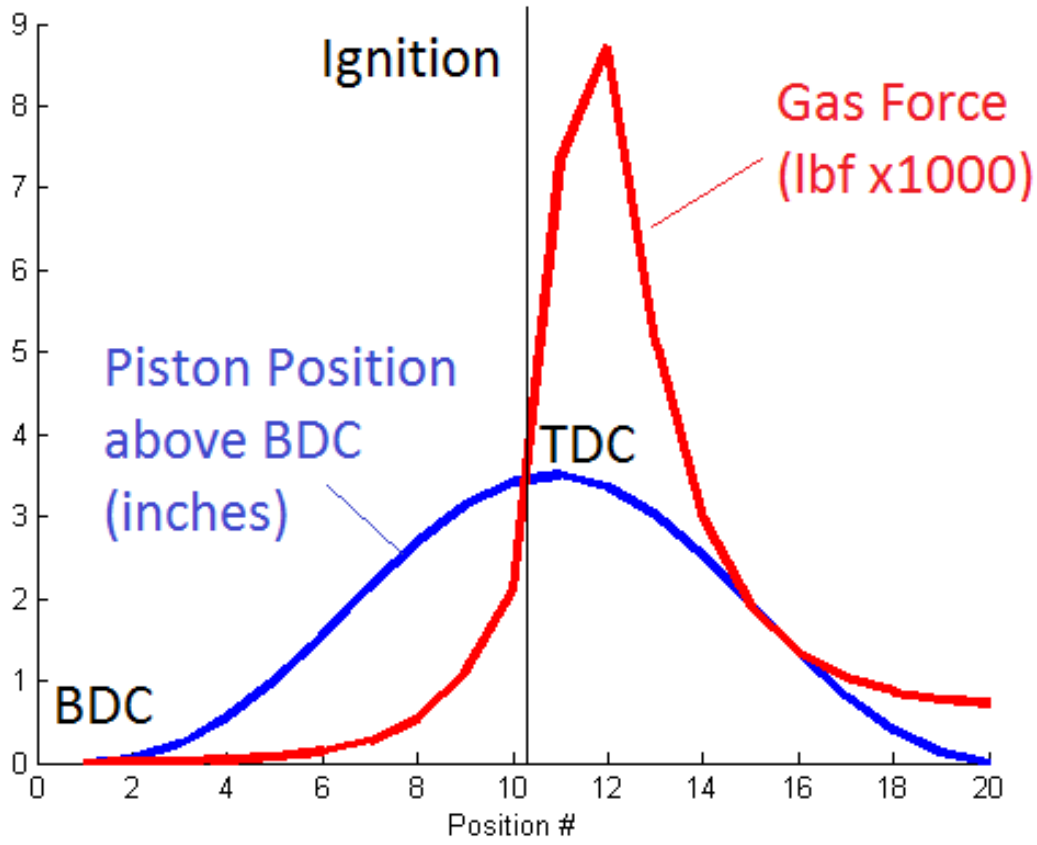


Figure 4.6: Sample piston position and gas force plot

The optimal spark timing will be different for each candidate mechanism, so a mini-optimization is performed at each call to the combustion subroutine. The combustion simulation is run assuming ignition at TDC, and then a simple gradient-descent approach is used to move the ignition timing to a nearby local (and usually global) optimum.

This process is relatively expensive, taking on the order of a full second to run. However, since it only depends on the mechanism kinematics it can be run at the outer level only, and thus does not significantly add to the run time of the optimization as a whole. This is a major benefit of the nested optimization structure; the combustion results can be reused at every inner level objective function call associated with the outer-level candidate solution. If a single-level optimization were used, the combustion would have to be simulated for

each candidate, greatly reducing overall computational feasibility.

4.3.3 Applied Force Balance

With the combustion simulation complete and the gas force profile as well as the linkage dimensions known, it is possible to perform a rigid-body force balance at each position to determine the load supported by each link. The force balance is shown graphically in Figure 4.7.

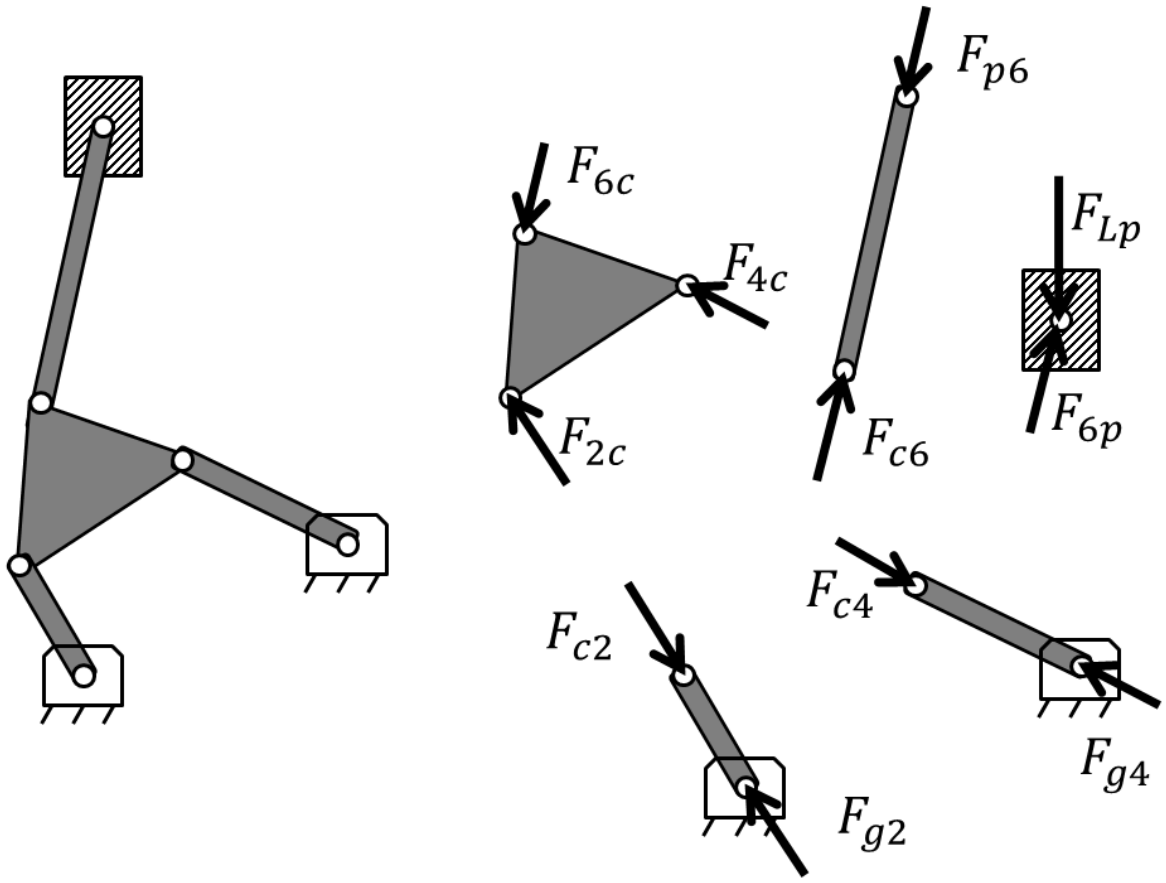


Figure 4.7: Force balance on linkage

The forces can be solved for sequentially, which is more efficient than a simultaneous matrix solution. This is done as follows (steps consisting of finding forces from known equal and opposite forces are omitted due to triviality). It is known that the only external

applied force (excepting the reaction forces at the ground pivots) is the vertical gas force on the piston F_{Lp}. Starting with the piston force balance, it is evident that $F_{6py} = -FLp$. Since the connecting rod is a two-force member, the x component can then be found from the angle of the link, $F_{6px} = \frac{R_{6x}}{R_{6y}} F_{6py}$. The maximum absolute value assumed by F_{6px} constitutes the value of the piston side-loading objective. Knowing F_{6p} gives F_{6c} trivially. It is now necessary to solve the force and moment balance on the coupler. All link vectors R_i are known from the position analysis, as well as F_{6c} from the above, with F_{2c} and F_{4c} unknown.

The two unknown forces each have two components, making four unknowns. These are found by solution of the following system of four equations:

$$F_{6cx} + F_{4cx} + F_{2cx} = 0 \qquad \text{x-force balance} \tag{4.12}$$

$$F_{6cy} + F_{4cy} + F_{2cy} = 0 \qquad \text{y-force balance} \tag{4.13}$$

$$F_{4cy} R_{4x} - F_{4cx} R_{4y} = 0 \qquad \text{axial force in rocker} \tag{4.14}$$

$$F_{6cy}(R_{2x} + R_{5x}) - F_{6cx}(R_{2y} + R_{5y}) = 0 \qquad \text{moment balance} \tag{4.15}$$

The system is solved sequentially by

$$F_{4cy} = \frac{F_{6cx} R_{5y} - F_{6cy} R_{5x}}{R_{2x} - R_{2y} \frac{R_{4x}}{R_{4y}}} \tag{4.16}$$

$$F_{4cx} = F_{4cy} \frac{R_{4x}}{R_{4y}} \tag{4.17}$$

$$F_{2cx} = -F_{4cx} - F_{6cx} \tag{4.18}$$

$$F_{2cy} = -F_{4cy} - F_{6cy} \tag{4.19}$$

which completes the force balance solution up to a few equal and opposites.

4.3.4 Structural Model

The results of the force balance are used to determine the link masses needed for purely structural purposes. The modeling strategy chosen for this problem is quite different from that employed for the rescue spreader problem. In that case, the computational expense of using finite element analysis was justified by several factors:

- Mass reduction was the primary purpose of the design study.
- A large portion of the mass was due to the jaws, which were characterized by complex loading and geometry and thus could not be accurately modeled by an analytical approximation.
- Mass minimization was the sole non-kinematic objective, allowing the bulk of the feasible computation time to be expended on it.

In constrast, for the present case, reducing engine mass would not constitute a major contribution to engine design. The primary purpose of the design study is therefore optimization of the combustion process, not mass minimization, and the relative importance of the accuracy of the structural model declines accordingly. Additionally, the structural components in this case are not as complex as the spreader jaws, and can be modeled analytically with greater accuracy. Finally, there a number of non-kinematic objectives, meaning that the computational expense of an FEA-based approach would be added to that required by the other objectives, resulting in an unacceptably large total.

In light of these considerations, a very simple model is employed in which the two-force members (i.e. the rocker and connecting rod) are treated as cylinders, with length equal to the distance between pin centers and area equal to the maximum force divided by the maximum allowable stress. The coupler is treated as three of these simple cylindrical links which happen to form a structure rather than a mobile part of a linkage. The crank is treated as a cantilever beam in bending, with length equal to the distance between pin

centers and a width to depth ratio of two, as shown in Figure 4.8. Thus determining the width of the beam suffices to completely specify it, using

$$W = \left(\frac{(2)6FL}{\sigma}\right)^{1/3} \tag{4.20}$$

where W is the width, F is the force, L is the length, σ is the maximum allowable stress, and the factor of 2 reflects the fixed aspect ratio. The force is the perpendicular component of F_{c2}.

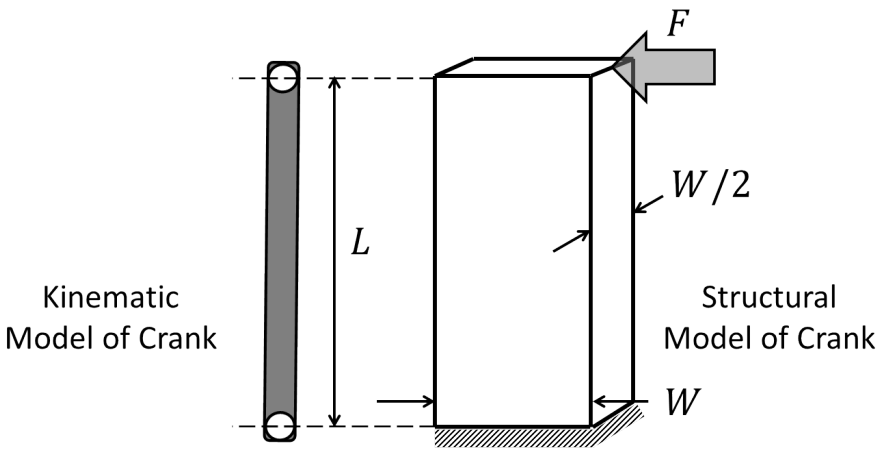


Figure 4.8: Structural Model of Crank

Specification of a material allows easy calculation of the mass of these simple links from the geometric parameters. Also, all are symmetrical shapes so that the center of mass is always located on the axis of the link halfway between the pins.

This concludes the primary objective function, with the major exception of calling the secondary optimization. The kinematic objectives, which are the piston side-load and the combustion objectives, have been determined, and all quantities necessary for the secondary optimization are known. At a conceptual level, the kinematics of the candidate design are now known, and the secondary optimization will now determine the best counterweighting of the links to optimize the non-kinematic objectives.

4.4 Secondary Objective Function

There are twelve secondary design variables: a length, angle, and mass each for the crank, coupler, rocker, and connecting rod. These define the size and position of a counterweighting point mass that is attached to each link to modify the existing center of mass (COM) properties resulting from the structural analysis. The objective function then computes the non-kinematic objectives based on the new COM properties as well as the kinematic information passed in from the primary level.

4.4.1 Link Center-of-Mass Properties

The two-point-mass determination of the COM properties for each link is illustrated in Figure 4.9. The mass, length, and angle set for the three COMs are defined in Figure 4.9, where the angles are defined relative to the line of centers of the link (i.e. in a local co-rotating coordinate system).

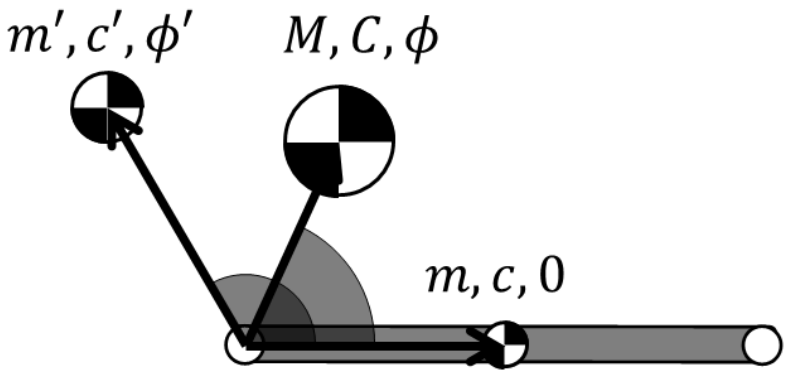


Figure 4.9: Two-point-mass model of link

The structural COM $\{m, c, 0\}$ is passed in from the primary level. The counterweight COM $\{m', c', \phi'\}$ is based on the values of the secondary design variables. The value of m' for each link is expressed as a fraction of the total structural mass of the linkage, and the value of c' is similarly expressed as a fraction of the longest link length, where the fractions are the actual design variables. The resulting overall COM $\{M, C, \phi\}$ is then determined by

$$
\begin{aligned}
M &= m + m' \\
C &= \frac{||mc + m'c'e^{i\phi'}||}{M} \\
\phi &= \frac{arg(mc + m'c'e^{i\phi'})}{M}
\end{aligned}
\tag{4.21}
$$

4.4.2 Polygonal Link Model

The point mass model provides a convenient way to define the design variables and integrate the structural and counterweighting aspects of link design. However, the shaking moment calculation requires a radius of gyration for each link, and the footprint calculation requires a geometric shape for each link. The point mass model is not suitable for estimating realistic values of these quantities. Instead, each link is assumed to be a polygon of order one higher than the link order (e.g. a ternary link is a quadrilateral), with a vertex at each pin center and the last vertex located so as to co-locate the centroid of the polygon with the center of mass of the link. This is admittedly a fairly crude modeling approach, and is not strictly consistent with the structural modeling of the links, but should be sufficient for capturing the correct trends for an approximate optimization. It also has the virtues of allowing analytical determination of the polygon centroids, rather than the expensive numerical approach required by an arbitrary shape, and greatly simplifies the footprint calculation as will be seen later on.

A separate subroutine was written for binary links and ternary links to determine the radius of gyration of the polygonal link K about its center of mass as well as the location of the last vertex so as to co-locate the centroid of the polygon with the center of mass of the link.

Binary Link

The binary case is quite simple, and is shown in Figure 4.10.

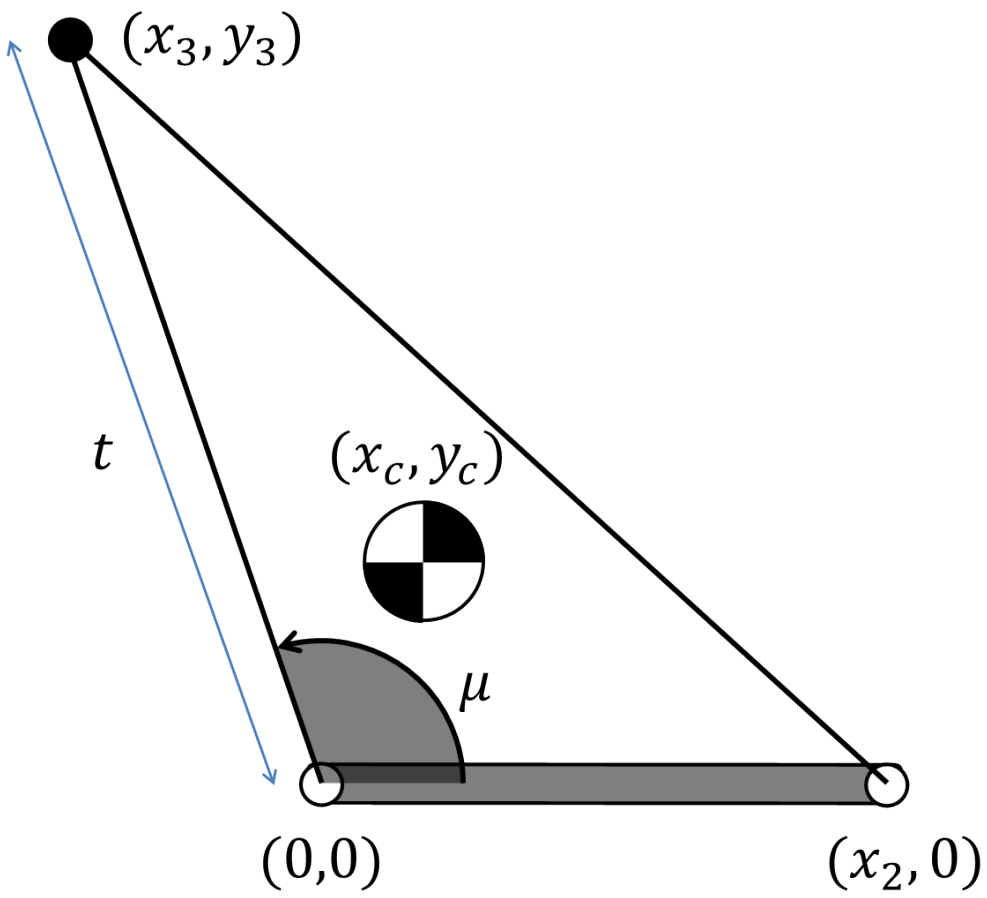


Figure 4.10: Polygonal model of binary link

The problem is solved in the local coordinate system where one pin as at the origin and the other is on the +x axis. These positions and the position of the COM (x_c, y_c) are known. Since the centroid of a triangle has x and y coordinates given by

$$\begin{aligned} x_c &= \frac{x_1 + x_2 + x_3}{3} \\ y_c &= \frac{y_1 + y_2 + y_3}{3} \end{aligned} \tag{4.22}$$

it is evident that when points 1,2, and c are known point 3 must be at

$$x_3 = 3x_c - x_2 - x_1$$
$$y_3 = 3y_c - y_2 - y_1 \tag{4.23}$$

which are expressed in polar form using angle μ and length t for output. The radius of gyration for the triangle about its COM is given by

$$K = \sqrt{\frac{x_2^2 - x_2 x_3 + x_3^2 + y_3^2}{18}}; \tag{4.24}$$

Ternary Link

The ternary case is more complicated. As the algebraic expression for the centroid of a quadrilateral is much more complicated and cannot be easily inverted as in the triangle case, it is necessary to treat the quadrilateral as the union of two triangles sharing a side. The starting situation is as shown in Figure 4.11. Note that $\vec{P}$ is the location of the centroid of the first triangle and $\vec{C}$ is the desired location of the quadrilateral centroid. The global linkage coordinate system is used as the joint locations are known in that system.

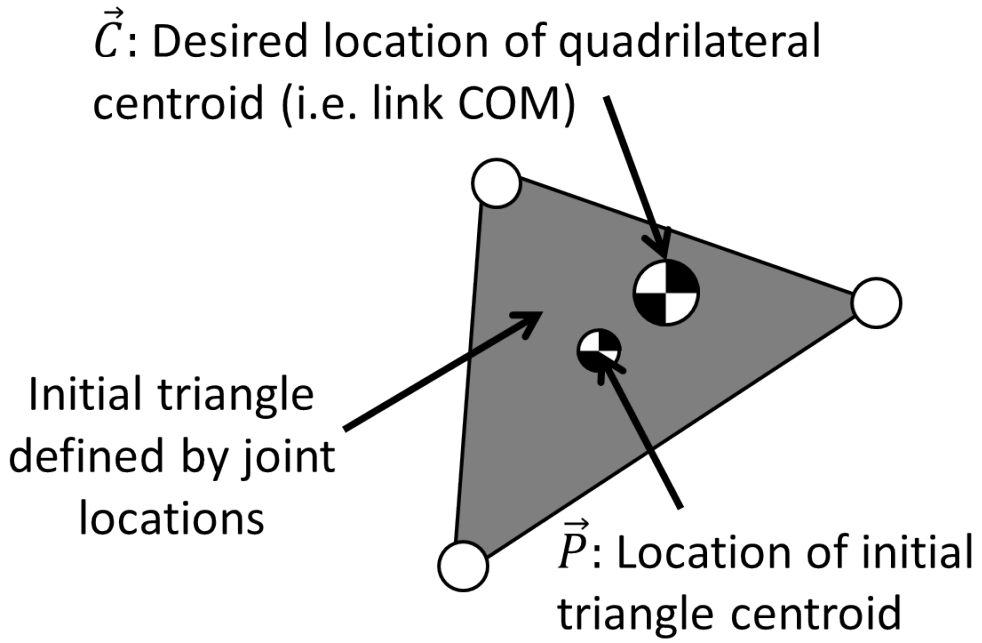


Figure 4.11: Polygonal model of ternary link (starting situation)

Next, a second triangle is added that shares a side with the first triangle. The shared side and location of the third point of the second triangle (i.e. the fourth and last point of the overall quadrilateral), denoted $\vec{Q}$, must be chosen so that the centroid of the resulting quadrilateral ends up in the desired location, which is equivalent to meeting the following two conditions:

- The centroid of the second triangle must be collinear with the centroid of the first triangle and the desired location of the quadrilateral centroid.
- The vector describing the location of the quadrilateral centroid is equal to the sum of the vector locations of the triangle centroids weighted by the areas of the triangles (consider computing the center of mass of a body consisting of two point masses).

These can be written mathematically as

$$\vec{Q} - \vec{P} = \gamma(\vec{C} - \vec{P}) \tag{4.25}$$

where γ is some scalar quantity, and

$$(A_1 + A_2)\vec{C} = A_1\vec{P} + A_2\vec{Q} \tag{4.26}$$

where A_i are the areas of the triangles. Note that A_2 depends on the choice of the location of the added fourth point.

The shared side is taken to be the side of the first triangle that is pierced by the semi-infinite ray starting $\vec{P}$ and passing through $\vec{C}$. It then remains to find the coordinates (x_4, y_4) of the fourth point such that $\vec{R}$ is correctly located to satisfy the two conditions. This is done using an iterative numerical algorithm.

Before describing this algorithm, it is necessary to discuss a lower bound on allowable γ, which is derived from the two conditions above:

$$\begin{aligned}
(A_1 + A_2)\vec{C} &= A_1\vec{P} + A_2\vec{Q} && \text{Start with equation 4.26} \\
(A_1 + A_2)\vec{C} &= (A_1 + A_2)\vec{P} + A_2(\vec{Q} - \vec{P}) \\
(A_1 + A_2)(\vec{C} - \vec{P}) &= A_2(\vec{Q} - \vec{P}) \\
(A_1 + A_2)(\vec{C} - \vec{P}) &= A_2\gamma(\vec{C} - \vec{P}) && \text{Substitute in equation 4.25} \\
\therefore \frac{A_1 + A_2}{A_2} &= \gamma
\end{aligned} \tag{4.27}$$

This proves that $\gamma > 1$. This is equivalent to the intuitively apparent stipulation that the COM be located between the centroids of the triangles.

Proceeding to the method of solution, the basic idea is as follows (note that $\vec{P}$ and $\vec{C}$ are known):

1. Guess a value γ', where the ' denotes a guess value
2. Evaluate $\vec{Q}'$ in equation 4.25
3. Compute the location of the fourth point using $\vec{Q}'$ (as in equation 4.23)

4. With all vertices of the second triangle known, compute A_2'

5. Compute $\vec{C}'$ using equation 4.26

6. Evaluate $(\vec{C}' - \vec{P}) - (\vec{C} - \vec{P})$

7. Formulate a better guess for γ, repeat

An attempt at an analytical solution resulted in a nonlinear system of five equations which resisted attempts at direct solution. The difficulty arises from the fact that moving the location of the fourth point changes both the location of the second triangle's centroid and the area of the triangle, i.e. the "weight" of the centroid.

The specific method of solution chosen is a binary search to within a finite error tolerance (chosen to be 5×10^{-3}) on a continuum. This makes use of the fact that the objective value $(\vec{C}' - \vec{P}) - (\vec{C} - \vec{P})$ will be positive if γ is too big, negative if γ is too small, and zero if γ is exactly right. The binary search can look for the point of sign change without needing to use gradients. Additionally, if bounding values of γ that are known to contain the correct value of γ are known, the algorithm is guaranteed to converge (with the stipulation that there is only one point of sign change), and moreover will do so quickly in a manner independent of the exact nature of the function, since only the sign of the function value matters.

An objective function is defined as follows:

error = objFxn(γ')

Compute candidate location $\vec{Q}$ of second centroid:

$Q'_x = P_x + \gamma'(C_x - P_x)$

$Q'_y = P_y + \gamma'(C_y - P_y)$

Find corresponding candidate location of fourth point:

$x'_4 = 3Q'_x - \alpha_x - \beta_x$

$y'_4 = 3Q'_y - \alpha_y - \beta_y$

Find corresponding candidate area A_2:

$A'_2 = ||(1/2)((\beta_x y'_4 - x'_4 \beta_y) + (x'_4 \alpha_y - \alpha_x y'_4) + (\alpha_x \beta_y - \beta_x \alpha_y))||$

Find corresponding candidate value of $\vec{C}$

$C_x = \frac{A_1 P_x + A'_2 Q_x}{(A_1 + A'_2)}$

$C_y = \frac{A_1 P_y + A'_2 Q_y}{(A_1 + A'_2)}$

Find error based on inconsistency between $\vec{C}'$ and $\vec{C}$

error $= |\vec{C}' - \vec{P}| - |\vec{C} - \vec{P}|$

return error

Here α and β are two of the three points of the first triangle, chosen on a case-by-case basis to define the shared side correctly.

The lower bound is $\gamma = 1$, based on the proof that $\gamma > 1$. The upper bound is found by a preliminary subroutine. The main function goes as follows:

γ = FindGamma()

Find upper limit on γ

γ_2 = 1, error $= -1$

while error < 0

$\gamma_1 = \gamma_2$

$\gamma_2 = 2\,\gamma_1$

error = objFxn(γ_2)

end while

Bounds established; binary search to within tolerance

while |error| $>$ tolerance

$\gamma' = (1/2)(\gamma_1 + \gamma_2)$

error = objFxn(γ')

if error < 0

$\gamma_1 = \gamma'$

else

$\gamma_2 = \gamma'$

end if

end while

return γ'

The value of γ calculated using this algorithm is correct within the tolerance and can be used to find the correct location of $\vec{Q}$, resulting in the successfully constructed quadrilateral shown in Figure 4.12.

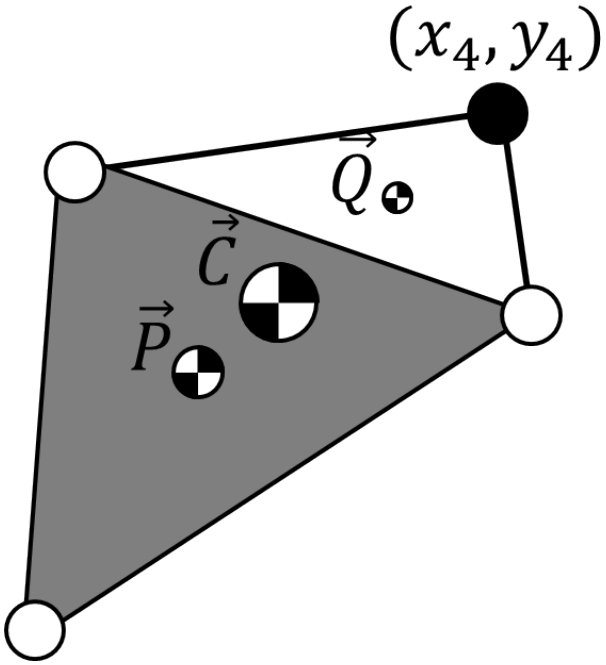


Figure 4.12: Polygonal model of ternary link (solution complete)

It should be noted that the technique used for the ternary link can be generalized almost without modification to a link of any order, adding a triangle to a polygon of the appropriate order.

The final step is to compute the polar moment of inertia of each triangle about its own centroid, convert to the polar moments of inertia about the quadrilateral centroid using the parallel axis theorem, and then convert to a radius of gyration using

$$K = \sqrt{\frac{I_1 + I_2}{A_1 + A_2}} \tag{4.28}$$

where I_i are the polar moments of inertia about the quadrilateral centroid.

4.4.3 Shaking Force and Moment

The shaking force associated with a mechanism at any given instant is equivalent to the time rate of change of the linear momentum of the mechanism's center of mass (COM), since by Newton's second law

$$F = \frac{d}{dt}(MV) \tag{4.29}$$

Therefore if M is the mechanism mass and V is the linear velocity of the COM, F is the shaking force, i.e. the force the linkage exerts on the ground structure in order to change the momentum of its COM. Similarly, the shaking moment associated with a mechanism at any given instant is equivalent to the time rate of change of the angular momentum of the mechanism with respect to a given point in the plane. The rotational analogue of equation 4.29 is

$$H = \frac{dL}{dt} \tag{4.30}$$

where H is the shaking moment and L is the angular momentum of the mechanism.

In order to determine the shaking force and moment, then, it is necessary to compute the linear and angular momentum of each link, sum them to find the overall momenta, and then differentiate with respect to time. The linear momentum of link j can be found simply by multiplying its mass m_j by the linear velocity of its COM, v_j. Letting R_{sj} be the vector from the origin to the COM of link j, the linear momentum of the link P_j is

$$P_j = m_j \frac{dR_{sj}}{dt} \tag{4.31}$$

and the total linear momentum of the linkage is

$$P = \sum_j P_j \tag{4.32}$$

In practice, it is necessary to find the x and y components of the linear momentum separately in order to enable a scalarized calculation suitable for a computer program. This means that equation 4.31 splits into

$$P_{j,x} = m_j \dot{x_j} \tag{4.33}$$

$$P_{j,y} = m_j \dot{y_j} \tag{4.34}$$

where (x_j, y_j) are the coordinates of the link COM with respect to the origin and $(\dot{x_j}, \dot{y_j})$ is the linear velocity of the COM.

Each link contributes angular momentum in two ways: through the revolution of the link COM about the origin, and through the rotation of the link about its own COM. The angular momentum of revolution of link j is given by crossing the position of the link COM with its linear momentum:

$$L_{rev} = R_{sj} \times (m_j v_j) \tag{4.35}$$

which can be written as

$$L_{rev} = m_j(x_j \dot{y_j} - y_j \dot{x_j}) \tag{4.36}$$

The angular momentum of rotation of link j is given by

$$L_{rot} = m_j k_j^2 \dot{\theta}_j \tag{4.37}$$

where k_j is the radius of gyration of the link about its COM and $\dot{\theta}_j$ is the time rate of change of the angle of the link in the plane, which is the same as the angular velocity of the link about its COM. Therefore the total angular momentum of link j is

$$L_j = m_j(x_j\dot{y}_j - y_j\dot{x}_j + k_j^2\dot{\theta}_j) \tag{4.38}$$

and the total angular momentum of the linkage is

$$L = \sum_j L_j \tag{4.39}$$

The calculation of the link momenta uses the notation shown in Figure 4.13. In general, for link j, θ_j is the angle of the link with respect to the +x axis, $\vec{R}_j$ is the link vector, $\vec{C}_j$ is the vector from the base of the link vector to the position of the link's COM, and ϕ_j is the angle from $\vec{R}_j$ to $\vec{C}_j$. The masses, COM locations, and radii of gyration used in these calculations are the ones determined previously using point-mass counterweighting and polygonal modeling.

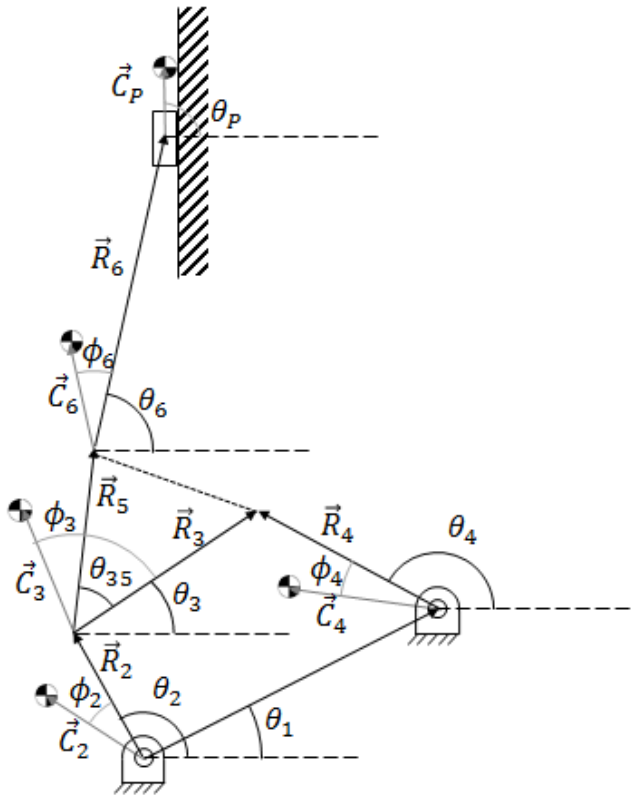


Figure 4.13: Linkage Geometry

Link 2 - Crank

Link COM coordinates and their time derivatives:

$$x_2 = c_2 cos(\theta_2 + \phi_2) \tag{4.40a}$$

$$y_2 = c_2 sin(\theta_2 + \phi_2) \tag{4.40b}$$

$$\dot{x_2} = -\dot{\theta_2} c_2 sin(\theta_2 + \phi_2) \tag{4.40c}$$

$$\dot{y_2} = \dot{\theta_2} c_2 cos(\theta_2 + \phi_2) \tag{4.40d}$$

The linear momentum is found from equations 4.33 and 4.34,

$$
\begin{aligned}
P_{2,x} &= -m_2[\dot{\theta_2} c_2 sin(\theta_2 + \phi_2)] \\
P_{2,y} &= m_2[\dot{\theta_2} c_2 cos(\theta_2 + \phi_2)]
\end{aligned}
\tag{4.41}
$$

and the angular momentum is found from equation 4.38,

$$
\begin{aligned}
L_2 &= m_2(c_2^2 cos^2(\theta_2 + \phi_2)\dot{\theta_2} + c_2^2 sin^2(\theta_2 + \phi_2)\dot{\theta_2} + k_2^2\dot{\theta_2}) \\
&= m_2[\dot{\theta_2}(c_2^2 + k_2^2)]
\end{aligned}
\tag{4.42}
$$

Link 3 Coupler

Link COM coordinates and their time derivatives:

$$x_3 = r_2 cos(\theta_2) + c_3 cos(\theta_3 + \phi_3) \tag{4.43a}$$

$$y_3 = r_2 sin(\theta_2) + c_3 sin(\theta_3 + \phi_3) \tag{4.43b}$$

$$\dot{x_3} = -\dot{\theta_2} r_2 sin(\theta_2) - \dot{\theta_3} c_3 sin(\theta_3 + \phi_3) \tag{4.43c}$$

$$\dot{y_3} = \dot{\theta_2} r_2 cos(\theta_2) + \dot{\theta_3} c_3 cos(\theta_3 + \phi_3) \tag{4.43d}$$

The linear momentum is found from equations 4.33 and 4.34,

$$
\begin{aligned}
P_{3,x} &= -m_3[\dot{\theta_2} r_2 sin(\theta_2) + \dot{\theta_3} c_3 sin(\theta_3 + \phi_3)] \\
P_{3,y} &= m_3[\dot{\theta_2} r_2 cos(\theta_2) + \dot{\theta_3} c_3 cos(\theta_3 + \phi_3)]
\end{aligned}
\tag{4.44}
$$

and the angular momentum is found from equation 4.38,

$$L_3 = m_3[\dot{\theta_2} r_2^2 + \dot{\theta_3}(c_3^2 + k_3^2) + (\dot{\theta_2} + \dot{\theta_3})(r_2 c_3 cos(\theta_3 + \phi_3 - \theta_2))] \tag{4.45}$$

Link 4 - Rocker

Link COM coordinates and their time derivatives:

$$x_4 = r_1 cos(\theta_1) + c_4 cos(\theta_4 + \phi_4) \tag{4.46a}$$

$$y_4 = r_1 sin(\theta_1) + c_4 sin(\theta_4 + \phi_4) \tag{4.46b}$$

$$\dot{x_4} = -\dot{\theta_4} c_4 sin(\theta_4 + \phi_4) \tag{4.46c}$$

$$\dot{y_4} = \dot{\theta_4} c_4 cos(\theta_4 + \phi_4) \tag{4.46d}$$

$$\tag{4.46e}$$

The linear momentum is found from equations 4.33 and 4.34,

$$\begin{aligned} P_{4,x} &= -m_4[\dot{\theta_4} c_4 sin(\theta_4 + \phi_4)] \\ P_{4,y} &= m_4[\dot{\theta_4} c_4 cos(\theta_4 + \phi_4)] \end{aligned} \tag{4.47}$$

and the angular momentum is found from equation 4.38,

$$L_4 = m_4[\dot{\theta_4}(c_4^2 + k_4^2 + r_1 c_4 cos(\theta_4 + \phi_4 - \theta_1))] \tag{4.48}$$

Link 6 - Connecting Rod

Link COM coordinates and their time derivatives:

$$x_6 = r_2 cos(\theta_2) + r_5 cos(\theta_3 + \theta_{35}) + c_6 cos(\theta_6 + \phi_6) \tag{4.49a}$$

$$y_6 = r_2 sin(\theta_2) + r_5 sin(\theta_3 + \theta_{35}) + c_6 sin(\theta_6 + \phi_6) \tag{4.49b}$$

$$\dot{x_6} = -\dot{\theta_2} r_2 sin(\theta_2) - \dot{\theta_3} r_5 sin(\theta_3 + \theta_{35}) - \dot{\theta_6} c_6 sin(\theta_6 + \phi_6) \tag{4.49c}$$

$$\dot{y_6} = \dot{\theta_2} r_2 cos(\theta_2) + \dot{\theta_3} r_5 cos(\theta_3 + \theta_{35}) + \dot{\theta_6} c_6 cos(\theta_6 + \phi_6) \tag{4.49d}$$

The linear momentum is found from equations 4.33 and 4.34,

$$\begin{aligned} P_{6,x} &= -m_6[\dot{\theta_2} r_2 sin(\theta_2) + \dot{\theta_3} r_5 sin(\theta_3 + \theta_{35}) + \dot{\theta_6} c_6 sin(\theta_6 + \phi_6)] \\ P_{6,y} &= m_6[\dot{\theta_2} r_2 cos(\theta_2) + \dot{\theta_3} r_5 cos(\theta_3 + \theta_{35}) + \dot{\theta_6} c_6 cos(\theta_6 + \phi_6)] \end{aligned} \tag{4.50}$$

and the angular momentum is found from equation 4.38,

$$\begin{aligned} L_6 = m_6[&\dot{\theta_2} r_2^2 + \dot{\theta_3} r_5^2 + \dot{\theta_6}(c_6^2 + k_6^2) \\ &+ cos(\theta_3 + \theta_{35} - \theta_2)(r_2 r_5(\dot{\theta_2} + \dot{\theta_3})) \\ &+ cos(\theta_6 + \phi_6 - \theta_2)(r_2 c_6(\dot{\theta_2} + \dot{\theta_6})) \\ &+ cos(\theta_3 + \theta_{35} - \theta_6 - \phi_6)(r_5 c_6(\dot{\theta_3} + \dot{\theta_6}))] \end{aligned} \tag{4.51}$$

Piston

Link COM coordinates and their time derivatives:

$$x_P = r_2 cos(\theta_2) + r_5 cos(\theta_3 + \theta_{35}) + r_6 cos(\theta_6) + c_P cos(\phi_P) \tag{4.52a}$$

$$y_P = r_2 cos(\theta_2) + r_5 cos(\theta_3 + \theta_{35}) + r_6 cos(\theta_6) + c_P sin(\phi_P) \tag{4.52b}$$

$$\dot{x_P} = -\dot{\theta_2} r_2 sin(\theta_2) - \dot{\theta_3} r_5 sin(\theta_3 + \theta_{35}) - \dot{\theta_6} r_6 sin(\theta_6) \tag{4.52c}$$

$$\dot{y_P} = \dot{\theta_2} r_2 cos(\theta_2) + \dot{\theta_3} r_5 cos(\theta_3 + \theta_{35}) + \dot{\theta_6} r_6 cos(\theta_6)) \tag{4.52d}$$

The linear momentum is found from equations 4.33 and 4.34,

$$\begin{aligned} P_{P,x} &= -m_P[\dot{\theta_2} r_2 sin(\theta_2) + \dot{\theta_3} r_5 sin(\theta_3 + \theta_{35}) + \dot{\theta_6} r_6 sin(\theta_6)] \\ P_{P,y} &= m_P[\dot{\theta_2} r_2 cos(\theta_2) + \dot{\theta_3} r_5 cos(\theta_3 + \theta_{35}) + \dot{\theta_6} r_6 cos(\theta_6)] \end{aligned} \tag{4.53}$$

and the angular momentum is found from equation 4.38,

$$\begin{aligned} L_P = m_P[&\dot{\theta_2} r_2^2 + \dot{\theta_3} r_5^2 + \dot{\theta_6}(r_6^2 + r_6 c_P cos(\theta_6 - \phi_P)) \\ &+ cos(\theta_3 + \theta_{35} - \theta_2)(r_2 r_5(\dot{\theta_2} + \dot{\theta_3})) \\ &+ cos(\theta_6 + \phi_6 - \theta_2)(r_2 c_6(\dot{\theta_2} + \dot{\theta_6})) \\ &+ cos(\phi_2 - \phi_P)(\dot{\theta_2} r_2 c_P) + cos(\theta_3 + \theta_{35} - \phi_b)(\dot{\theta_3} r_5 c_P)] \end{aligned} \tag{4.54}$$

Summation and Differentiation of Results

With all links complete, the results are added to form the total momenta:

$$
\begin{aligned}
P_x = & -m_2[\dot{\theta}_2 c_2 sin(\theta_2 + \phi_2)] \\
& - m_3[\dot{\theta}_2 r_2 sin(\theta_2) + \dot{\theta}_3 c_3 sin(\theta_3 + \phi_3)] \\
& - m_4[\dot{\theta}_4 c_4 sin(\theta_4 + \phi_4)] \\
& - m_6[\dot{\theta}_2 r_2 sin(\theta_2) + \dot{\theta}_3 r_5 sin(\theta_3 + \theta_{35}) + \dot{\theta}_6 c_6 sin(\theta_6 + \phi_6)] \\
& - m_P[\dot{\theta}_2 r_2 sin(\theta_2) + \dot{\theta}_3 r_5 sin(\theta_3 + \theta_{35}) + \dot{\theta}_6 r_6 sin(\theta_6)]
\end{aligned}
\tag{4.55}
$$

$$
\begin{aligned}
P_y = & \; m_2[\dot{\theta}_2 c_2 cos(\theta_2 + \phi_2)] \\
& + m_3[\dot{\theta}_2 r_2 cos(\theta_2) + \dot{\theta}_3 c_3 cos(\theta_3 + \phi_3)] \\
& + m_4[\dot{\theta}_4 c_4 cos(\theta_4 + \phi_4)] \\
& + m_6[\dot{\theta}_2 r_2 cos(\theta_2) + \dot{\theta}_3 r_5 cos(\theta_3 + \theta_{35}) + \dot{\theta}_6 c_6 cos(\theta_6 + \phi_6)] \\
& + m_P[\dot{\theta}_2 r_2 cos(\theta_2) + \dot{\theta}_3 r_5 cos(\theta_3 + \theta_{35}) + \dot{\theta}_6 r_6 cos(\theta_6)]
\end{aligned}
\tag{4.56}
$$

$$
\begin{aligned}
L = {} & m_2\dot{\theta}_2(c_2^2 + k_2^2) \\
& + m_3[\dot{\theta}_2 r_2^2 + \dot{\theta}_3(c_3^2 + k_3^2) + (\dot{\theta}_2 + \dot{\theta}_3)(r_2 c_3 cos(\theta_3 + \phi_3 - \theta_2))] \\
& + m_4[\dot{\theta}_4(c_4^2 + k_4^2 + r_1 c_4 cos(\theta_4 + \phi_4 - \theta_1))] \\
& + m_6[\dot{\theta}_2 r_2^2 + \dot{\theta}_3 r_5^2 + \dot{\theta}_6(c_6^2 + k_6^2) \\
& \quad + cos(\theta_3 + \theta_{35} - \theta_2)(r_2 r_5(\dot{\theta}_2 + \dot{\theta}_3)) \\
& \quad + cos(\theta_6 + \phi_6 - \theta_2)(r_2 c_6(\dot{\theta}_2 + \dot{\theta}_6)) \\
& \quad + cos(\theta_3 + \theta_{35} - \theta_6 - \phi_6)(r_5 c_6(\dot{\theta}_3 + \dot{\theta}_6))] \\
& + m_P[\dot{\theta}_2 r_2^2 + \dot{\theta}_3 r_5^2 + \dot{\theta}_6(r_6^2 + r_6 c_P cos(\theta_6 - \phi_P)) \\
& \quad + cos(\theta_3 + \theta_{35} - \theta_2)(r_2 r_5(\dot{\theta}_2 + \dot{\theta}_3)) \\
& \quad + cos(\theta_6 + \phi_6 - \theta_2)(r_2 c_6(\dot{\theta}_2 + \dot{\theta}_6)) \\
& \quad + cos(\theta_2 - \phi_P)(\dot{\theta}_2 r_2 c_P) + cos(\theta_3 + \theta_{35} - \phi_P)(\dot{\theta}_3 r_5 c_P)]
\end{aligned}
\tag{4.57}
$$

The final step is to take the time derivatives of equations 4.55, 4.56, and 4.57 to find the shaking force in the x and y directions and the shaking moment, respectively.

The x shaking force is:

$$
\begin{aligned}
F_x = & -m_2[\ddot{\theta}_2 c_2 sin(\theta_2+\phi_2) + \dot{\theta_2}^2 c_2 cos(\theta_2+\phi_2)] \\
& - m_3[\ddot{\theta}_2 r_2 sin(\theta_2) + \dot{\theta_2}^2 r_2 cos(\theta_2) + \ddot{\theta}_3 c_3 sin(\theta_3+\phi_3) + \dot{\theta_3}^2 c_3 cos(\theta_3+\phi_3)] \\
& - m_4[\ddot{\theta}_4 c_4 sin(\theta_4+\phi_4) + \dot{\theta_4}^2 c_4 cos(\theta_4+\phi_4)] \\
& - m_6[\ddot{\theta}_2 r_2 sin(\theta_2) + \dot{\theta_2}^2 r_2 cos(\theta_2) + \ddot{\theta}_3 r_5 sin(\theta_3+\theta_{35}) \\
& \quad + \dot{\theta_3}^2 r_5 cos(\theta_3+\theta_{35}) + \ddot{\theta}_6 c_6 sin(\theta_6+\phi_6) + \dot{\theta_6}^2 c_6 cos(\theta_6+\phi_6)] \\
& - m_P[\ddot{\theta}_2 r_2 sin(\theta_2) + \dot{\theta_2}^2 r_2 cos(\theta_2) + \ddot{\theta}_3 r_5 sin(\theta_3+\theta_{35}) \\
& \quad + \dot{\theta_3}^2 r_5 cos(\theta_3+\theta_{35}) + \ddot{\theta}_6 r_6 sin(\theta_6) + \dot{\theta_6}^2 r_6 cos(\theta_6)]
\end{aligned}
\tag{4.58}
$$

The y shaking force is:

$$
\begin{aligned}
F_y = & \; m_2[\ddot{\theta}_2 c_2 cos(\theta_2+\phi_2) - \dot{\theta_2}^2 c_2 sin(\theta_2+\phi_2)] \\
& + m_3[\ddot{\theta}_2 r_2 cos(\theta_2) - \dot{\theta_2}^2 r_2 sin(\theta_2) + \ddot{\theta}_3 c_3 cos(\theta_3+\phi_3) - \dot{\theta_3}^2 c_3 sin(\theta_3+\phi_3)] \\
& + m_4[\ddot{\theta}_4 c_4 cos(\theta_4+\phi_4) - \dot{\theta_4}^2 c_4 sin(\theta_4+\phi_4)] \\
& + m_6[\ddot{\theta}_2 r_2 cos(\theta_2) - \dot{\theta_2}^2 r_2 sin(\theta_2) + \ddot{\theta}_3 r_5 cos(\theta_3+\theta_{35}) \\
& \quad - \dot{\theta_3}^2 r_5 sin(\theta_3+\theta_{35}) + \ddot{\theta}_6 c_6 cos(\theta_6+\phi_6) - \dot{\theta_6}^2 c_6 sin(\theta_6+\phi_6)] \\
& + m_P[\ddot{\theta}_2 r_2 cos(\theta_2) - \dot{\theta_2}^2 r_2 sin(\theta_2) + \ddot{\theta}_3 r_5 cos(\theta_3+\theta_{35}) \\
& \quad - \dot{\theta_3}^2 r_5 sin(\theta_3+\theta_{35}) + \ddot{\theta}_6 r_6 cos(\theta_6) - \dot{\theta_6}^2 r_6 sin(\theta_6)]
\end{aligned}
\tag{4.59}
$$

The shaking moment is:

$$
\begin{aligned}
H = {} & m_2\ddot{\theta}_2(c_2^2 + k_2^2) \\
& + m_3[\ddot{\theta}_2 r_2^2 + \ddot{\theta}_3(c_3^2 + k_3^2) + (\ddot{\theta}_2 + \ddot{\theta}_3)(r_2 c_3 cos(\theta_3 + \phi_3 - \theta_2)) \\
& \quad - (\dot{\theta_3}^2 - \dot{\theta_2}^2)(r_2 c_3 sin(\theta_3 + \phi_3 - \theta_2))] \\
& + m_4[\ddot{\theta}_4(c_4^2 + k_4^2 + r_1 c_4 cos(\theta_4 + \phi_4 - \theta_1)) - \dot{\theta_4}^2 r_1 c_4 sin(\theta_4 + \phi_4 - \theta_1)] \\
& + m_6[\ddot{\theta}_2 r_2^2 + \ddot{\theta}_3 r_5^2 + \ddot{\theta}_6(c_6^2 + k_6^2) \\
& \quad + (\ddot{\theta}_2 + \ddot{\theta}_3)(r_2 r_5 cos(\theta_3 + \theta_{35} - \theta_2)) - (\dot{\theta_3}^2 - \dot{\theta_2}^2)(r_2 r_5 sin(\theta_3 + \theta_{35} - \theta_2)) \\
& \quad + (\ddot{\theta}_2 + \ddot{\theta}_6)(r_2 c_6 cos(\theta_6 + \phi_6 - \theta_2)) - (\dot{\theta_6}^2 - \dot{\theta_2}^2)(r_2 c_6 sin(\theta_6 + \phi_6 - \theta_2)) \\
& \quad + (\ddot{\theta}_6 + \ddot{\theta}_3)(r_5 c_6 cos(\theta_3 + \theta_{35} - \theta_6 - \phi_6)) - (\dot{\theta_3}^2 - \dot{\theta_6}^2)(r_5 c_6 sin(\theta_3 + \theta_{35} - \theta_6 - \phi_6))] \\
& + m_P[\ddot{\theta}_2 r_2^2 + \ddot{\theta}_3 r_5^2 + \ddot{\theta}_6(r_6^2 + r_6 c_P cos(\theta_6 - \phi_P)) - \dot{\theta_6}^2 r_6 c_P sin(\theta_6 - \phi_P) \\
& \quad + (\ddot{\theta}_2 + \ddot{\theta}_3)(r_2 r_5 cos(\theta_3 + \theta_{35} - \theta_2)) - (\dot{\theta_3}^2 - \dot{\theta_2}^2)(r_2 r_5 sin(\theta_3 + \theta_{35} - \theta_2)) \\
& \quad + (\ddot{\theta}_2 + \ddot{\theta}_6)(r_2 c_6 cos(\theta_6 + \phi_6 - \theta_2)) - (\dot{\theta_6}^2 - \dot{\theta_2}^2)(r_2 c_6 sin(\theta_6 + \phi_6 - \theta_2)) \\
& \quad + \ddot{\theta}_2 r_2 c_P cos(\theta_2 - \phi_P) - \dot{\theta_2}^2 r_2 c_P sin(\theta_2 - \phi_P) \\
& \quad + \ddot{\theta}_3 r_5 c_P cos(\theta_3 + \theta_{35} - \phi_P) - \dot{\theta_3}^2 r_5 c_P sin(\theta_3 + \theta_{35} - \phi_P)]
\end{aligned}
\tag{4.60}
$$

4.4.4 Planar Footprint

Estimation of the size of the linkage's planar footprint is done as follows. Rather than taking a circumscribing rectangular box, the footprint is defined as the area of the convex hull of the set of points in the plane that any part of the linkage passes through during a full revolution. Thus determination of the footprint is broken into two phases: assembly of the set of points passed through, followed by reduction to the convex hull of that set and computation of the area.

The set of points in the plane occupied for at least one instant of time by the linkage during a revolution can be thought of as the union of the infinite set of all sets of points

occupied by the linkage at each position it assumes during the course of its motion. Of course, for computational purposes only a finite number of positions are evaluated, so the footprint is correspondingly approximated by the union of a finite number of sets of points corresponding to the evaluated positions. Thus the problem reduces to finding the set of points occupied at a given instantaneous position of the linkage, which in turn reduces to finding the set of points occupied by each link for that position. Here the virtues of the polygonal model become apparent. The convex hull of a polygon is equal to the convex hull of its vertices, so an expensive high-resolution outline approximation need not be used as simply keeping track of the vertices suffices. Thus in practice to find the convex hull of the space swept by the linkage one simply computes the locations of the link polygon vertices at a number of positions, concatenates the results, and finds the convex hull of the resulting set. In the notation of set theory, this process can be expressed as

$$\begin{aligned} conv(L_{i,j}) &= conv(\{V_k\}_{i,j}) \\ P_i &= \bigcup_{j=1}^{l} L_{i,j} \\ A &= \bigcup_{i=1}^{p} P_i \\ \therefore conv(A) &= conv\Big(\bigcup_{i=1}^{p}\Big(\bigcup_{j=1}^{l}\{V_k\}_{i,j}\Big)\Big) \end{aligned} \tag{4.61}$$

where $L_{i,j}$ is the set of all points in polygonal link j at linkage position i, $\{V_k\}_{i,j}$ is the set of the k vertex locations of polygonal link j at linkage position i, P_i is the set of all the vertex locations of the linkage at position i, and A is the set of all vertex locations at all positions. Once set A is assembled, the actual computation of the convex hull and the area thereof is done using the built-in Matlab function "convhull". An example is illustrated in Figure 4.14. On the left, a linkage is shown in one position, with the kinematic vectors outlined in black and the polygonal link shapes filled in. On the right, twenty positions of

the same linkage are shown, with the convex hull of the linkage positions computed and shown in gray in the background.

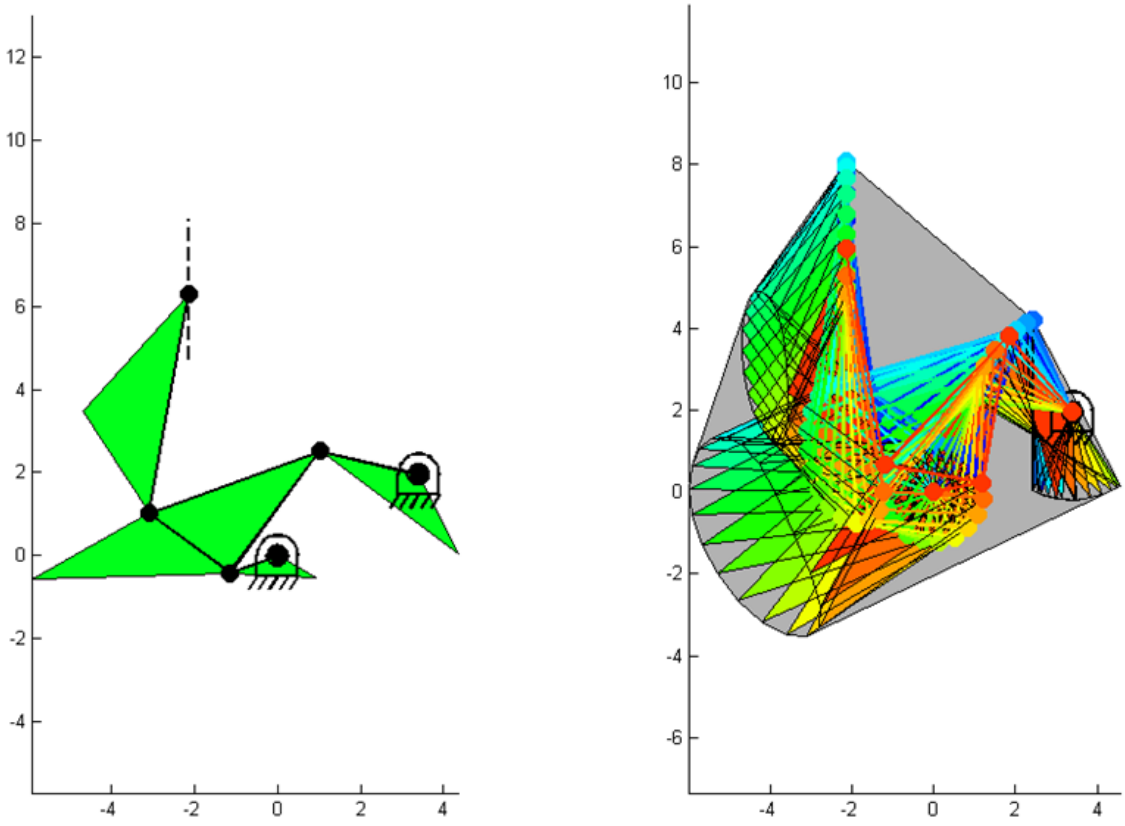


Figure 4.14: Linkage footprint example

With the shaking force and moment computed and the footprint determined, the only remaining objective is the total mass, which is obtained simply by adding the known link masses. The secondary objective function evaluation can now return.

4.5 Optimization Execution and Results

The specification of the entire optimization procedure is now complete, and it is possible to proceed to the specific parameters used as well as the results obtained.

4.5.1 Optimization Parameters

The optimization algorithm used was NSGA-II at the inner level and the modified version described in Section 2.4.4 at the outer level. Both were implemented by the author in Matlab. Two separate optimizations were run in parallel on two computers, with the only difference being whether the open or crossed configuration was taken for the base four-bar. The solutions were combined into a single pool for examination after algorithm termination. In both cases, 300 generations of 150 individuals were used at the outer level and 150 generations of 150 individuals at the inner level, with each individual's chromosome having a 20-digit binary representation at both levels. There were six objectives:

- Maximize combustion efficiency
- Minimize peak side-loading of piston
- Minimize RMS shaking force
- Minimize RMS shaking moment
- Minimize total mass of linkage
- Minimize planar footprint of linkage

In practice maximization of the efficiency was accomplished by minimizing its negative. There were eight design variables at the outer level, namely $\{t_1, n_3, n_4, n_5, m_6, \theta_1, \theta_{35}, t_s\}$, which are indirectly specified as described in Section 4.2.2. There were twelve design variables at the inner level, which specified the position and size of the counterweighting

point masses added to four links as described in Section 4.4. The bounds on the design variables can be found in Tables 4.2 and 4.3, along with the values of the selected solution.

4.5.2 Results

The optimization process took about 100 hours on a pair of desktop computers with Intel i5 processors. Each of the two parallel optimizations involved 300 x 150 or 45,000 primary objective function calls, each of which solved an inner level optimization with 150 x 150 or 22,500 secondary objective function calls. Thus the secondary objective function was called over a billion times in the course of the optimization,which averages out to about 2,800 secondary objective function calls per second. This is a good illustration of the fact that for a nested optimization with both levels having a multi-objective genetic algorithm, objective function cost must be extremely low (sub-millisecond in this case) to ensure computational feasibility.

A polygon plot of the Pareto-optimal solutions returned is given in Figure 4.15, using the system described in Section 3.8.2. As a reminder, proximity to a vertex indicates doing relatively badly at that objective, and bluer solutions have a better unweighted average of normalized objective values. There are far more solutions plotted than in the previous study, 25,884 instead of 90, as each solution at the outer level has many corresponding inner-level solutions since both levels were multi-objective in this optimization.

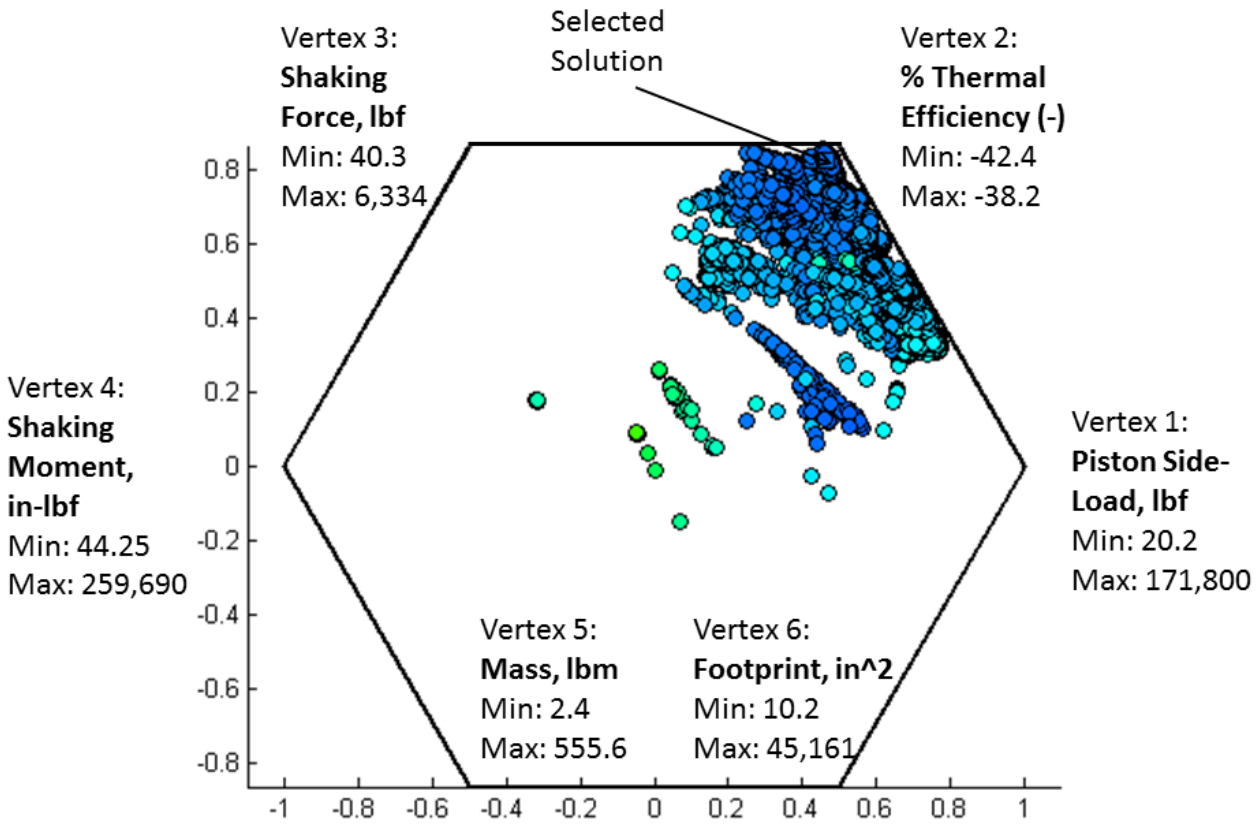


Figure 4.15: Polygon plot of solutions to ICE problem

This plot must be interpreted carefully. At first glance, the crowding of all but a few high-average-value solutions near Vertex 2 looks dramatic, but it is important to note that Objective 2 is the only one that does not vary over several orders of magnitude, and in fact only has a very slight variation in the values that the Pareto-optimal set assumes. The solution that was selected has a very low average value but is very close to Vertex 2. This is equivalent to the fact that a percentage point or two of efficiency was sacrificed to achieve order of magnitude improvements with respect to the mechanical objectives. The objective values of the selected solution are given below in Table 4.1, followed by the design variable values and the bounds that were set on the design space in Tables 4.2 and 4.3.

Table 4.1: Objective values for selected solution

Objective	Value
Piston Side-Load, lbf	168
% Thermal Efficiency (-)	-39.7
RMS Shaking Force, lbf	178
RMS Shaking Moment, in-lbf	130
Total Linkage Mass, lbm	5.3
Planar Footprint, in^2	273.4

Table 4.2: Outer level design variable values

Variable	Min	Max	Optimal	Variable	Min	Max	Optimal
t_2	0	1	0.7162	n_3	1.42	10	7.9302
n_4	1.42	10	3.1735	n_5	0	10	3.2628
m_6	1.1	10	4.4146	t_s	0	1	0.4933
θ_1	0	2 π	6.2109	θ_{35}	0	2 π	1.219

Table 4.3: Inner level design variable values

Variable	Min	Max	Optimal	Variable	Min	Max	Optimal
m'_2	0	5	0.6230	c'_2	0	5	0.0097
ϕ'_2	0	2π	3.1509	m'_3	0	5	0.0178
c'_3	0	5	0.0649	ϕ'_3	0	2π	2.9264
m'_4	0	5	0.0510	c'_4	0	5	0.4308
ϕ'_4	0	2π	3.2625	m'_6	0	5	0.0385
c'_6	0	5	0.3266	ϕ'_6	0	2π	2.3148

The optimal values in Table 4.2 are safely in the interior of the design space, with the

exception of θ_1, which is of course periodic so this is not troublesome. Some m and c values in Table 4.3, corresponding to counterweight mass and radius respectively, are near the lower bound of zero. Since zero is a natural lower bound for both these variable types, this is also not a problem. For example, having an optimal counterweight mass of near zero simply suggests that the solution is very efficient, not that the lower bound needs to be extended (which is impossible anyway, as it would require negative mass). Thus the design space bounds appear sufficiently generous in all respects, which is in part due to the efficient formulation of the design variables in keeping with the general method of Chapter 2.

As these lists of numbers likely convey little understanding of qualitative aspects of the solution, some graphical representations are now presented, to scale and with axes in inches. Figures 4.16 and 4.17 show the linkage skeleton at BDC and TDC respectively. The gray area represents the footprint, calculated by the sweep of the link polygons (not shown for clarity). The axis of slide is the vertical dotted line near the top. The crank ground pivot is near the bottom left and the rocker's is near the bottom right. Link COM symbols are shown, drawn with areas proportional to the link masses. The largest one is the overall COM, drawn to the same scale. The black circles are centered on the link COMs and represent the radius of gyration of each link.

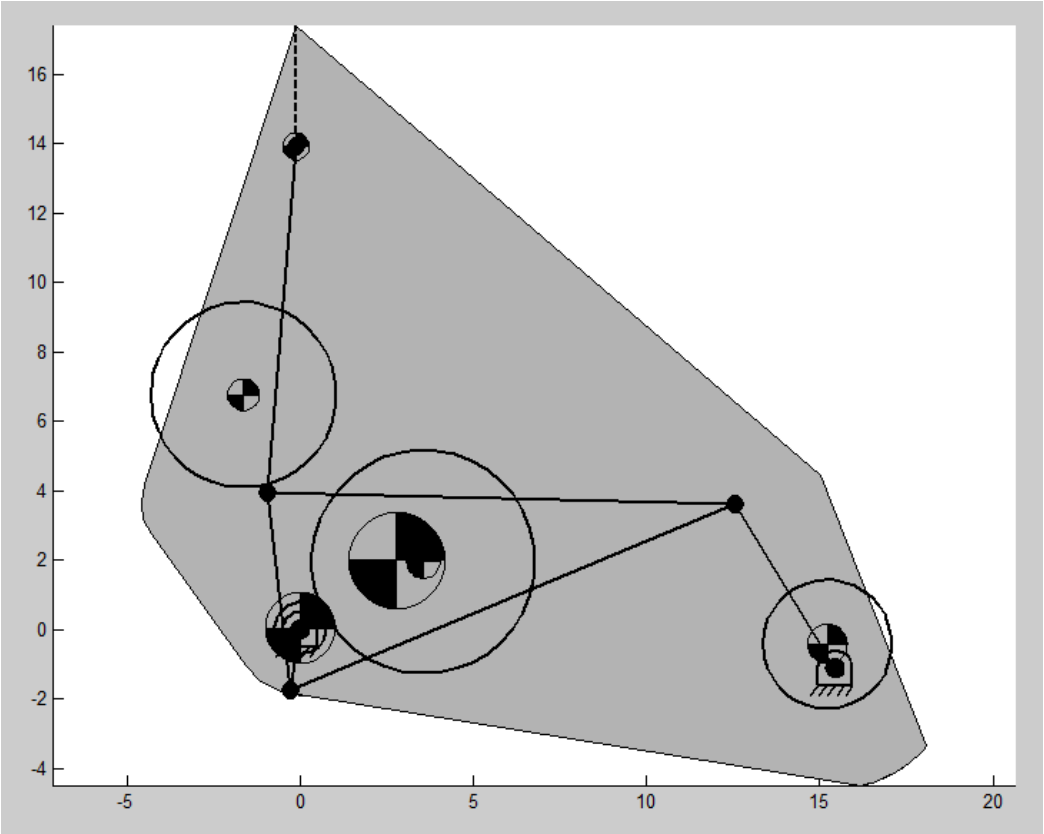


Figure 4.16: Linkage skeleton and mass properties at BDC

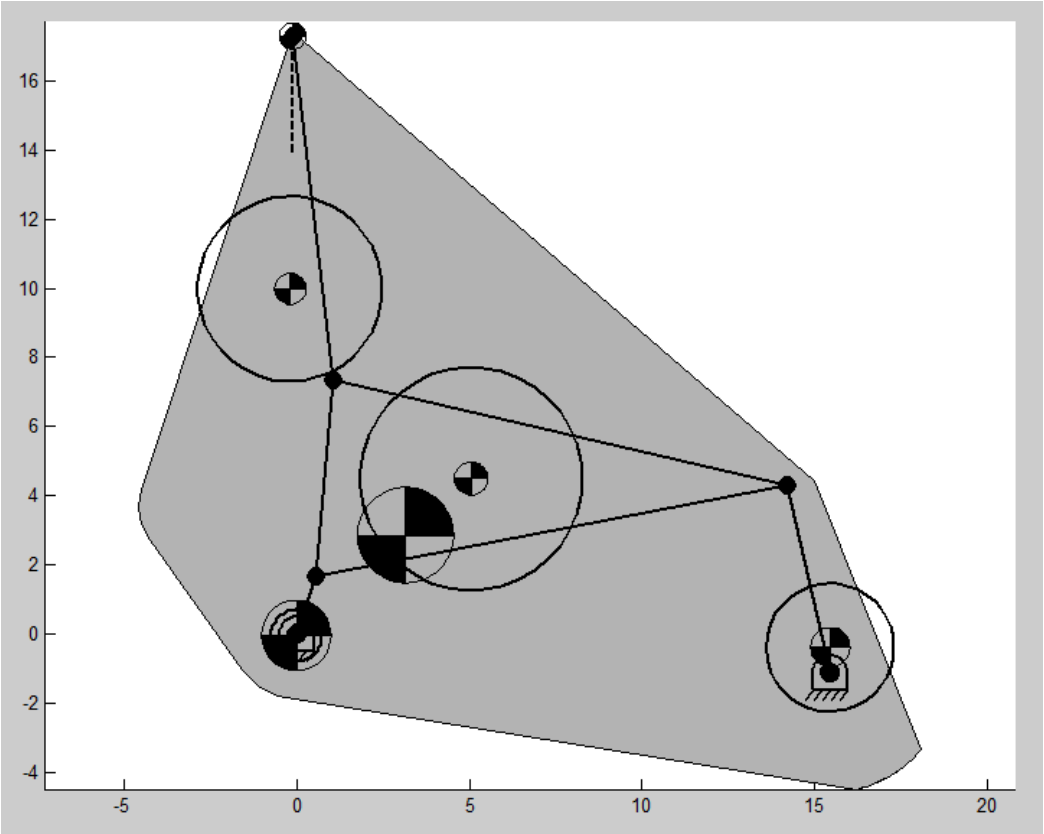


Figure 4.17: Linkage skeleton and mass properties at TDC

Figure 4.18 shows the skeleton and COM trajectories at all 20 positions analyzed. Note that the crank, rocker, and overall COMs exhibit very little motion.

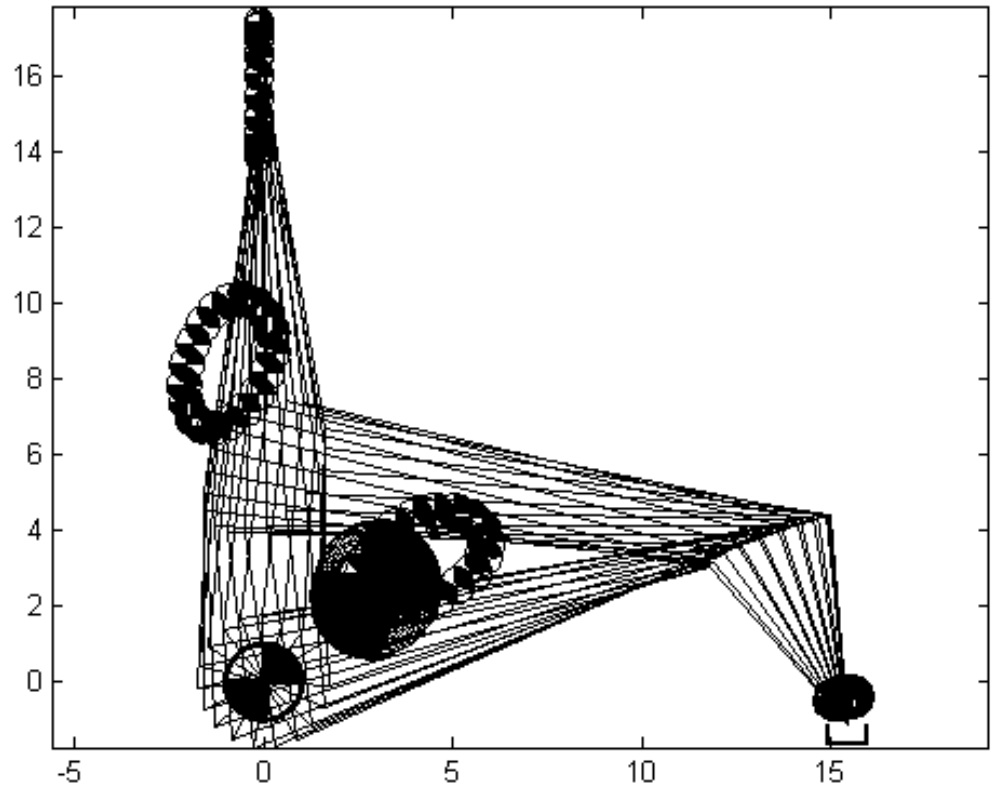


Figure 4.18: Linkage motion and COM trajectories

The linkage exhibits nearly sinusoidal piston motion. A plot of piston position above BDC in inches (blue) and gas force on the piston in thousands of lbf (red) is shown in Figure 4.19. The peak gas force is truncated due to the low resolution of the position analysis.

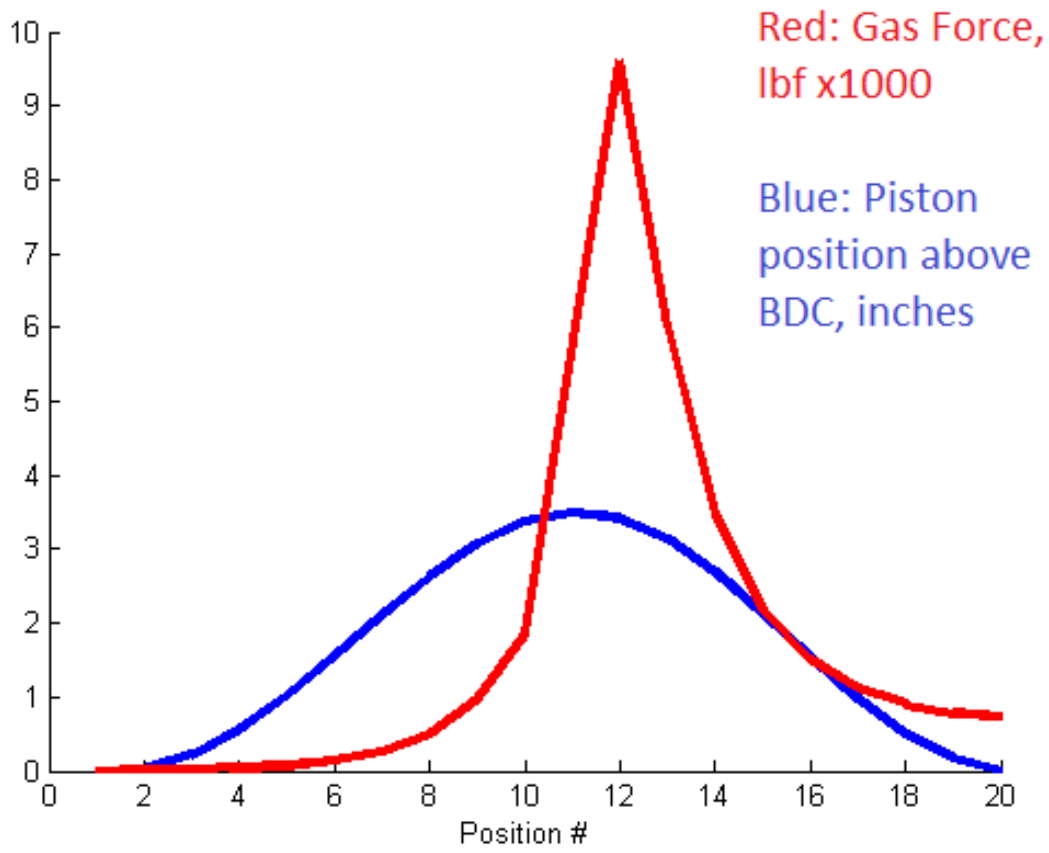


Figure 4.19: Piston trajectory and gas force profile

The dynamic forces on the linkage were post-processed in order to evaluate their neglect relative to applied forces. The dynamic analysis of the linkage is given in Appendix C, resulting in the dynamic loads on the joints shown in Figure 4.20. The axes represent x and y force components in lbf, with each line being a different position of the linkage.

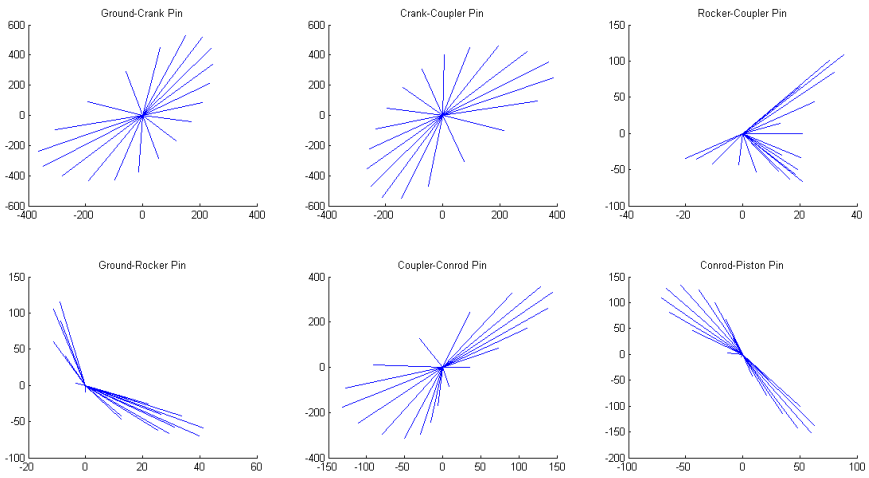


Figure 4.20: Polar plots of dynamic loading of joints

It can be seen that dynamic loads are on the order of a thousand lbf or less, while the peak gas force shown in Figure 4.19 is on the order of ten thousand lbf. Broadly speaking, then dynamic forces are of a lower order of magnitude then the applied force and their neglect is valid as a first approximation in determining link masses.

4.5.3 Combustion Model Limitations

These results must be interpreted in light of the original motivation for the use of a six-bar linkage for the IC engine: it was hoped that a non-sinusoidal piston trajectory would improve combustion efficiency. As a control case, an exact sinusoid was fed into the combustion simulator and resulted in a thermal efficiency of 39.5%. As the solution presented above has a substantially sinusoidal trajectory and an efficiency of 39.7%, clearly this goal was not met. Even if the efficiency objective were completely prioritized in the results set, only 42.4% is possible at the expense of severe degradation of the mechanical objectives.

There are only two logically possible reasons for this outcome: a better solution does not exist, or it does exist and the optimization failed to find it. In order to determine if there existed any possible piston trajectory that would result in a higher efficiency, a sim-

ple single-objective optimization using a basic genetic algorithm was undertaken. This optimization acted directly on the piston profile without reference to how it would be accomplished kinematically, by using the 20 points fed into the combustion subroutine as the design variables. Failure of this optimization to improve on the results of the main optimization would show that no possible piston trajectory would further improve efficiency and thus that the general method was not failing to find part of the Pareto-optimal set.

It proved necessary to make a slight change to the combustion model when conducting this direct optimization. A correlation had been used between mean piston velocity during the compression stroke and burn duration, reflecting the fact that increased turbulent mixing during compression would accelerate combustion. This was

$$D = 95 - .07S \tag{4.62}$$

where D is the burn duration in crank angle degrees (CAD) and S is the mean piston velocity in cm/sec between BDC and TDC during the compression stroke. Clearly this correlation cannot be valid for all S; in fact it will result in negative burn durations for $S > 1357$. This correlation is valid for the simple trajectories of traditional slider-cranks, but breaks down for some cases of the Stephenson-III that exhibit particularly fast compression. The mechanical objectives kept the piston trajectories well-behaved in the main optimization, but the totally unconstrained direct optimization would result in many cases with negative burn durations. Thus this correlation was removed and the burn duration D was fixed at the value for the sinusoidal control case, which was $D = 48.62$ CAD, corresponding to $S = 662.6$ cm/sec.

With this modification made, the direct optimization was executed. Convergence to a final value of 41.51% is shown in Figure 4.21, with the optimal trajectory (inches above BDC, blue) and corresponding gas force (lbf x1000, red) shown in Figure 4.22.

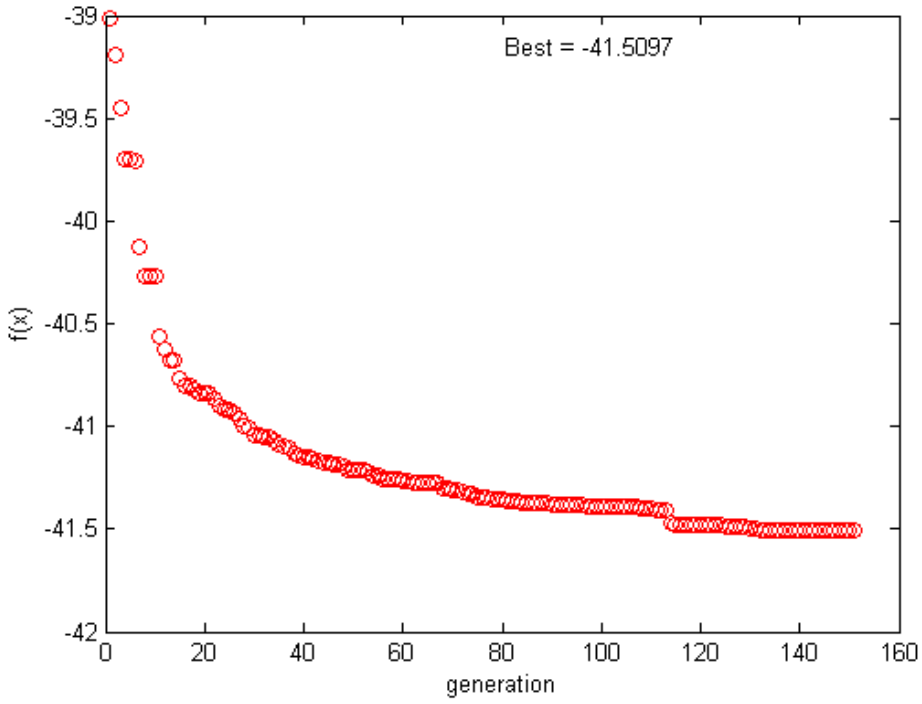


Figure 4.21: Convergence history

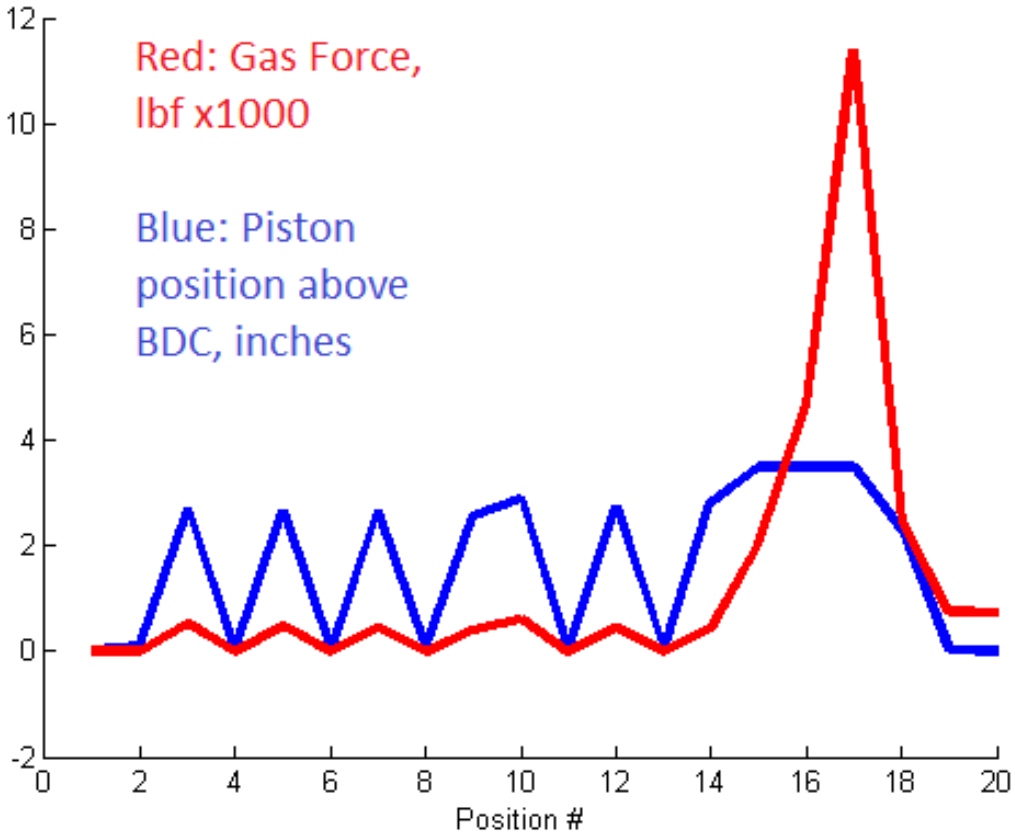


Figure 4.22: Piston trajectory and gas force profile

Up to about position 13 is just noise, since the model is frictionless and isentropic it makes no difference what the piston does before ignition. Peak force and efficiency are marginally higher than the control case.

Just to confirm that the noise was irrelevant, ignition was fixed at position 10 (since there are no kinematics, the phase location of ignition was not important), and the first 9 points were set to zero. Convergence to a final value of -41.1776% is shown in Figure 4.23, with the optimal trajectory (inches above BDC, blue) and corresponding gas force (lbf x1000, red) shown in Figure 4.24.

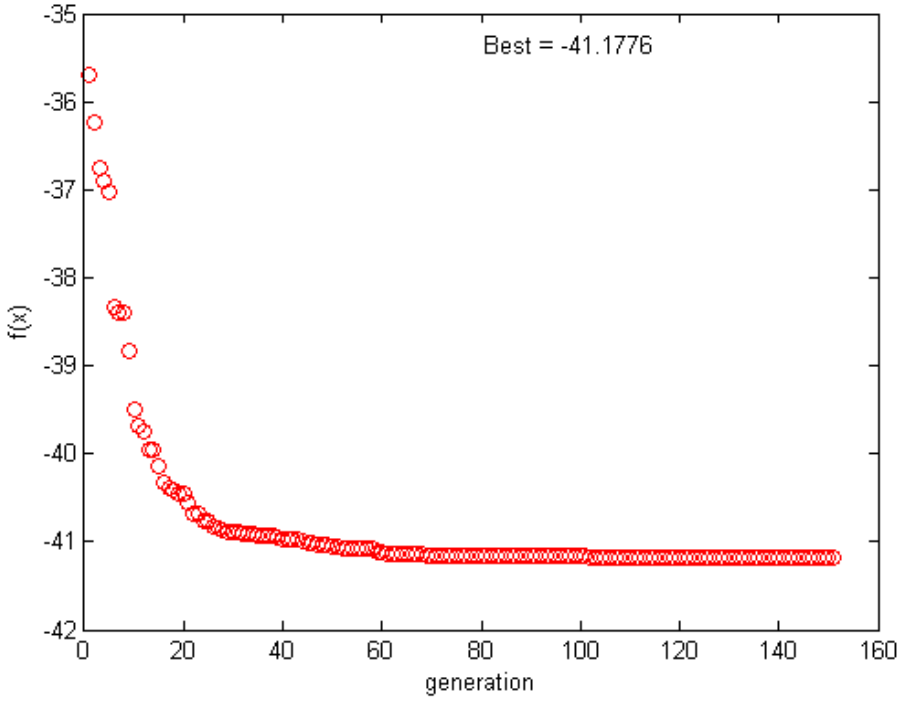


Figure 4.23: Convergence history

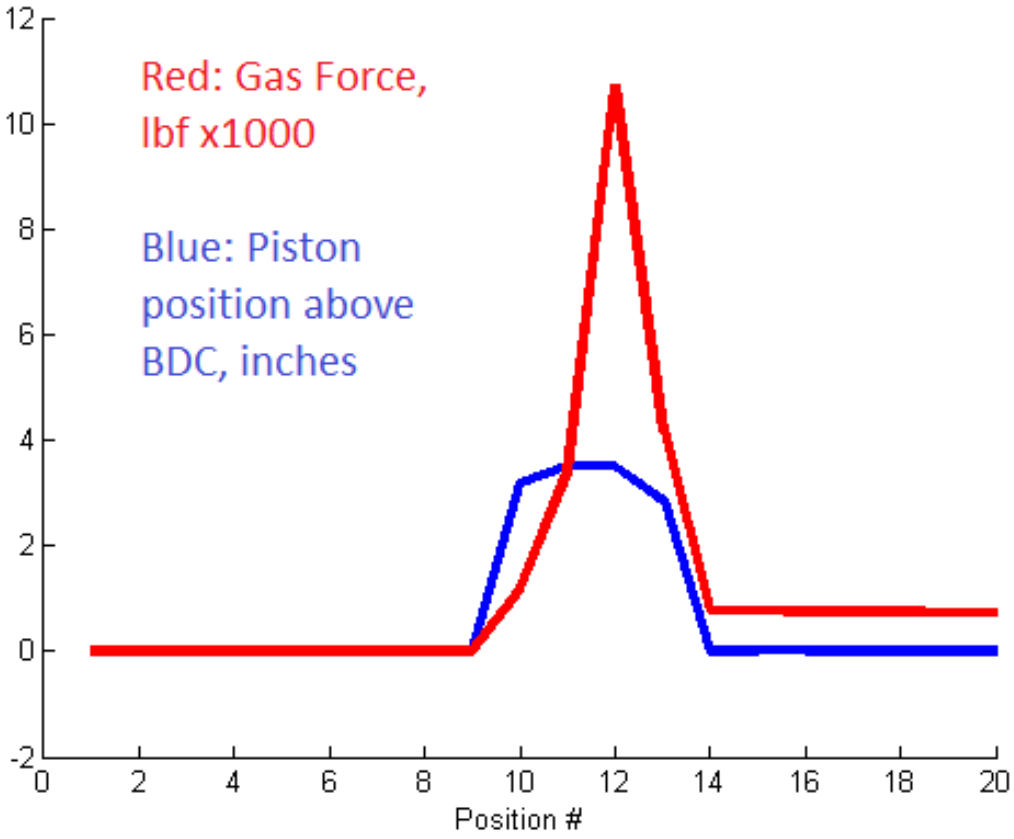


Figure 4.24: Piston trajectory and gas force profile

The two results are clearly essentially the same except for the noise and ignition point. Also of note is the fact that the final efficiency is essentially the same in the convergence plots, and that there is indeed a short dwell after ignition, as was anticipated.

As a supplementary measure, 32,000 random linkages were generated using a uniform distribution on the bounds of Table 4.2 and their efficiencies evaluated. A histogram of the results is shown in Figure 4.25.

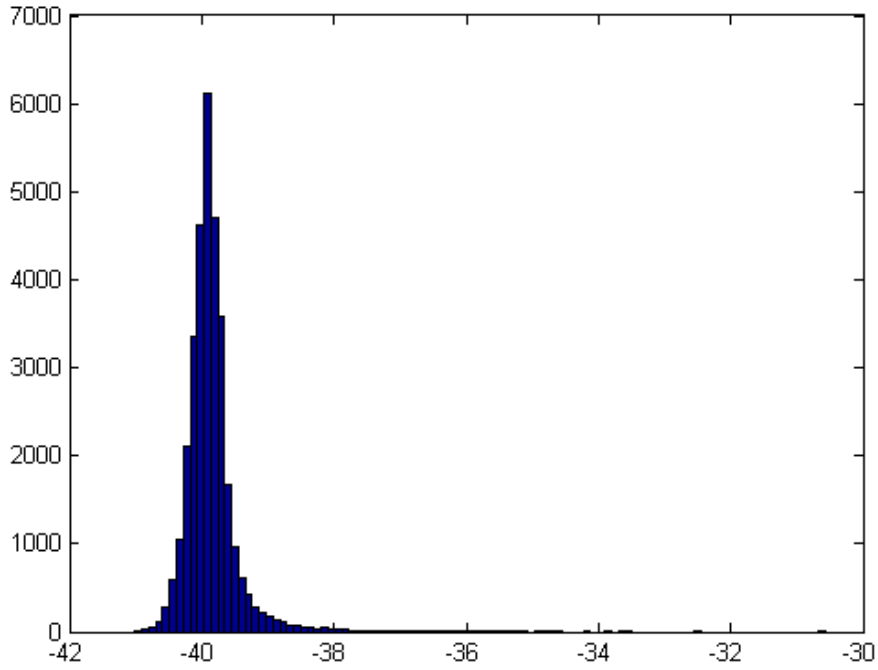


Figure 4.25: Histogram of efficiencies of randomly generated linkages

The result looks roughly like a normal distribution centered around the control case efficiency of 39.5%, which is quite reasonable. The best efficiency in the entire sample was 41.0%, which is entirely consistent with the best efficiencies in the genetic optimizations above. Therefore it can be concluded that the best possible efficiency given the combustion model used (without the turbulence correlation) is slightly over 41%. In cases where the correlation is valid, a slight improvement is possible, explaining the slightly higher value of 42.4% from the main optimization, but improvement of more than a few percentage points over the control case is not possible regardless of what kinematics are used.

Nonetheless, the optimization still produced a very high quality solution to the problem given this limitation. A light, compact linkage was found, with efficiency near the model's maximum possible value, with shaking force and moment an order of magnitude below what engine mounts are designed to accomodate, and a maximum piston side-load of only

168 lbf that is well within the capabilities of piston rings to handle. The moderate to favorable outcomes in all six objectives for a generalized formulation of a very complex multi-domain problem should not be ignored, even if the combustion model limitations excluded attainment of the desired goal of higher thermal efficiency.

Chapter 5:

Conclusion

After a review of the work presented in this thesis, the implications of this work are discussed and recommendations for further development are given.

5.1 Review

Chapter 1 began with a discussion of the impact of modern computing capabilities on the practice of engineering design, particularly on the shift from analysis to optimization that is made possible. The literature relating to a number of fields relevant to this thesis was then reviewed. These fields were general multi-objective optimization, mechanism-specific optimization, the optimization of elastic structures, and the use of six-bar linkages in engine design. The latter two are specific to the two case studies treated here, so emphasis should be given to the former two fields. From these, it was concluded that recent developments in multi-objective genetic algorithms have the potential to revolutionize mechanism design and optimization techniques, and that while a number of authors have begun to exploit the new capabilities, the approach taken in this thesis is novel to the best of the author's knowledge.

Chapter 2 then presented this general approach, beginning with an introduction to the

underlying concepts of optimization, Pareto-optimality, and the workings of the NSGA-II algorithm. The drawbacks to the naive approach to mechanism optimization were studied, and a three-part strategy to avoid these drawbacks was given, consisting of indirect design variable specification, reduction of the design space by preliminary application of traditional kinematic techniques, and the use of a nested optimization structure. This nested structure necessitated the introduction of a modification to the standard NSGA-II algorithm, which was also summarized.

With the general method presented, two case studies were explored in detail to illustrate its potential. Chapter 3 concerned the optimization of a hydraulic rescue spreader. The objectives were to minimize overall mass and length of the mechanism, as well as to minimize the required jaw rotation and variation in spreading force. This optimization used the standard NSGA-II at the outer level and a nonlinear programming algorithm operating on a finite-element-based structural analysis at the inner level. Favorable solutions were obtained, with the final manual selection from the Pareto-optimal set equaling or exceeding the industry standard in all objectives.

The second case study was then given in Chapter 4, relating to the use of a Stephenson-III six-bar linkage to replace the traditional slider-crank in an IC engine, in hopes of achieving a better piston trajectory. The objectives were maximization of the thermal efficiency of the combustion process, as well as minimization of piston side-loading, minimization of shaking force and moment, and minimization of overall mass and footprint. In constast to the first case study, a multi-objective approach at both levels was used, requiring employment of the modified NSGA-II algorithm. While a mechanically satisfactory solution that matches the thermal efficiency of the traditional design was found, the desired improvement in thermal efficiency was not attained, due to limitations imposed by the simplicity of the combustion model.

5.2 Conclusions

As was discussed in the literature review, while a number of authors have begun to apply the relatively new capabilities of multi-objective genetic algorithms to the optimization of mechanisms from a kinematic standpoint, there is a surprising dearth of attempts to use these new capabilities to move beyond the traditional kinematic and dynamic considerations to a broader multi-domain approach. Solution of a multi-objective, multi-domain problem is often a complex and computationally expensive undertaking, so efficient and robust formulation of the optimization is critical. It was shown that in many cases, the dimension of the design space can be reduced by preliminary application of more traditional kinematic analysis, and through normalization considering fixed length scales in the problem. Furthermore, transformation of design variables can in many cases result in switching from arbitrary or intuitive bounds on the design space to non-arbitrary or natural bounds, reducing the size of the search space and precluding the accidental exclusion of optima. Finally, a nested optimization structure was proposed in which design variables and objectives are partitioned into kinematic and non-kinematic categories, allowing optimization of objectives from all the disciplines involved in terms of the kinematic design variables at the outer level and promoting the stability and robustness of the optimization.

Application of this methodology to the optimization of a hydraulic rescue spreader, such as that shown in Figure 5.1, resulted in a solution equal or superior to the new SP-310 model from market-leading manufacturer Hurst in all objectives.

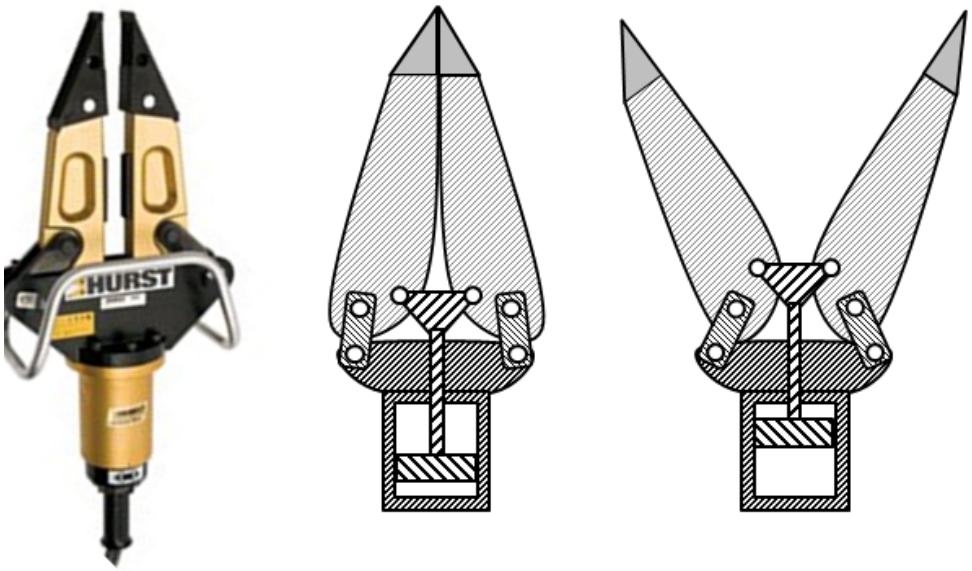


Figure 5.1: A typical rescue spreader and motion schematic [2]

Application of shape optimization techniques from the literature in conjunction with integrated finite element analysis resulted in a lower estimated overall mass while retaining structural integrity. Simultaneously, reduction of the dimension of the optimization problem using traditional motion generation analysis combined with design variable transformation and numerical optimization allowed significant increase in spreading force, decrease in force variation at the start and end positions, and reduced jaw rotation for better grip, without increasing overall mechanism length. Tabular and graphical comparison of the optimized mechanism to the SP-310 is found in Table 5.2 and Figure 5.2. The achievement of both structural and kinematic improvements demonstrates the power of the multi-domain approach developed in this thesis.

Table 5.1: Comparison of objectives values between result and Hurst SP-300 model [89]

	Optimized	Hurst
Mass - lbs	31.08	37.70
Length - inches	29.96	29.53
Jaw Rotation - deg	36.25	54.00
LSF - lbf	10,000	7,419
HSF - lbf	10,704	8,992

A side-by-side graphical comparison is shown in Figure 5.2.

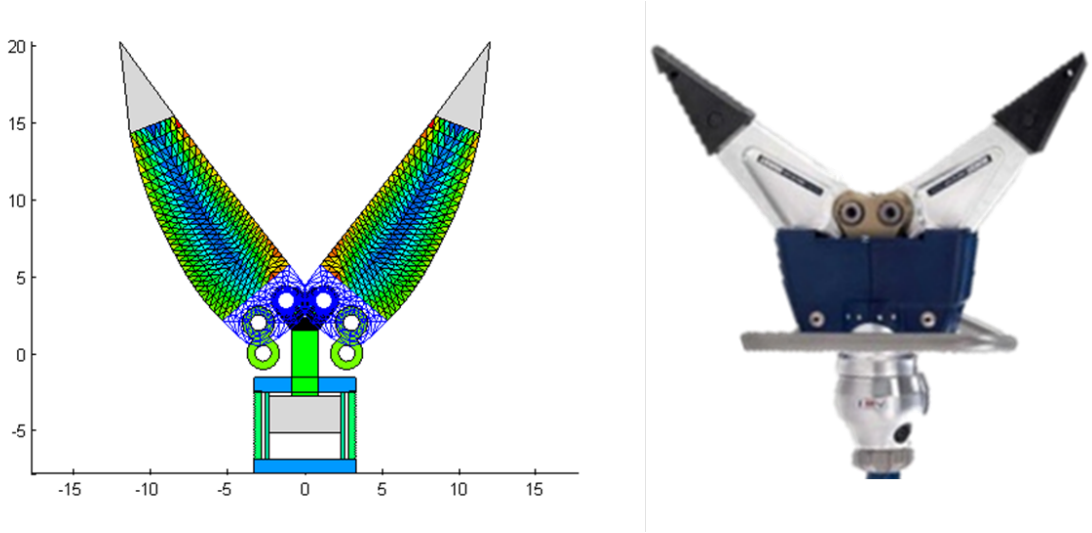


Figure 5.2: Graphical comparison of result to Hurst SP-300 model [89]

This methodology is flexible as well as powerful. The optimization of a slow-moving handheld tool considering the interdependence of kinematic and structural effects is summarized above; also successful was the optimization of a high-speed linkage for an internal combustion engine. Using the Stephenson-III six-bar linkage shown in Figure 5.3 in place of the traditional slider-crank, thermodynamic, kinematic, and dynamic objectives were optimized simultaneously.

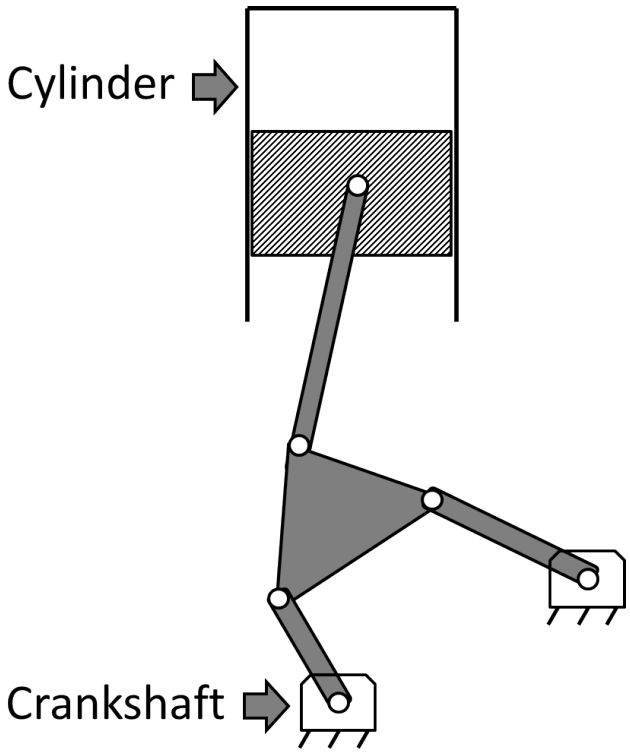


Figure 5.3: Stephenson-III linkage for an internal combustion engine

Despite the specification of no less than six mutually conflicting objectives, a solution satisfactory in all respects was obtained. Table 5.2 lists the objectives and their optimized values. Figure 5.4 shows the optimized linkage in the Top Dead Center position with the link centers of mass and radii of gyration as well as the overall center of mass of the linkage displayed graphically. The areas of the center of mass symbols are proportional to the masses they represent, and the figure is to scale with the axes in inches. The linkage motion and center of mass trajectories are illustrated in Figure 5.5. The crank and rocker centers of mass are essentially stationary, with the connecting rod and coupler trajectories optimized to minimize motion of the overall center of mass.

Table 5.2: Objective values for selected solution

Objective	Value
Piston Side-Load, lbf	168
% Thermal Efficiency (-)	-39.7
RMS Shaking Force, lbf	178
RMS Shaking Moment, in-lbf	130
Total Linkage Mass, lbm	5.3
Planar Footprint, in^2	273.4

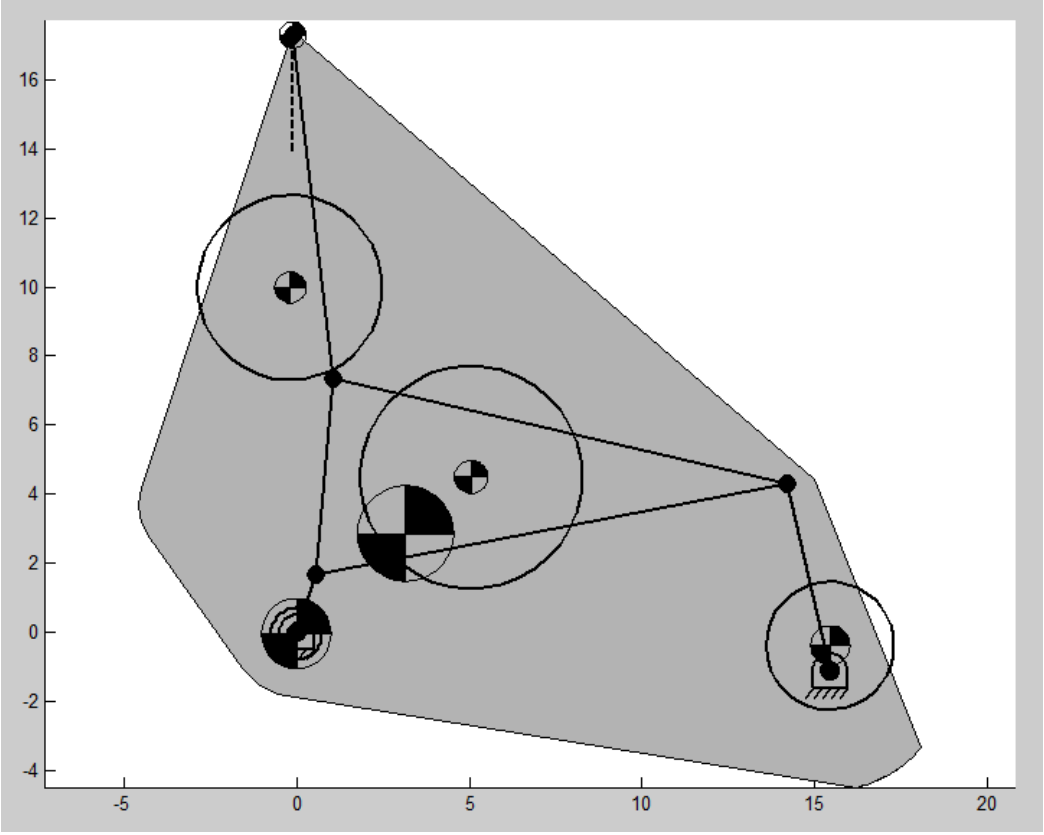


Figure 5.4: Linkage skeleton and mass properties at TDC

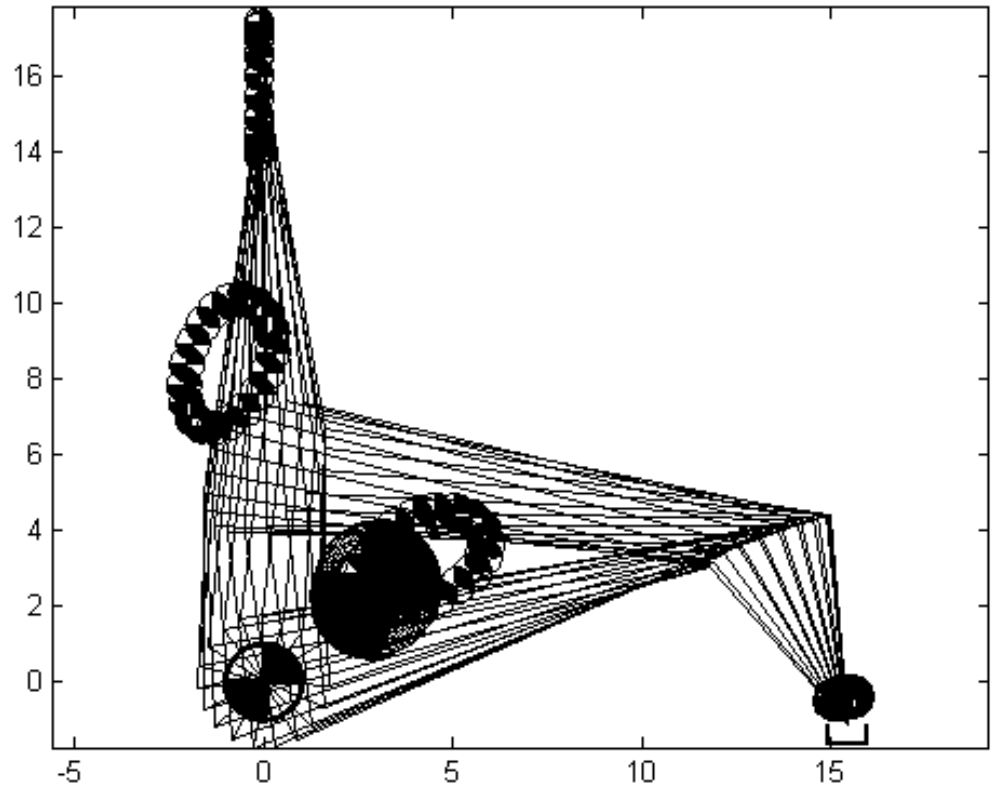


Figure 5.5: Linkage motion and COM trajectories

A relatively light, compact linkage was found, with shaking force and moment an order of magnitude below what engine mounts are designed to accomodate, and a maximum piston side-load of only 168 lbf that is well within the capabilities of piston rings to handle. The thermal efficiency was within a few percentage points of the maximum value attainable given the combustion model used, as discussed in Chapter 4.

5.3 Future Work

Notwithstanding the favorable results obtained in both cases, there is still a great deal of room for future improvements. The second case study would benefit from re-optimization

using an improved combustion model that could address the difficulties dicussed at the end of Chapter 4 relating to turbulent mixing and handling of non-traditional kinematics, in order to further improve the thermal efficiency of the solution. It would also be possible to include iterative structural analysis considering the dynamic as well as applied loading of the links, possibly in the form of a shape optimization using dynamic FEA.

Considering the work presented here more broadly, the general method would greatly benefit from systematic investigation of optimal formulation of design variables. It would make sense to start with the most commonly used inversions and transformations of the four-bar linkage and the two six-bar isomers, so that if one were optimizing (for instance) a pin-jointed Watt-II six-bar the choice of design variables that provided the best robustness and natural bounding of the design space could be simply looked up in a table. A major benefit of this tabulation of formulations would be realized if it were used in conjunction with the addition of type synthesis to the optimization method. The ability of genetic algorithms to operate on a mixture of discrete and continuous variables makes them a perfect choice for a combined type-dimensional optimization; for instance see the work of Fang [34]. If a lookup table of best-practice design variable formulations were available to the optimizer, a variety of linkage topologies could be explored in an integrated fashion without the need for the designer to pre-define formulations each time for all possible topological options. These best-practice formulations would also aim to have as many design variables as possible naturally bounded above and below, so that arbitrary bounds need not be selected and arbitrary problem-specific formulation choices would be kept to a minimum.

Inclusion of type synthesis would increase the already considerable computational expense. Another merit of genetic algorithms is that the evaluation of the individuals in each generation can be done in parallel, so that large reductions in computation time can be achieved by parallelization. This could also be done in a highly standardized way since the structure of objective function calls on each individual in a generation does not change between different optimizations.

A better means of conducting a nested optimization that is multi-objective at both levels is also needed. The modified NSGA-II algorithm used here is able to produce a reasonable result, but was essentially an *ad hoc* or stopgap measure to produce the required capability quickly and it is very likely that significantly better ways of accomplishing this task can be found. Since it is the fact that the outer level must deal with objective function calls that produce Pareto fronts rather than single-valued objective vectors that precludes the use of standard techniques, it may be the case that this problem can be solved by some combination of *a priori* and *a posteriori* methods at the two levels.

Finally, the development of standardized multi-domain analysis modules designed to interface with the standardized kinematic formulations would create a very powerful capability. For example, if one were optimizing a hydraulic pump, one would designate one of the links as the piston (or potentially set link selection as a discrete variable to be optimized), and the pumping module would accept piston trajectory data from the standardized kinematic position analysis and return data relating to force, efficiency, or whatever else might be required. Modules could be combined flexibly; the pump linkage might also apply a structural FEA module to the crank, for example.

Even without these envisioned improvements, the method developed in this thesis has proven effective at solving two challenging and substantially different problems in multi-domain, multi-objective mechanism optimization. It is expected that application of the formulation strategies of Chapter 2 will help designers solve a wide variety of other optimization problems, taking much of the guesswork and intuition out of complex multi-domain mechanism design tasks and presenting explicit trade-off options between competing objectives through the examination of the Pareto-optimal sets returned by the optimizer. In summary, the integration of traditional mechanism design, multi-domain analysis, and multi-objective optimization has resulted in a powerful and flexible approach to the solution of a variety of real-world design problems.

Bibliography

[1] A. Erdman, G. Sandor, S. Kota, Mechanism Design: Analysis and Synthesis, 4th Edition, Vol. 1, Prentice-Hall, New Jersey, 1997.

[2] Photo from hurst website.
URL `http://www.jawsoflife.com`

[3] J. Andersson, A survey of multiobjective optimization in engineering design, Reports of the Department of Mechanical Engineering, LiTH-IKP.

[4] R. T. Marler, J. S. Arora, Survey of multi-objective optimization methods for engineering, Structural and Multi-Disciplinary Optimization 26 (6) (2004) 369–395.

[5] W. Stadler, Fundamentals of multicriteria optimization, in: Multicriteria Optimization in Engineering and in the Sciences, Plenum Press, New York, NY, 1988, pp. 1–25.

[6] R. Benayoun, J. De Montgolfier, J. Tergny, O. Laritchev, Linear programming with multiple objective functions: Step method (stem), Mathematical programming 1 (1) (1971) 366–375.

[7] S. R. E., C. R. U., An interactive weighted tchebycheff procedure for multiple onjective programming, Mathematical Programming 26 (1983) 326–344.

[8] J. D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, in: Proceedings of the 1st international Conference on Genetic Algorithms, L. Erlbaum Associates Inc., 1985, pp. 93–100.

[9] M. P. Fourman, Compaction of symbolic layout using genetic algorithms, in: Proceedings of the 1st International Conference on Genetic Algorithms, L. Erlbaum Associates Inc., 1985, pp. 141–153.

[10] G. D., Genetic Algorithms in Search and Machine Learning, Addison Wesley, Reading, MA, 1989.

[11] S. N., D. K., Multiobjective optimization using nondominated sorting in genetic algorithms, Evolutionary Computation 2 (1995) 221–248.

[12] K. Deb, A. Pratap, S. Agarwal, T. A. M. T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, Evolutionary Computation, IEEE Transactions on 6 (2) (2002) 182–197.

[13] H. J., N. N., Multiobjective optimization using the niched pareto genetic algorithm, Technical Report 93005, Illinois Genetic Algorithm Laboratory, Dept. of General Engineering, University of Illinois at Urbana- Champaign, Urbana, USA (1993).

[14] F. C. M., F. P. J., Multiobjective optimization and multiple constraint handling with evolutionary algorithms - part i: a unified formulation, IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans 28 (1998) 26–37.

[15] Z. E., T. L., Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach, IEEE Transactions on evolutionary computation 3 (1999) 257–271.

[16] C. C. Coello, R. L. Becerra, Evolutionary multiobjective optimization using a cultural algorithm, in: Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE, IEEE, 2003, pp. 6–13.

[17] A. Toffolo, E. Benini, Genetic diversity as an objective in multi-objective evolutionary algorithms, Evolutionary Computation 11 (2) (2003) 151–167.

[18] G. N. Sandor, A general complex number method for plane kinematic synthesis with applications, Ph.D. thesis, Columbia University, New York, NY (1959).

[19] A. G. Erdman, Three and four precision point kinematic synthesis of planar linkages, Mechanism and Machine Theory 16 (5) (1981) 227–245.

[20] R. E. Kaufman, Mechanism design by computer, Machine Design Magazine (1978) 94–100.

[21] R. J. Loerch, A. G. Erdman, G. N. Sandor, A. Mihda, Synthesis of four bar linkages with specified ground pivots, in: Proceedings of 4th Applied Mechanisms Conference, Chicago, IL, 1975, pp. 101–106.

[22] S. N. Kramer, G. N. Sandor, Selective precision synthesis-a general method of optimization for planar mechanisms, Journal of Engineering for Industry 97 (1975) 689.

[23] J. A. Mirth, Quasi-precision synthesis of four-bar linkages, in: Proc. of 23rd Biennial Mechanisms Conference, Minneapolis, MN, 1994, p. 215.

[24] R. L. Fox, K. D. Willmert, Optimum design of curve-generating linkages with inequality constraints, Journal of Engineering for Industry 89.

[25] A. Youssef, K. Oldham, L. Maunder, Optimal kinematic synthesis for planar linkage mechanisms, in: Proceedings of the I MECH E, 1975, pp. 393–398.

[26] F. L. Conte, G. R. George, R. W. Mayne, J. P. Sadler, Optimum mechanism design combining kinematic and dynamic-force considerations, Journal of Engineering for Industry 97 (1975) 662.

[27] A. J. Kakatsios, S. J. Tricamo, Integrated kinematic and dynamic optimal design of flexible planar mechanisms, ASME, 1987.

[28] S. S. Rao, R. L. Kaplan, Optimal balancing of high-speed linkages using multiobjective programming techniques, ASME, 1986.

[29] S. Krishnamurty, D. A. Turcic, Optimal synthesis of mechanisms using nonlinear goal programming techniques, Mechanism and Machine Theory 27 (5) (1992) 599–612.

[30] A. Kunjur, S. Krishnamurty, Genetic algorithms in mechanical synthesis, Journal of Applied Mechanism and Robotics 4 (2) (1997) 18–24.

[31] J. A. Cabrera, A. Simon, M. Prado, Optimal synthesis of mechanisms with genetic algorithms, Mechanism and Machine Theory 37 (2002) 1165–1177.

[32] M. A. Laribi, A. Mlike, L. Romdhane, S. Zeghloul, A combined genetic algorithm-fuzzy logic method (ga-fl) in mechanism synthesis, Mechanism and Machine Theory 39 (2004) 717–735.

[33] A. Kanarachos, D. Koulocheris, H. Vruzopoulos, Evolutionary algorithms with deterministic mutation operators used for the optimization of the trajectory of a four-bar mechanism, Mathematics and Computers in Simulation 63 (2003) 483–492.

[34] W. E. Fang, Simultaneous type and dimensional synthesis of mechanisms by genetic algorithms, Mechanism Synthesis and Analysis 70 (1994) 35–41.

[35] H. Zhou, E. H. M. Cheung, Adjustable four-bar linkages for multi-phase motion generation, Mechanism and Machine Theory 39 (2004) 261–279.

[36] J. A. Cabrera, F. Nadal, J. P. Munoz, A. Simon, Multiobjective constrained optimal synthesis of planar mechanisms using a new evolutionary algorithm, Mechanism and Machine Theory 42 (7) (2007) 791–806.

[37] N. Nariman-Zadeh, M. Felezi, A. Jamali, M. Ganji, Pareto optimal synthesis of four-bar mechanisms for path generation, Mechanism and Machine Theory 44 (1) (2009) 180–191.

[38] M. Khorshidi, M. Soheilypour, M. Peyro, A. Atai, M. Shariat Panahi, Optimal design

of four-bar mechanisms using a hybrid multi-objective ga with adaptive local search, Mechanism and Machine Theory 46 (10) (2011) 1453–1465.

[39] B. El-Kribi, A. Houidi, Z. Affi, L. Romdhane, Application of multi-objective genetic algorithms to the mechatronic design of a four bar system with continuous and discrete variables, Mechanism and Machine Theory 61 (2013) 68–83.

[40] H. K. Fathy, J. A. Reyer, P. Y. Papalambros, A. G. Ulsov, On the coupling between the plant and controller optimization problems, in: Proceedings of the 2001 American Control Conference, 2001, pp. 1864–1869.

[41] K. Saitou, K. Izui, S. Nishiwak, P. Y. Papalambros, A survey of structural optimization in mechanical product development, ASME, 2005.

[42] J. A. Samareh, Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization, AIAA journal 39 (5) (2001) 877–884.

[43] C. Dyken, M. S. Floater, Transfinite mean value interpolation, Computer Aided Geometric Design 26 (1) (2009) 117–134.

[44] A. L. Gaitonde, S. P. Fiddes, A three-dimensional moving mesh method for the calculation of unsteady transonic flows, Aeronautical Journal 99 (984) (1995) 150–160.

[45] B. K. Soni, Two-and three-dimensional grid generation for internal flow applications of computational fluid dynamics, AIAA paper (1985) 85–1526.

[46] M. E. Botkin, Three-dimensional shape optimization using fully automatic mesh generation, AIAA journal 30 (7) (1992) 1932–1934.

[47] J. T. Batina, Unsteady euler airfoil solutions using unstructured dynamic meshes, AIAA journal 28 (8) (1990) 1381–1388.

[48] G. Chiandussi, G. Bugeda, E. Onate, A simple method for automatic update of finite element meshes, Communications in Numerical Methods in Engineering 16 (1) (2000) 1–19.

[49] J. Forsberg, L. Nilsson, On polynomial response surfaces and kriging for use in structural optimization of crashworthiness, Structural and multidisciplinary optimization 29 (3) (2005) 232–243.

[50] N. Lyu, J. Park, H. Urabe, H. Tokunaga, K. Saitou, Design of Automotive Torsion Beam Suspension Using Lumped-Compliance Linkage Models, ASME, 2006.

[51] U. Kirsch, P. Y. Papalambros, Structural reanalysis for topological modifications-a unified approach, Structural and Multidisciplinary Optimization 21 (5) (2001) 333–344.

[52] U. Kirsch, P. Y. Papalambros, Exact and accurate reanalysis of structures for geometrical changes, Engineering with Computers 17 (4) (2001) 363–372.

[53] T. Sullivan, J. D. Van de Ven, Topological synthesis and integrated kinematic-structural optimization of a ten-bar linkage for a hydraulic rescue spreader, in: ASME 2013 International Design Engineering Technical Conference, Portland, OR, 2013.

[54] P. Y. Papalambros, K. Shea, Creating structural configurations, Cambridge University Press, New York, NY, 2002.

[55] A. D. Belegundu, S. D. Rajan, A shape optimization approach based on natural design variables and shape functions, Computer Methods in Applied Mechanics and Engineering 66 (1) (1988) 87–106.

[56] H. Azegami, M. Shimoda, E. Katamine, Z. C. Wu, A domain optimization technique for elliptic boundary value problems, Computer Aided Optimization Design of Structures IV, Structural Optimization (1995) 51–58.

[57] R. T. Haftka, R. V. Grandhi, Structural shape optimization-a survey, Computer Methods in Applied Mechanics and Engineering 57 (1) (1986) 91–106.

[58] M. Scherer, R. Denzer, P. Steinmann, A fictitious energy approach for shape optimization, International Journal for Numerical Methods in Engineering 82 (3) (2010) 269–302.

[59] S. Arnout, M. Firl, K. U. Bletzinger, Parameter free shape and thickness optimisation considering stress response, Structural and Multidisciplinary Optimization 45 (6) (2012) 801–814.

[60] R. M. Pickett, M. Rubinstein, R. B. Nelson, Automated structural synthesis using a reduced number of design coordinates, AIAA Journal 11 (4) (1973) 489–494.

[61] E. S. Kristensen, N. F. Madsen, On the optimum shape of fillets in plates subjected to multiple in-plane loading cases, International Journal for Numerical Methods in Engineering 10 (5) (1976) 1007–1019.

[62] M. I. Bloor, M. J. Wilson, Efficient parametrization of generic aircraft geometry, Journal of Aircraft 32 (6) (1995) 1269–1275.

[63] A. H. Barr, Global and local deformations of solid primitives, ACM Siggraph Computer Graphics 18 (3) (1984) 21–30.

[64] T. W. Sederberg, S. R. Parry, Free-form deformation of solid geometric models, In ACM Siggraph Computer Graphics 20 (4) (1986) 151–160.

[65] S. Chen, D. A. Torterelli, Three-dimensional shape optimization with variational geometry, Structural optimization 13 (2-3) (1997) 81–94.

[66] S. S. Bhavikatti, C. V. Ramakrishnan, Optimum design of fillets in flat and round tension bars, in: Proceedings of the ASME 1977 Design Engineering Technical Conference, Chicago, Illinois, 1977.

[67] V. Braibant, C. Fleury, Shape optimal design using b-splines, Computer Methods in Applied Mechanics and Engineering 44 (3) (1984) 247–267.

[68] U. Schramm, W. D. Pilkey, R. I. DeVries, M. P. Zebrowski, Shape design for thin-walled beam cross sections using rational b splines, AIAA journal 33 (11) (1995) 2205–2211.

[69] H. N. Pouliot, W. R. Delameter, C. W. Robinson, Variable-displacement spark-ignition engine, Tech. Rep. CONF-770205-69, Society of Automotive Engineers, Warrendale, PA (1977).

[70] F. Freudenstein, E. Maki, Development of an optimum variable-stroke internal-combustion engine mechanism from the viewpoint of kinematic structure, ASME, 1983.

[71] H. Ozcan, J. A. Yamin, Performance and emission characteristics of lpg powered four stroke si engine under variable stroke length and compression ratio, Energy Conversion and Management 49 (5) (2008) 1193–1201.

[72] J. A. Yamin, M. H. Dado, Performance simulation of a four-stroke engine with variable stroke-length and compression ratio, Applied energy 77 (4) (2004) 447–463.

[73] T. A. Good, Optimum non-uniform stroke piston engine synthesis, Master's thesis, Lehigh University, Bethlehem, PA (1995).

[74] S. S. Balli, S. Chand, Transmission angle in mechanisms (triangle in mech), Mechanism and Machine Theory 37 (2) (2002) 175–195.

[75] A. G. Erdman, G. N. Sandor, Mechanism Design: Analysis and Synthesis, 1st Edition, Vol. 2, Prentice-Hall, 1984.

[76] Hurst rescue products catalog: High pressure.
URL `http://www.jawsoflife.com/Downloads/Default.aspx`

[77] Hurst rescue products catalog: 5,000 psi.

URL http://www.jawsoflife.com/Downloads/Default.aspx

[78] Holmatro website.

URL http://www.holmatro-usa.com

[79] Genesis rescue systems website.

URL http://www.genesisrescue.com/html/spreaderss49xl.php

[80] R. C. Juvinall, K. M. Marshek, Fundamentals of Machine Component Design, John Wiley and Sons, Hoboken, NJ, 2006.

[81] G. Farin, Curves and Surfaces for CAGD: A Practical Guide, Academic Press, London, UK, 2002.

[82] J. Gallier, Curves and Surfaces in Geometric Modeling: Theory and Algorithms, Morgan Kaufmann Publishers, San Francisco, CA, 2000.

[83] H. P. Langtangen, Computational Partial Differential Equations, Springer-Verlag, New York, NY, 2003.

[84] M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, Computational Geometry: Algorithms and Applications, Springer-Verlag, Santa Clara, CA, 2008.

[85] J. O'Rourke, Computational Geometry in C, Cambridge University Press, Cambridge, UK, 1998.

[86] NFPA Standard On Powered Rescue Tools, 2010 Edition, Chap. 7-8, Quincy, MA.

[87] F. Cardarelli, Materials Handbook, Springer, London, UK, 2000.

[88] D. J. Walker, R. M. Everson, J. E. Fieldsend, Visualising mutually non-dominating solution sets in many-objective optimisation, IEEE Transactions on Evolutionary Computation 17 (2) (2013) 165–184.

[89] Sp-300 photo and data from hurst website.
URL http://www.jawsoflife.com/en/product/10000-psi-sp300-spreader

[90] R. L. Norton, Design of machinery: An Introduction to the Synthesis and Analysis of Mechanisms and Machines, 5th Edition, McGraw-Hill, New York, NY, 2012.

[91] K. Bindiganavle, An optimal approach to geometric trimming of b-spline surfaces, Master's thesis, Virginia Polytechnic Institute, Blacksburg, VA (2000).

[92] G. A. Mohr, Finite elements for solids, fluids, and optimization, Oxford University Press, New York, NY, 1992.

Appendices

Appendix A:

B-Splines

This appendix serves as an introduction to the mathematics of the B-spline curves used to parameterize the part boundaries in the inner optimization of the rescue spreader problem. References are included to direct the reader to more detailed expositions if desired.

A.1 Preliminaries

Two basic mathematical concepts underlie B-splines, namely parametric functions and linear interpolation. Parametric functions take one or more variables, known as the parameters (hence parametric), as inputs and return the same number of outputs, which is also equal to the dimension of the resulting manifold. For example, a one parameter function gives a curve, a two parameter function gives a surface, etc. As the Euclidean plane is of interest for this problem, parametric functions will in this thesis be taken to be restricted to a single parameter.

A parametric function s of a single parameter u can be written

$$s(u) = \{x(u), y(u)\} \tag{A.1}$$

The distinguishing feature of a parametric function is that each dimension of the result-

ing curve is given by an independent function, in this case $x(u)$ and $y(u)$. To take a familiar example, the unit circle can be generated parametrically by the function

$$s(u) = \{cos(u), sin(u)\} \tag{A.2}$$

as illustrated in Figure A.1

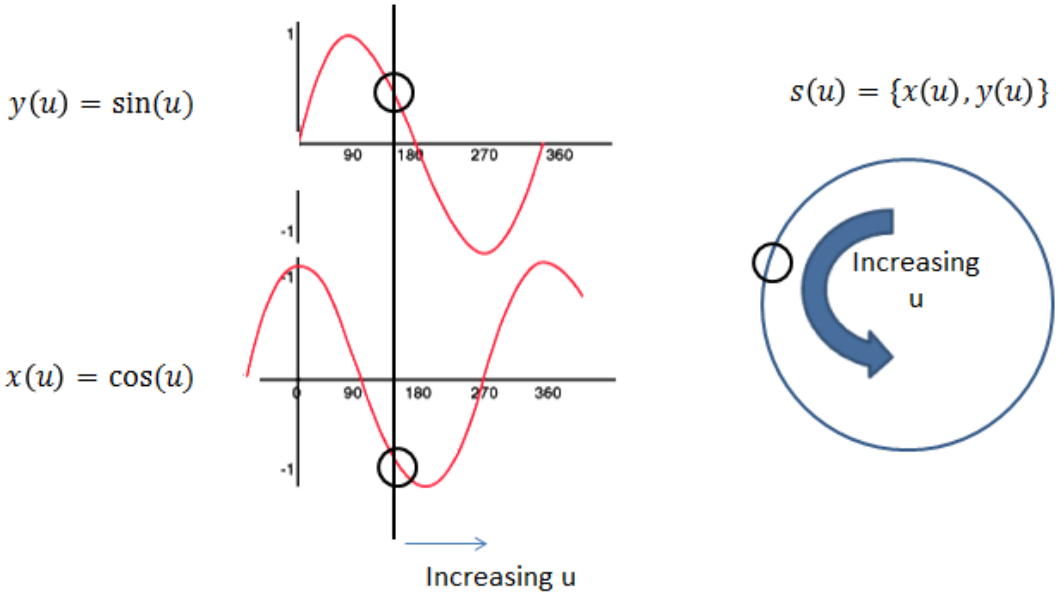


Figure A.1: Parametric definition of unit circle

The concept of a parametric function can be used to help explain the concept of linear interpolation. Let a and b be any two distinct vectors specifying points in space. The infinite line that passes through those two points can be thought of as a one-dimensional parametric curve s, parameterized by t as in equation A.3

$$s(t) = (1 - t)a + (t)b \tag{A.3}$$

where $-\infty < t < \infty$. If t is instead restricted to the closed interval $[0, 1]$, only the

line segment connecting a and b is included. It is said that $s(t)$ is a linear interpolation of a and b. If E^n is restricted to E^2, $s(t)$ is a curve in two-dimensional space, composed of an x function and a y function as in equation A.1. These two functions are also linear interpolations, but in a single dimension:

$$x(t) = (1-t)a_x + (t)b_x \tag{A.4}$$

and

$$y(t) = (1-t)a_y + (t)b_y \tag{A.5}$$

where the subscripts x and y denote those components of the vectors a and b. Linear interpolation in the plane on the closed interval $[0, 1]$ is illustrated in Figure A.2

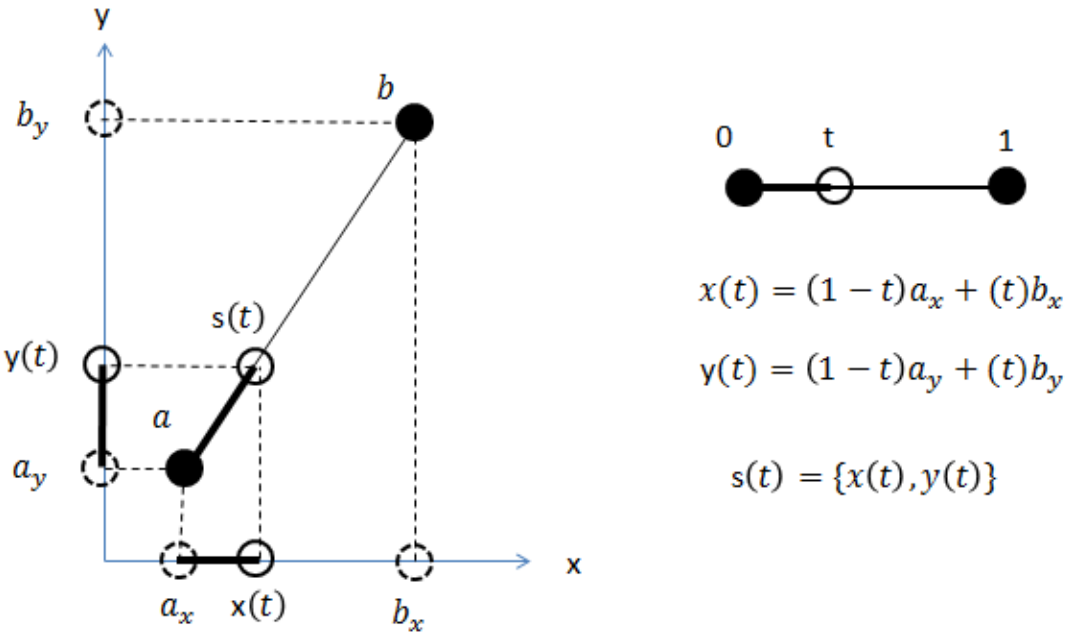


Figure A.2: Planar linear interpolation

A.2 Polynomial Curves by Recursive Linear Interpolation

With linear interpolation between two points established, it is now shown that polynomial curves can be generated using recursive linear interpolation. Although not explicity stated thus far, the set of two points $\{a, b\}$ that are interpolated between must in fact be an *ordered* set (a, b) so that the direction that the interpolation is constructed in is unambiguous (e.g. if $t = 1/4$, is the resulting point $1/4$ of the way from a to b or from b to a?). A third point c can be added to the set, resulting in (a, b, c). If this set is subdivided into all possible two-element sets of consecutive elements, the original set (a, b) is recovered, along with a new set (b, c). A linear interpolation can then be constructed on each of the two sets, just as in the previous case, resulting in

$$s_1(t_1) = (1 - t_1)a + (t_1)b \tag{A.6}$$

and

$$s_2(t_2) = (1 - t_2)b + (t_2)c \tag{A.7}$$

With the additional stipulation that the same parameter is used for both interpolations, that is, $t_1 = t_2 = t$, the result is as shown in Figure A.3

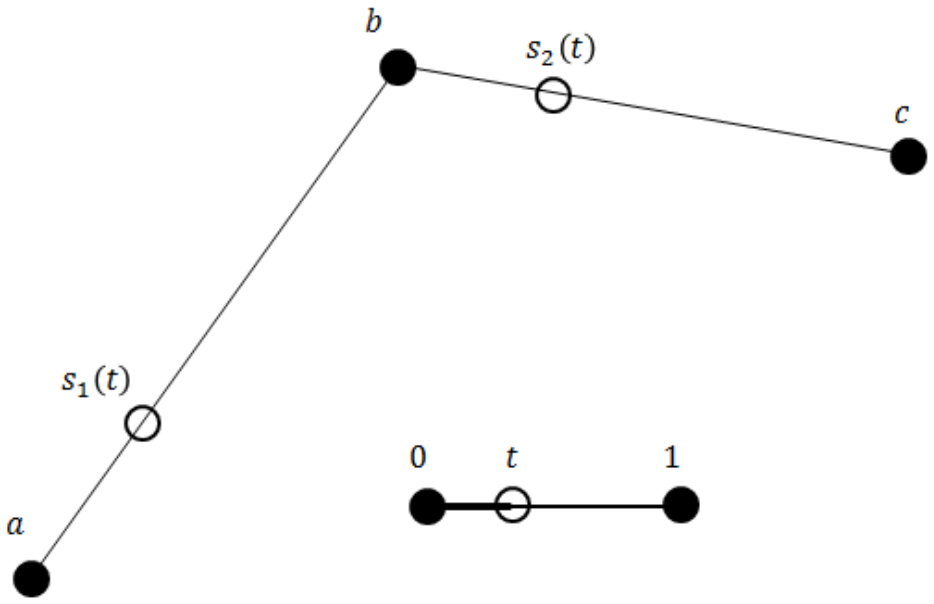


Figure A.3: Double linear interpolation

Now a recursive aspect is introduced. If s_1 and s_2 are themselves used as the endpoints of an interpolative interval, a third function $q(t)$ can be defined recursively based on the original interpolations as

$$q(t) = (1-t)s_1 + (t)s_2 \tag{A.8}$$

which combined with equations A.6 and A.7 can also be expressed explicitly as

$$q(t) = (1-t)\left[(1-t)a + (t)b\right] + (t)\left[(1-t)b + (t)c\right] \tag{A.9}$$

It will be noted that the result is now a quadratic in t. Simple linear interpolation generates a polynomial of degree 1 (a linear function), and a single level of recursion generates a polynomial of degree 2 (a quadratic function), as illustrated in Figure A.4. One can imagine s_1, s_2, and q as beads sliding along wires pegged to a table at a, b, and c, with the progress of each bead from the first to the second peg on the wire being controlled simultaneously

by moving the "control bead" t. As t is slid from 0 to 1, $q(t)$ will sweep out a quadratic curve.

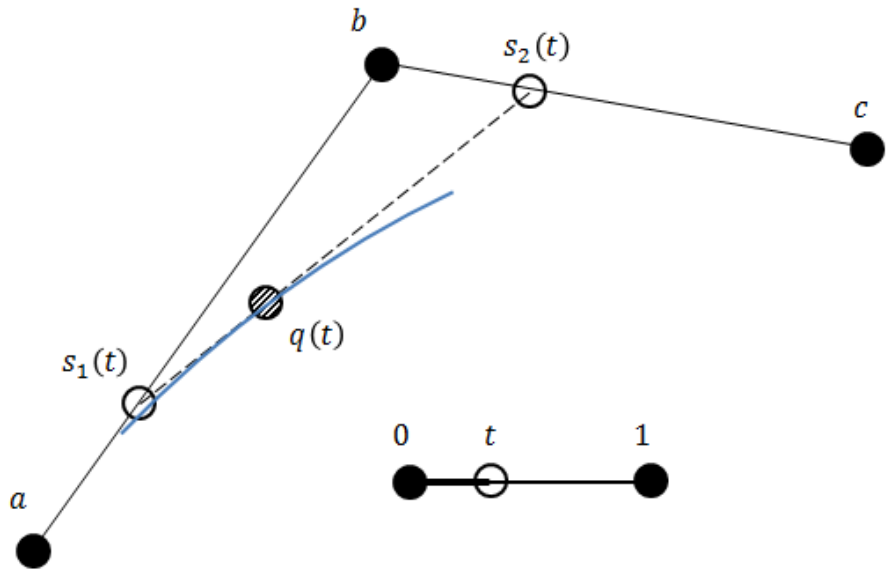


Figure A.4: Quadratic curve from recursive linear interpolation

As one might imagine, there is no reason that this type of recursion is limited to a single level. If a fourth point is added to the set, it becomes (a, b, c, d), and the set of all possible two-element sets of consecutive elements becomes $\{(a, b), (b, c), (c, d)\}$. Linear interpolations are then constructed on each of these three sets, with two quadratic recursive interpolations now being possible. The first is as in the previous example, from (a, b) and (b, c), and the second is naturally from (b, c) and (c, d). A cubic curve $c(t)$ can then be constructed by linear interpolation between the two quadratic functions, as shown in Figure A.5.

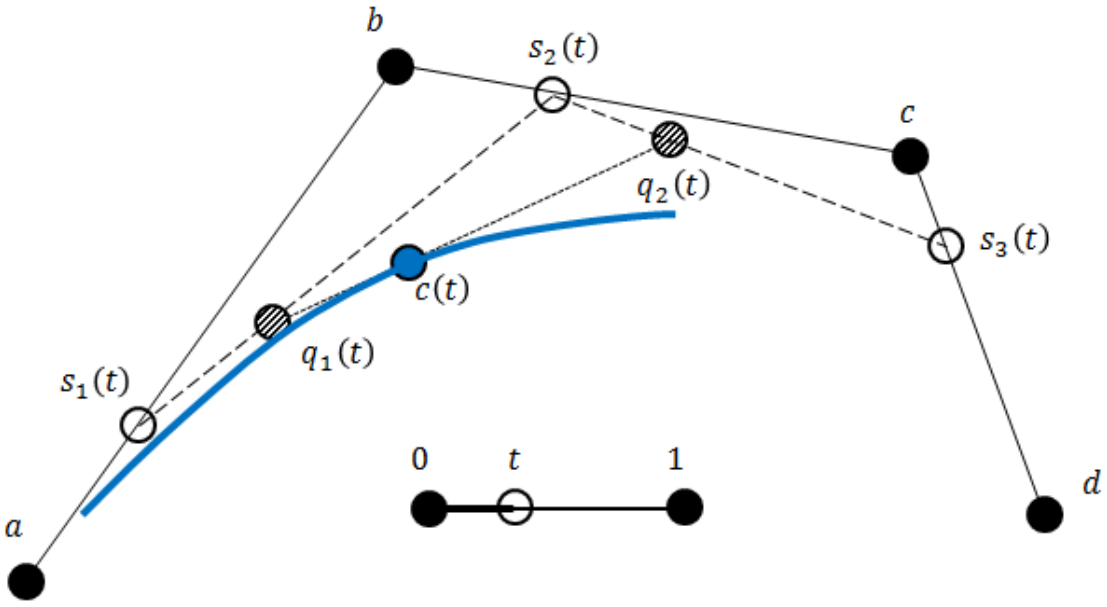


Figure A.5: Cubic curve from recursive linear interpolation

This recursive procedure can be continued indefinitely to produce a polynomial of any order, given enough points in the set. A polynomial constructed in this manner is known as a Bezier curve. B-splines are closely related to Bezier curves, and as B-splines of cubic order (by far the most common order) are used in this thesis, no further recursion need be applied beyond that shown in Figure A.5.

A.3 B-splines as Piecewise Bezier curves

So far the parameter t that is used generate the parametric interpolations has been restricted to the interval $[0, 1]$. As will be seen shortly it proves convenient to generalize this interval to any two nonnegative real numbers. Suppose it is desired to use the two numbers $\{u_0, u_1\}$ as the endpoints of the interval that the interpolating parameter, which will now be called u instead of t, can occupy. That is, $u_0 \leq u \leq u_1$. It becomes necessary to map the parameter u to an intermediate parameter α in order to maintain the 0 to 1 behavior that

allows the definition of linear interpolation given in equation A.3 to work. That is, define

$$\alpha = \frac{u - u_0}{u_1 - u_0} \tag{A.10}$$

This results in α assuming a value between 0 and 1, and reproducing the behavior of the parameter t in the simpler case, for any value of u between u_0 and u_1. Using this definition, exactly the same Bezier curve in the x-y Cartesian plane as is shown in Figure A.5 is produced in Figure A.6, with the only difference being that the parameter is $u_0 \leq u \leq u_1$ rather than $0 \leq t \leq 1$.

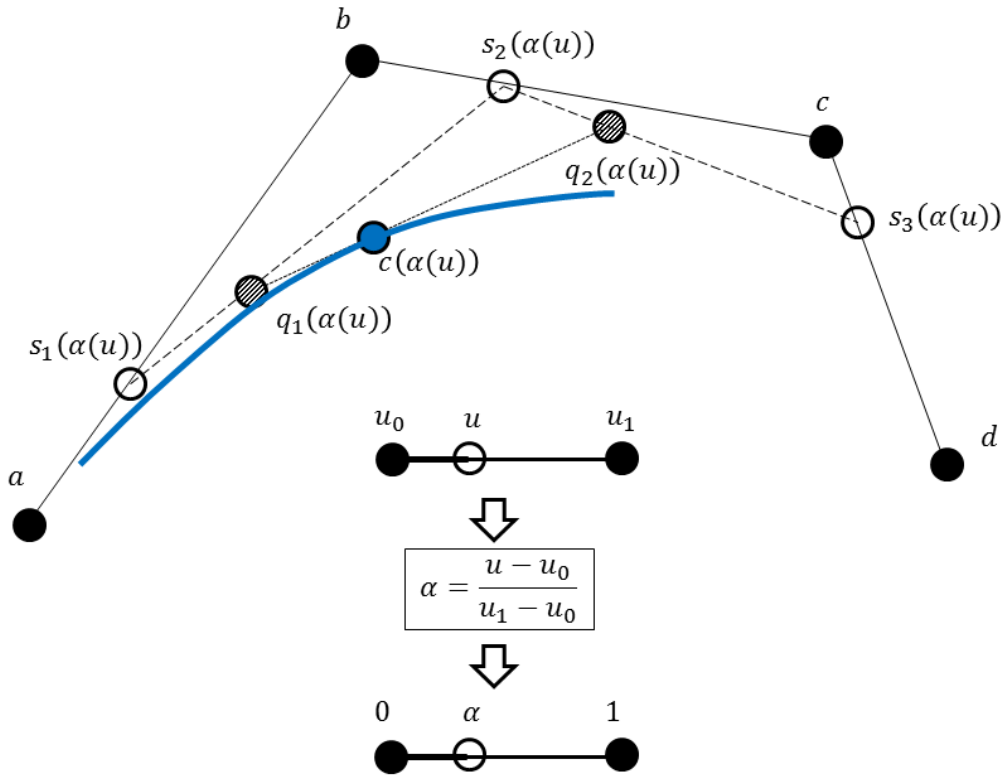


Figure A.6: Cubic Bezier curve on generalized parametric interval

Moving on from the details of the construction process in order to focus on a simple

input/ouput relationship, the Bezier polynomial curve is completely specified by the order of the recursion (cubic in this case) and the positions of the points (a, b, c, d), assuming that the parameter u "sweeps" through the full allowable range $u_0 \leq u \leq u_1$ to generate the entire curve. The points (a, b, c, d) are known as *control points*, which form a *control polygon*. The edges of the polygon connect consecutive points in the set, with the final edge connecting the first and last points in the set to form a closed shape, although this edge is not used for the interpolative construction of the curve. The control polygon and Bezier curve from the example used thus far are shown in Figure A.7

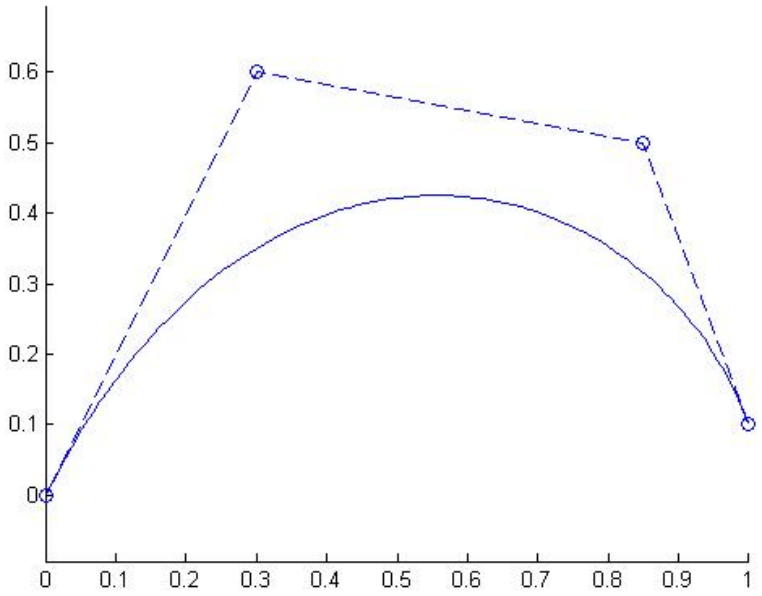


Figure A.7: Bezier curve and control polygon

Note that as the entire recursive interpolation collapses onto the first control point a when $u = u_0$ and the last control point d when $u = u_1$, the Bezier curve always starts and finishes coincident with the first and last control points.

With the change of parameter established and the behavior of Bezier curves understood, the construction of B-splines as piecewise Bezier curves can now be discussed. Let a

fifth point e be added to the set of control points, but without increasing the degree of the recursion to generate a quartic polynomial. Since only four points are needed for a cubic Bezier curve, the ordered set (a, b, c, d, e) is split into all possible subsets of four elements with the same order, i.e. (a, b, c, d) and (b, c, d, e). A corresponding change is made to the parameter range: the single interval $[u_0, u_1]$ is bifurcated into the double interval $[u_0, u_1] \cup [u_1, u_2]$. The curve generation begins with the parameter u at its starting value of u_0. As it sweeps through the first interval $[u_0, u_1]$, the first subset of control points (a, b, c, d) is active, and the first portion of the Bezier curve is generated exactly as in the previous case. This is illustrated in Figure A.8

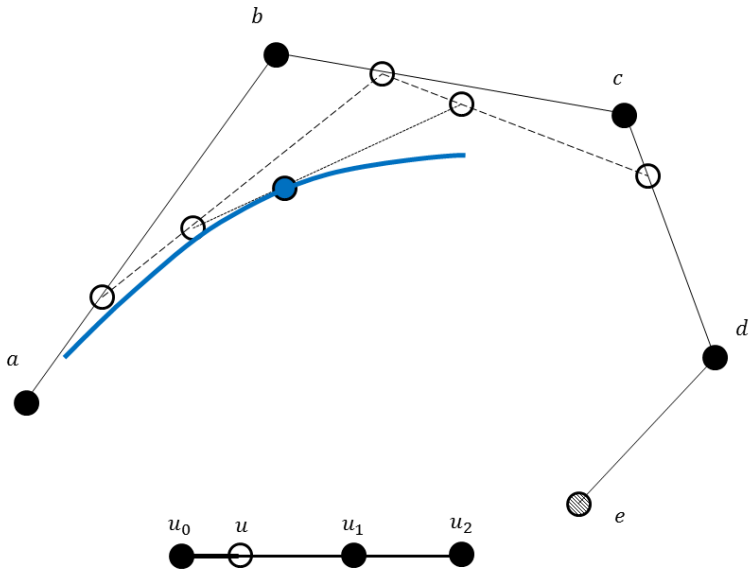


Figure A.8: First interval

As soon as the value of u passes u_1, however, it enters the second interval and the second subset of control points (b, c, d, e) activates. The remainder of the Bezier curve is then generated using this second subset as illustrated in Figure A.9

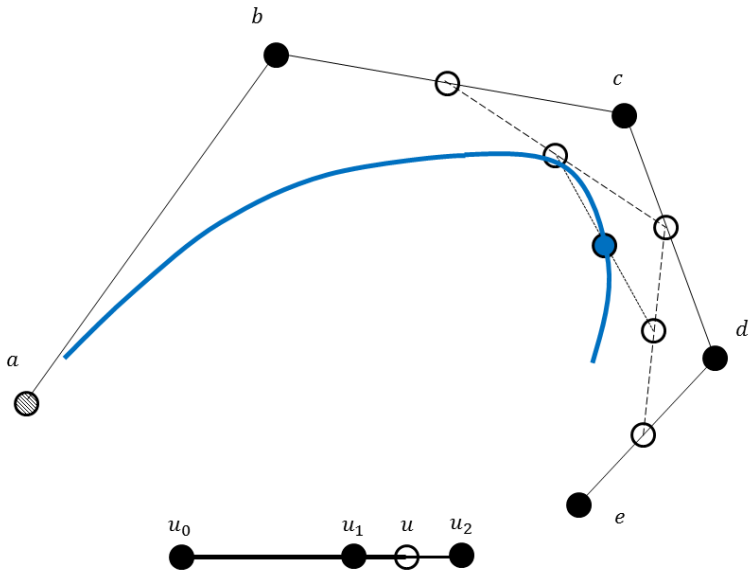


Figure A.9: Second interval

The entire curve thus generated is shown in Figure A.10 along with the control polygon. This curve is known as a B-spline.

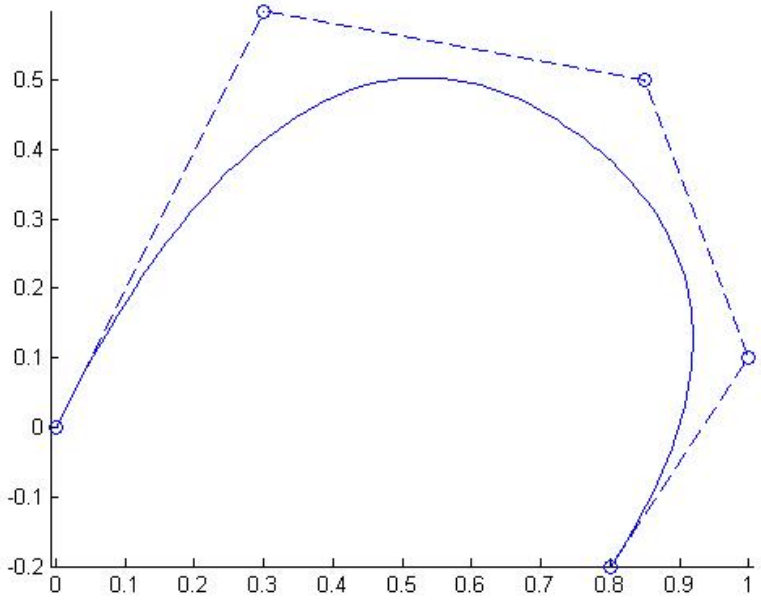


Figure A.10: B-spline and control polygon

A.4 Knots and Knot Sequences

The ordered set (u_0, u_1, u_2) is known as the *knot sequence*, and each element of the set is a *knot*. It is important to avoid confusion between the knots and control points. The knots have no direct meaning in the Cartesian plane in which the B-spline exists, and only exist in the separate one-dimensional space of the parameter u to specify the u values at which the next set of control points should activate during construction of the spline. The control points, on the other hand, exist in the Cartesian plane and give a specific shape to the 2D curve which is being parametrically swept out. Thus Figure A.10, which is a picture of the portion of the Cartesian plane which contains the B-spline, shows only the control points and not the knots. Indeed there is no meaningful way to depict the knots as existing in the plane.

During solution of the rescue spreader problem it becomes necessary to generate a B-

spline curve from a prescribed set of control point locations. This entails coming up with a knot sequence. The easiest method is to use a uniform B-spline, where the knots are all evenly spaced, that is, $(u_1 - u_0) = (u_2 - u_1) = (u_3 - u_2) = ...$ However, this can result in reduced spline quality at tightly clustered control points where the curve changes shape dramatically. Non-uniform B-splines, where the knots are not necessarily evenly spaced, will produce higher quality results, but are more mathematically complex. Another consequence of using non-uniform B-splines is the need to find a way of determining what the non-uniform knot interval lengths will be. A popular method is the chord length method, where the ratio of knot interval lengths is proportional to the ratio of Cartesian distances between control points,

$$\frac{u_{i+1} - u_i}{u_{i+2} - u_{i+1}} = \frac{\|d_{i+1} - d_i\|}{\|d_{i+2} - d_{i+1}\|} \tag{A.11}$$

where u_i are knots and d_i are vectors specifying control point locations [91]. This is the method used in this thesis. Note that as only the ratio of consecutive interval lengths matters, the absolute length of the intervals and thus of the entire knot sequence can be chosen arbitrarily. Here $u = 0$ is taken as the start of the curve and $u = 1$ as the end.

A.5 Parametric Nature of B-Splines

The B-spline example shown in Figure A.10 is the most basic that does not reduce to a simple Bezier curve, i.e. it is only a two-interval B-spline. Arbitrarily high numbers of control points and u-intervals may be used to generate much more complex B-splines. An example with seven control points and four knot intervals is shown in Figure A.11. Note that while one can count the control points in the graphical representation, the knot intervals only serve to subtly reshape the curve and have no directly observable manifestation.

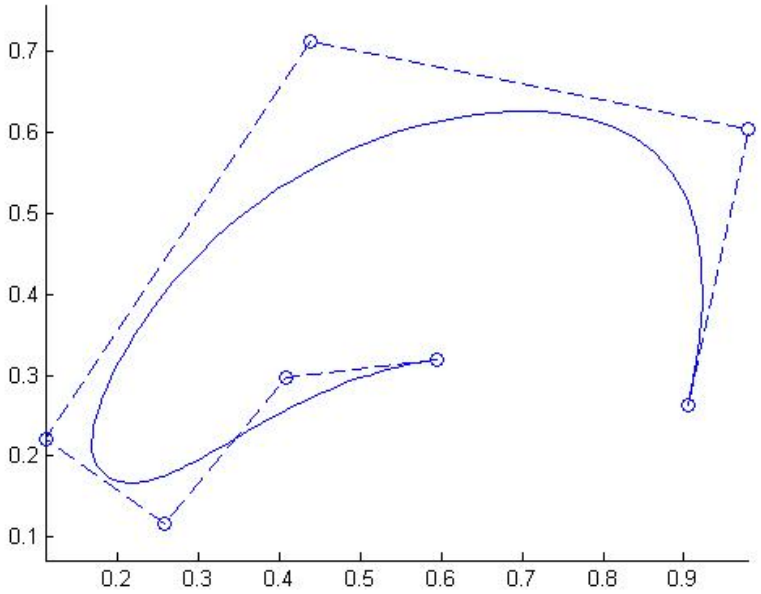


Figure A.11: More complex B-spline example

As of yet the parametric nature of the B-spline formulation has not been apparent, so a pause to clarify this point is in order. Just as in Figure A.2 a linear interpolation in the plane is produced parametrically from two one-dimensional linear interpolations, and similarly as in Figure A.1 the unit circle is produced parametrically from two one-dimensional sinusoidal functions, a B-spline in the plane is produced parametrically from an x function and a y function. The graphical development used here was chosen for intuitive clarity, but corresponds more closely to a vector-based approach that treats both x and y together than to the parametric approach actually used in the computer implementation. Without getting sidetracked into the detailed mathematics, suffice it to say that when one wishes to evaluate a point on a B-spline, one uses what is known as the recursive de Boor algorithm [91]. This can only handle one dimension at a time, so the actual procedure is to generate two polynomials $x(u)$ and $y(u)$, considering first the x-coordinates and then the y-coordinates of the control polygon, which when parametrically combined produce the planar curves

$s(u) = \{x(u), y(u)\}$ that are shown in the figures. For a mathematically rigorous yet admirably clear approach to the mathematics of B-splines, the reader is referred to [81], or for the more mathematically inclined reader [82] might be more suitable.

Appendix B:

Basics of FEA for Planar Elasticity

This appendix presents the mathematics behind the Constant Strain Triangle (CST) finite element used for the rescue spreader problem, as well as some additional material on how a collection of such elements can be assembled into a unified mathematical structure and used to solve a problem in planar elasticity.

B.1 Displacement, Strain, and Stress in the CST

A diagram of the CST is shown in Figure B.1. The CST is a simple element consisting of a triangle with straight sides and a node at each vertex. The local node numbering proceeds CCW so as to produce a positive signed area. The nodal coordinates are given in the global coordinate system.

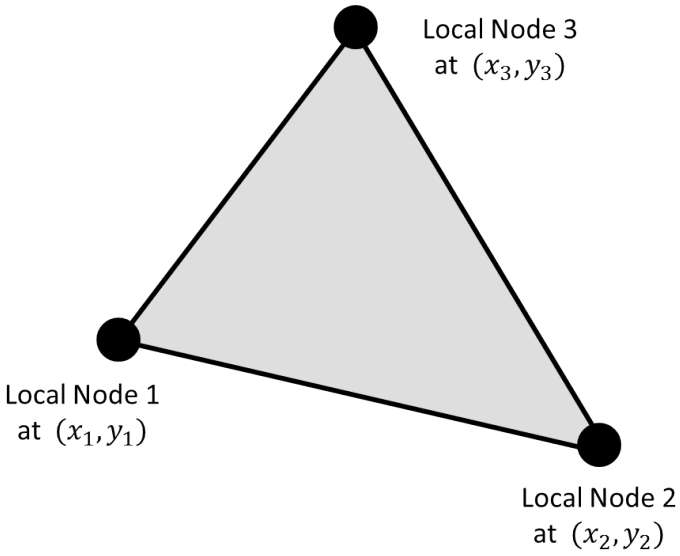


Figure B.1: Constant Strain Triangle finite element

The way that a finite element approximates a continuously variable quantity (such as the displacement field in a deforming solid) is by computing the value of the field at the nodal locations and extrapolating from that information to estimate the value over the rest of the element. The method of extrapolation depends on the element type. As the CST by definition takes the simple approach of assuming a constant strain in the interior, it must be the case that the displacement field varies in a linear fashion throughout the element in order to produce a constant function when differentiated. Given only the information that the displacement field is linear and two-dimensional over the triangular element, the displacement functions u and v (in the x and y directions respectively) are given by

$$\begin{aligned} u &= c_1 + c_2 x + c_3 y \\ v &= c_4 + c_5 x + c_6 y \end{aligned} \tag{B.1}$$

following the derivation given in [92], although virtually every basic FEA text will have an equivalent derivation. As will be seen, the only other information available comes in the

form of nodal displacements resulting from solution of the FEA system, which allows solution for the constants $c_1 - c_6$ so as to come up with explicit expressions for the displacement functions u and v. These known nodal displacements are displayed in Figure B.2

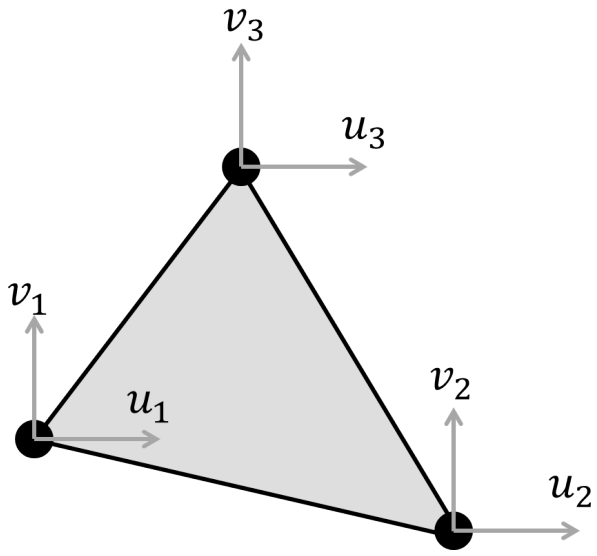


Figure B.2: Nodal displacements for CST

With this information, the next step is to solve for the six unknown constants using the six known nodal displacement components. The solution comes out to

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_2 & \beta_3 \\ \gamma_1 & \gamma_2 & \gamma_3 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{B.2}$$

and

$$\begin{bmatrix} c_4 \\ c_5 \\ c_6 \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_2 & \beta_3 \\ \gamma_1 & \gamma_2 & \gamma_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{B.3}$$

where

$$\begin{array}{lll} \alpha_1 = x_2y_3 - y_2x_3 & \alpha_2 = y_1x_3 - x_1y_3 & \alpha_3 = x_1y_2 - y_1x_2 \\ \beta_1 = y_2 - y_3 & \beta_2 = y_3 - y_1 & \beta_3 = y_1 - y_2 \\ \gamma_1 = x_3 - x_2 & \gamma_2 = x_1 - x_3 & \gamma_3 = x_2 - x_1 \end{array} \tag{B.4}$$

and

$$2A = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) \tag{B.5}$$

with A also corresponding to the area of the triangle. This result can be re-written at an increased level of abstraction as

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix} \tag{B.6}$$

where the N_i are known as the *shape functions*, a concept of great importance in finite element theory, and are the means by which nodal quantities are extrapolated into continuous intra-element quantities. From equations B.2 and B.3 it can be deduced that the definitions of the shape functions (for the CST) are:

$$\begin{aligned} N_1 &= \frac{1}{2A}(\alpha_1 + \beta_1 x + \gamma_1 y)u_1 \\ N_2 &= \frac{1}{2A}(\alpha_2 + \beta_2 x + \gamma_2 y)u_2 \\ N_3 &= \frac{1}{2A}(\alpha_3 + \beta_3 x + \gamma_3 y)u_3 \end{aligned} \tag{B.7}$$

Writing equation B.6 in even more compact form,

$$\underline{\boldsymbol{\psi}} = \underline{\boldsymbol{N}}\,\underline{\boldsymbol{d}} \tag{B.8}$$

where $\underline{\boldsymbol{\psi}}$ is the 2x1 matrix of the displacement field components u and v, $\underline{\boldsymbol{N}}$ is the 2x6 matrix of displacement functions, and $\underline{\boldsymbol{d}}$ is the 6x1 matrix of nodal displacement components.

The next step is to determine the strain field in the element from the displacement field. The three components of strain in the planar case are the x strain ϵ_x, the y strain ϵ_y, and the shear strain γ_{xy}. These can be found from the displacement field components using

$$\begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} du/dx \\ dv/dy \\ du/dy \;+\; dv/dx \end{bmatrix} \tag{B.9}$$

If one applies the definitions of u and v in terms of shape functions and cranks through the algebra, the result is

$$\begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} \beta_1 & 0 & \beta_2 & 0 & \beta_3 & 0 \\ 0 & \gamma_1 & 0 & \gamma_2 & 0 & \gamma_3 \\ \gamma_1 & \beta_1 & \gamma_2 & \beta_2 & \gamma_3 & \beta_3 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix} \tag{B.10}$$

or in compact form,

$$\underline{\boldsymbol{\epsilon}} = \underline{\boldsymbol{B}}\,\underline{\boldsymbol{d}} \tag{B.11}$$

where $\underline{\boldsymbol{\epsilon}}$ is the 3x1 matrix of strain components and $\underline{\boldsymbol{B}}$ is the 3x6 matrix of terms depending on the nodal coordinates. So it is seen, unsurprisingly, that the strain is determined

by the nodal coordinates and the nodal displacements. Additionally, as all the entries in $\underline{\boldsymbol{B}}$ and $\underline{\boldsymbol{d}}$ are constant, the constant strain condition is indeed fulfilled.

It is a short step to go from strains to stresses for this linear elastic case. The matrix of stress components $\underline{\boldsymbol{\sigma}}$ for the planar case can be found from the strains using

$$\underline{\boldsymbol{\sigma}} = \underline{\boldsymbol{D}}\,\underline{\boldsymbol{\epsilon}} \tag{B.12}$$

where the matrix $\underline{\boldsymbol{D}}$ reflects the constitutive law describing the relevant physics. For a one-dimensional situation, the matrix reduces to a 1x1 or scalar value, which is none other than the spring constant of Hooke's Law. For a general 3D problem one must apply the full 6x6 $\underline{\boldsymbol{D}}$ matrix that describes the general linear elastic constitutive law. For a 2D problem, one of two simplifying assumptions is applied to obtain a 3x3 $\underline{\boldsymbol{D}}$ matrix. Plane strain is applied when the part is of a very large uniform thickness, and specifies that the strain in the out-of-plane direction is negligible, since the thickness is so large that any displacement does not create significant strain. Conversely, the plane stress approximation is used for parts of a very small uniform thickness, where since the out-of-plane stress must go to zero on the front and back faces it is hypothesized that there is not enough room in the middle for it to grow to a significant values. Each of these assumptions produces a different $\underline{\boldsymbol{D}}$ matrix. For this problem, plane stress is adopted, which corresponds to [92]

$$\underline{\boldsymbol{D}} = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{B.13}$$

so that the expanded form of equation B.12 is

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}\begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} \tag{B.14}$$

where E is Young's Modulus and ν is Poisson's Ratio. Since the strain is constant over the entire element, it is of course the case that the stress is as well.

B.2 Global Stiffness for an FEA Mesh

By the manipulations in the preceding section, it is shown how the stress in a CST element can be determined from the nodal displacements. However, the method by which the nodal displacements are calculated in the first place has not yet been discussed. This is done by application of what is perhaps the most fundamental equation in finite element theory,

$$\underline{\boldsymbol{F}} = \underline{\boldsymbol{K}}\,\underline{\boldsymbol{D}} \tag{B.15}$$

For a finite element mesh with n nodes possessing δ degrees of freedom each, there are $n\delta$ degrees of freedom needed to completely describe the state of the system. $\underline{\boldsymbol{F}}$ is a $1 \times n\delta$ vector of nodal force components, $\underline{\boldsymbol{D}}$ is a $1 \times n\delta$ vector of nodal displacement components, and $\underline{\boldsymbol{K}}$ is a $n\delta \times n\delta$ square matrix of stiffnesses. For this 2D problem, $\delta = 2$ as each node has an x and y degree of freedom. Thus the expanded form of equation B.15 for this case is

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ \dots \\ f_{2n} \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & \dots & k_{1\ 2n} \\ & k_{22} & k_{23} & k_{24} & \dots & k_{2\ 2n} \\ & & k_{33} & k_{34} & \dots & k_{3\ 2n} \\ & & & k_{44} & \dots & k_{4\ 2n} \\ & & & & & \dots \\ sym & & & & & k_{2n\ 2n} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ \dots \\ d_{2n} \end{bmatrix} \tag{B.16}$$

To understand the significance of equation B.16, it must first be realized that each node in the mesh has a global node number from 1 to $2n$ that serves as a unique identifier or

"name" for that node. These global node numbers also identify which degrees of freedom in equation B.16 correspond to which node. Thus to look at global node #1, for example, f_1 is the force acting on node 1 in the x direction and f_2 is the nodal force in the y direction. Similarly, d_1 and d_2 are the x and y nodal displacements of global node 1. Thus the entries in $\underline{\boldsymbol{K}}$ determine the degree to which a given member of the force vector affects a given member of the displacement vector, that is, k_{ij} is the *stiffness* of displacement j with respect to force i.

For most problems, including the present case of structural elasticity, it only makes sense for neighboring nodes (that is, nodes that share at least one element) to have nonzero influence on each other. Applying a force to a node will tend to deform elements that it is a member of, thus inducing displacements in all the other nodes in those elements, but this force will have no direct impact on a node all the way on the other side of the mesh, or even on one two elements away. Thus for all but the tiniest meshes the global stiffness matrix $\underline{\boldsymbol{K}}$ will consist almost entirely of 0 entries.

It can now be seen how solution of equation B.16 for $\underline{\boldsymbol{D}}$ will determine the nodal displacements necessary to calculate the stresses in each element of the mesh. Before this can happen, there are two additional steps needed. The boundary conditions of the problem are applied by filling in known values in the $\underline{\boldsymbol{F}}$ and/or $\underline{\boldsymbol{D}}$ vectors. For example, if a load is applied along one side of the mesh, known f values are placed in the positions corresponding to the appropriate degrees of freedom in $\underline{\boldsymbol{F}}$. Similarly, support conditions can be implemented by requiring that certain nodes have zero displacement in one or both degrees of freedom. More complex boundary conditions are also possible, for example adding an explicit constraint equation on f and d for a particular degree of freedom, corresponding to an elastic support. It is important to note that only one constraint can be made on any given degree of freedom, that is, only f *or* d (or a single constraint equation) can be prescribed without overconstraining the solution. The "default" boundary condition for a degree of freedom is $f = 0$, which is applied to a typical node in the interior, saying that the node

should displace as needed when the system is solved in order to go to a new equilibrium.

Besides application of boundary conditions, the other necessary step is to fill in the entries of the global stifffness matrix $\underline{\boldsymbol{K}}$. This is done by calculating the elemental stiffness matrices for each element in the mesh and assembling them into the global matrix. This approach is possible since only nodes that share an element affect each other, so simply combining all nodal interdependences from each element will produce the full behavior of the entire system. Care must be taken to keep track of the correct correspondence between local node numbers in each element and the unique global node numbering, as each node has a dual identity as a member of the mesh and a member of particular elements. This is illustrated in Figure B.3, on an example mesh of 50 nodes and 43 elements. In the example, the fact that the three local nodes of Element #31 also correspond to global nodes 26,33, and 27 respectively is highlighted.

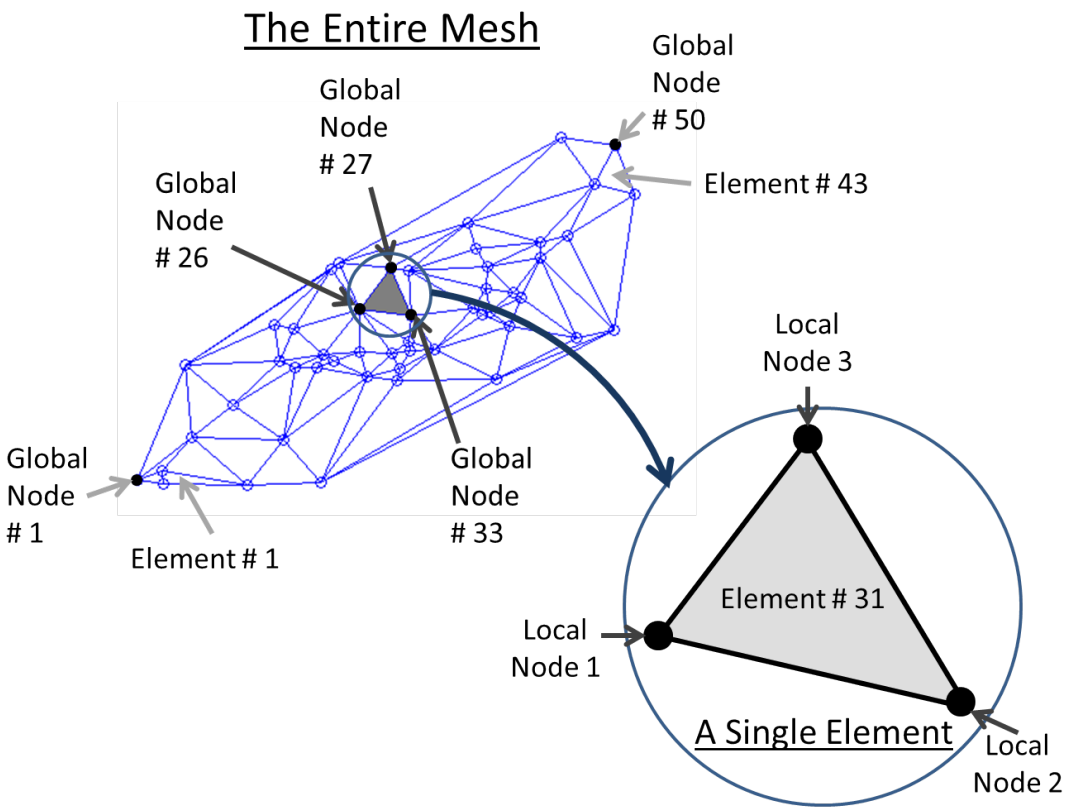


Figure B.3: Local and global node numbering

By making use of this correspondence, stiffnesses calculated for each element can be inserted into the correct degree of freedom location in $\underline{\boldsymbol{K}}$ during assembly.

B.3 Elemental Stiffness for the CST

The elemental stiffness calculation depends on the element type and the relative locations of the local nodes. Here the elemental stiffness matrix for the CST element is found. In general, the elemental stiffness $\underline{\boldsymbol{k}}$ of any finite element can be found by integrating over the elemental volume [92]

$$\underline{\boldsymbol{k}} = \iiint_V \underline{\boldsymbol{B}}^T \underline{\boldsymbol{D}}\, \underline{\boldsymbol{B}}\, dV \tag{B.17}$$

where $\underline{\boldsymbol{B}}$ reflects the nodal coordinates and is as defined in equations B.10 and B.11, and $\underline{\boldsymbol{D}}$ reflects material properties and physics assumptions and is as defined in equation B.13. Unlike higher-order elements, as a constant strain element the CST is simple enough that this integration is performed analytically (higher-order elements use numerical integration at each evaluation, adding some computational expense). Since both $\underline{\boldsymbol{B}}$ and $\underline{\boldsymbol{D}}$ do not vary over the element volume, equation B.17 evaluates simply to

$$\underline{\boldsymbol{k}} = t\ A\ \underline{\boldsymbol{B}}^T \underline{\boldsymbol{D}}\ \underline{\boldsymbol{B}} \tag{B.18}$$

where t is the (uniform) element thickness and A is the area. To put $\underline{\boldsymbol{k}}$ for the CST in concrete form, if the definitions of $\underline{\boldsymbol{B}}$ and $\underline{\boldsymbol{D}}$ are plugged in one can obtain the 6x6 elemental stiffness matrix in terms of 2x2 components:

$$\underline{\boldsymbol{k}} = \begin{bmatrix} \begin{bmatrix} \boldsymbol{k_{11}} \end{bmatrix} & \begin{bmatrix} \boldsymbol{k_{12}} \end{bmatrix} & \begin{bmatrix} \boldsymbol{k_{13}} \end{bmatrix} \\ & \begin{bmatrix} \boldsymbol{k_{22}} \end{bmatrix} & \begin{bmatrix} \boldsymbol{k_{23}} \end{bmatrix} \\ sym & & \begin{bmatrix} \boldsymbol{k_{33}} \end{bmatrix} \end{bmatrix} \tag{B.19}$$

where

$$\begin{bmatrix} \boldsymbol{k_{ij}} \end{bmatrix} = t\ A\ \boldsymbol{B_i}^T \boldsymbol{D}\ \boldsymbol{B_j} \tag{B.20}$$

if $\boldsymbol{B_i}$ is the ith 3x2 portion of the 3x6 $\underline{\boldsymbol{B}}$ matrix in equation B.10, that is,

$$\boldsymbol{B_i} = \begin{bmatrix} \beta_i & 0 \\ 0 & \gamma_i \\ \gamma_i & \beta_i \end{bmatrix} \tag{B.21}$$

The elemental stiffness matrix $\underline{\boldsymbol{k}}$ is now known directly in terms of nodal locations and material properties, and the entries of $\underline{\boldsymbol{k}}$ can be assembled into the correct locations in the global stiffness matrix $\underline{\boldsymbol{K}}$ by looking up the global node numbers of each of the local nodes in the element.

Appendix C:

Dynamic Analysis of Stephenson-III Linkage

The dynamic forces in the Stephenson-III linkage are analyzed in order to validate the assumption that they could be neglected relative to the structural loading. For this analysis, it is necessary to know the mass and linear acceleration of the COM of each link for the linear portion of the analysis, as well as the mass moment of inertia about the COM and angular acceleration for the rotational portion. The mass properties are known from the dual-point-mass and polygon model, and the angular accelerations were determined during the initial kinematic analysis. All that remains before beginning the dynamic analysis proper is determination of the linear accelerations of the link COMs from the linkage geometry and angular accelerations.

C.1 Linear Accelerations of COMs

The notation used is shown in Figure C.1. In general, for link j, θ_j is the angle of the link with respect to the +x axis, $\vec{R}_j$ is the link vector, $\vec{C}_j$ is the vector from the base of the link vector to the position of the link's COM, and ϕ_j is the angle from $\vec{R}_j$ to $\vec{C}_j$. The masses,

COM locations, and radii of gyration used in these calculations are the ones determined using point-mass counterweighting and polygonal modeling.

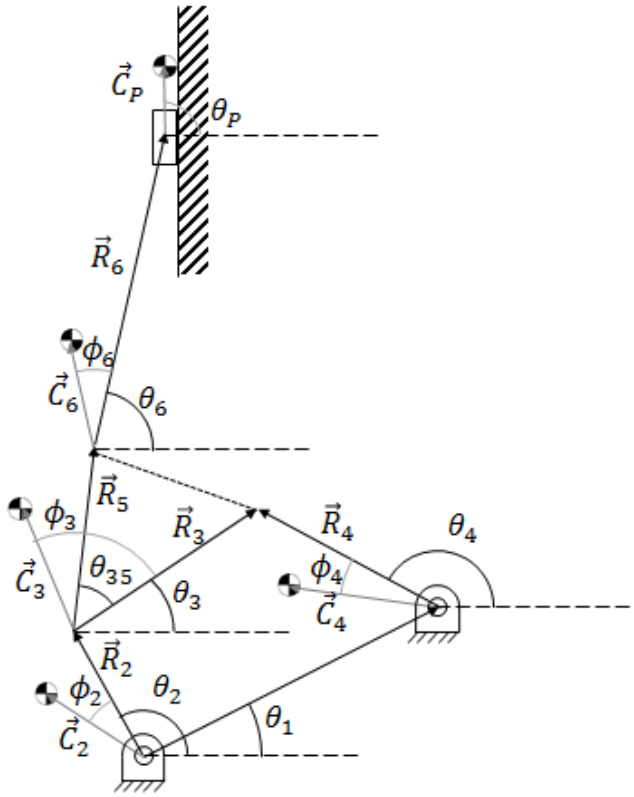


Figure C.1: Linkage geometry

The linear acceleration of each link's COM is found by writing the position vector S of the COM and taking two time derivatives.

Link 2 - Crank

The position vector of the crank's COM with respect to the origin is

$$S_2 = c_2 e^{i(\theta_2+\phi_2)} \tag{C.1}$$

Taking one time derivative, the linear velocity is then

$$V_2 = i\dot{\theta}_2 c_2 e^{i(\theta_2+\phi_2)} \tag{C.2}$$

and taking a second time derivative, the linear acceleration is

$$A_2 = i\ddot{\theta}_2 c_2 e^{i(\theta_2+\phi_2)} - \dot{\theta}_2^{\,2} c_2 e^{i(\theta_2+\phi_2)} \tag{C.3}$$

Link 3 - Coupler

The position vector of the coupler's COM with respect to the origin is

$$S_3 = r_2 e^{i\theta_2} + c_3 e^{i(\theta_3+\phi_3)} \tag{C.4}$$

Taking one time derivative, the linear velocity is then

$$V_3 = i\dot{\theta}_2 r_2 e^{i\theta_2} + i\dot{\theta}_3 c_3 e^{i(\theta_3+\phi_3)} \tag{C.5}$$

and taking a second time derivative, the linear acceleration is

$$A_3 = i\ddot{\theta}_2 r_2 e^{i\theta_2} - \dot{\theta}_2^{\,2} r_2 e^{i\theta_2} + i\ddot{\theta}_3 c_3 e^{i(\theta_3+\phi_3)} - \dot{\theta}_3^{\,2} c_3 e^{i(\theta_3+\phi_3)} \tag{C.6}$$

Link 4 - Rocker

The position vector of the rocker's COM with respect to the origin is

$$S_4 = r_1 e^{i\theta_1} c_4 e^{i(\theta_4+\phi_4)} \tag{C.7}$$

Taking one time derivative, the linear velocity is then

$$V_4 = i\dot{\theta}_4 c_4 e^{i(\theta_4+\phi_4)} \tag{C.8}$$

and taking a second time derivative, the linear acceleration is

$$A_4 = i\ddot{\theta}_4 c_4 e^{i(\theta_4+\phi_4)} - \dot{\theta}_4^{\,2} c_4 e^{i(\theta_4+\phi_4)} \tag{C.9}$$

Link 6 - Connecting Rod

The position vector of the connecting rod's COM with respect to the origin is

$$S_6 = r_2 e^{i\theta_2} + r_3 e^{i(\theta_3+\theta_{35})} + c_6 e^{i(\theta_6+\phi_6)} \tag{C.10}$$

Taking one time derivative, the linear velocity is then

$$V_6 = i\dot{\theta}_2 r_2 e^{i\theta_2} + i\dot{\theta}_3 r_3 e^{i(\theta_3+\theta_{35})} + i\dot{\theta}_6 c_6 e^{i(\theta_6+\phi_6)} \tag{C.11}$$

and taking a second time derivative, the linear acceleration is

$$A_6 = i\ddot{\theta}_2 r_2 e^{i\theta_2} - \dot{\theta}_2^{\,2} r_2 e^{i\theta_2} + i\ddot{\theta}_3 r_3 e^{i(\theta_3+\theta_{35})} - \dot{\theta}_3^{\,2} r_3 e^{i(\theta_3+\theta_{35})} + i\ddot{\theta}_6 c_6 e^{i(\theta_6+\phi_6)} - \dot{\theta}_6^{\,2} c_6 e^{i(\theta_6+\phi_6)} \tag{C.12}$$

Piston

The position vector of the piston's COM with respect to the origin is

$$S_P = r_2 e^{i\theta_2} + r_3 e^{i(\theta_3+\theta_{35})} + r_6 e^{i\theta_6} + c_P e^{i\theta_P} \tag{C.13}$$

Note that θ_P does not vary with time as the piston does not rotate. Taking one time derivative, the linear velocity is then

$$V_P = i\dot{\theta}_2 r_2 e^{i\theta_2} + i\dot{\theta}_3 r_3 e^{i(\theta_3+\theta_{35})} + i\dot{\theta}_6 r_6 e^{i\theta_6} \tag{C.14}$$

and taking a second time derivative, the linear acceleration is

$$A_P = i\ddot{\theta}_2 r_2 e^{i\theta_2} - \dot{\theta}_2^{\,2} r_2 e^{i\theta_2} + i\ddot{\theta}_3 r_3 e^{i(\theta_3+\theta_{35})} - \dot{\theta}_3^{\,2} r_3 e^{i(\theta_3+\theta_{35})} + i\ddot{\theta}_6 r_6 e^{i\theta_6} - \dot{\theta}_6^{\,2} r_6 e^{i\theta_6} \quad \text{(C.15)}$$

The calculation of linear accelerations is now complete. Next, a new system of coordinates is defined that is more amenable to calculation of dynamic forces.

C.2 Coordinate Transformation

The linkage with triangular binary links and a quadrilateral ternary (coupler) link is shown in Figure C.2. The problem setup is similar to that used for the applied force analysis, but it is important to understand that while some of the nomenclature is the same, the forces involved here are due only to dynamic effects and the applied force on the piston is not considered. The forces that the links exert on each other are labeled F_{ab} where link a is exerting a force on link b. The subscripts used are:

g: Ground (link 1)

2: Crank

c: Coupler

4: Rocker

6: Connecting Rod

P: Piston

A local non-rotating coordinate system is defined centered on each link's COM. A vector is defined from the origin of this system to each joint of the link, and is labeled $\vec{R}_{ab}$, where the vector is part of link b's coordinate system and points to the joint with link a. Note that the subscripting of the forces and vectors agrees so that $\vec{R}_{ab}$ is the moment arm that F_{ab}

acts through on link b's COM. This local coordinate system is illustrated using link 2 as an example in Figure C.2.

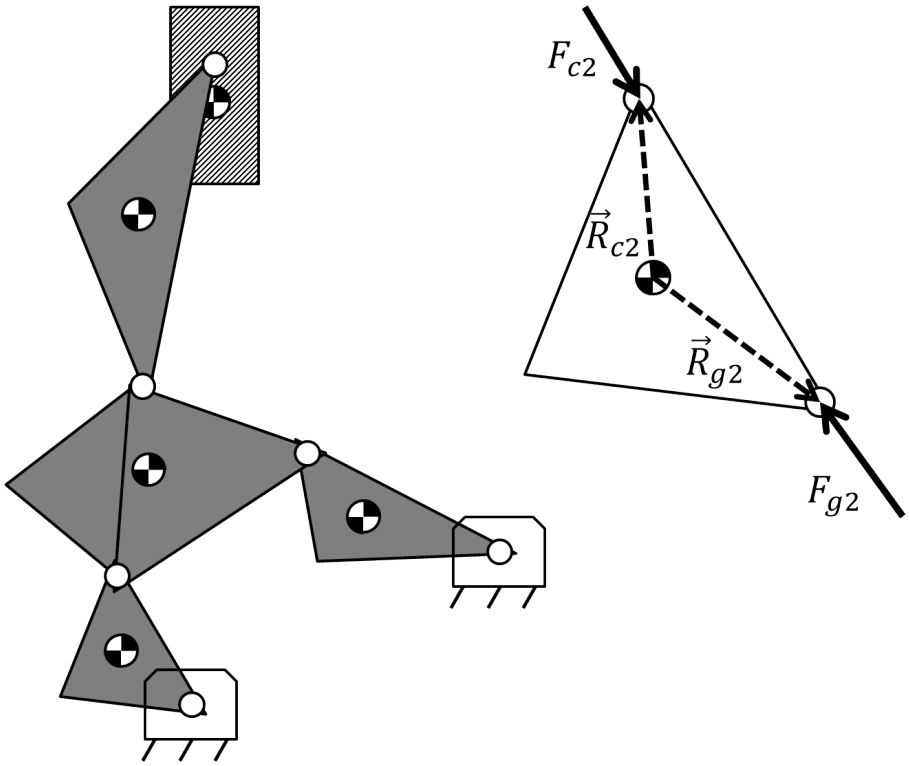


Figure C.2: Linkage diagram and local coordinate system definition

It will be noted that this coordinate system differs from that used for the motion analysis thus far (Figure C.1). Therefore the following transformations are needed, where the vectors in the system just described are on the left side and are put in terms of the system of Figure C.1.

Crank

$$\vec{R}_{g2} = -\vec{C}_2$$

$$\vec{R}_{c2} = \vec{R}_2 - \vec{C}_2$$

Coupler

$$\vec{R}_{2c} = -\vec{C}_3$$

$$\vec{R}_{4c} = \vec{R}_3 - \vec{C}_3$$

$$\vec{R}_{6c} = \vec{R}_5 - \vec{C}_3$$

Rocker (C.16)

$$\vec{R}_{g4} = -\vec{C}_4$$

$$\vec{R}_{c4} = \vec{R}_4 - \vec{C}_4$$

Connecting Rod

$$\vec{R}_{c6} = -\vec{C}_6$$

$$\vec{R}_{P6} = \vec{R}_6 - \vec{C}_6$$

Piston

$$\vec{R}_{6P} = -\vec{C}_P$$

Having defined this new coordinate system, the dynamic forces in the linkage can now be solved for.

C.3 Dynamic Force Solution

The solution for the dynamic forces amounts to simultaneous solution of a set of equations, with each link contributing three equations to the system. For link j, these equations are

$$\sum_i F_{ij,x} = m_j A_{j,x}$$
$$\sum_i F_{ij,y} = m_j A_{j,y} \tag{C.17}$$
$$\sum_i T_{ij} = I_j A_j$$

stating that the sum of the forces exerted by the other links i on link j must be equal to the mass of the link multiplied by its linear acceleration, and that similarly the sum of the torques exerted by the other links must be equal to the mass moment of inertia of the link multiplied by its angular acceleration. The right sides of all these equations are known by this point.

There linkage is assembled by means of six revolute joints, which can transmit forces but not torques. Therefore there are six unknown forces or twelve unknown force components. There is also an unknown output torque at the crankshaft (R_2 ground pivot). There are twelve known mass acceleration products, three each from the four links, and the thirteenth known comes from the single y-direction product for the piston on its vertical axis of slide. The problem can be written as a standard linear system solution $A\vec{x} = \vec{B}$. It is important to note that $\vec{B}$ must be divided by the gravitational constant (386 in/sec^2 in the IPS system used here) before solution of the system, as it is has units of lbm, which is a unit of weight and not mass. $\vec{x}$ is the vector of unknown forces:

$$\begin{bmatrix} F_{g2x} & F_{g2y} & F_{c2x} & F_{c2y} & F_{4cx} & F_{4cy} & F_{g4x} & F_{g4y} & F_{c6x} & F_{c6y} & F_{P6x} & F_{P6y} & T_{12} \end{bmatrix}^T$$

$\vec{B}$ is the vector of known mass acceleration products:

$$\begin{bmatrix} m_2a_{2x} & m_2a_{2y} & I_2\alpha_2 & m_3a_{3x} & m_3a_{3y} & I_3\alpha_3 & m_4a_{4x} & m_4a_{4y} & I_4\alpha_4 & m_6a_{6x} & m_6a_{6y} & I_6\alpha_6 & m_Pa_{Py} \end{bmatrix}^T$$

and A is the coefficient matrix:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -R_{g2y} & R_{g2x} & -R_{c2y} & R_{c2x} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & R_{2cy} & -R_{2cx} & -R_{4cy} & R_{4cx} & 0 & 0 & R_{6cy} & -R_{6cx} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & R_{c4y} & -R_{c4x} & -R_{g4y} & R_{g4x} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -R_{c6y} & R_{c6x} & -R_{P6y} & R_{P6x} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}$$