Formal vs. Discrete Optimization and HyperSizer

The Windows 95/NT based software package, HyperSizer™ is a new and powerful tool developed to analyze and optimize total structural systems. But aren't there other tools that already do this? Well, there are other general optimization tools, but it is our belief that no other tool addresses all of the large and small scale aspects of a structural system or automates structural optimization as completely as HyperSizer.

This white paper by Collier Research Corporation is not a review of optimization techniques, but rather a brief contrast between numerical, formal optimization techniques and the discrete optimization methods implemented in HyperSizer. After describing the strengths and shortcomings of formal methods as related to structural optimization, we describe discrete optimization and what makes HyperSizer ideal for optimizing structural systems.

Why Optimize?

In today's engineering environment, structures must first and foremost be designed to survive their environments. For many applications, however, such as aerospace vehicles or shipping containers, minimizing structural weight must also be of primary concern. The process of minimizing the weight of a structural design while ensuring structural integrity is called 'structural optimization.'

Optimization techniques are used in many fields of study. They generally involve the maximization or minimization of some pre-prescribed function (such as structural weight) with a set of mathematically specified constraints. Consider closing a rectangular plot of land with a given length of fence. Two variables x and y represent the length and width of the plot. A typical optimization problem involving this plot might be to determine the length and width so as to maximize the area. The area of the plot, called the *objective function*, is given by $A=xy$.

The optimization problem can be stated as:

Maximize the area of the plot of land, $A = xy$

The *constraint* on the problem is:

The total length of fencing, P must not exceed a certain length, P_{max}, or $P = 2x+2y \leq P_{max}$.

This is a rather simple optimization problem where the answer is obviously a square plot where $x=y=P_{max}/4$. Now assume that for some reason, the length of the plot cannot exceed some maximum value, x_{max}. This is an additional *constraint*, which can be stated mathematically as $x \leq x_{max}$. Now, the problem is slightly more complicated. The answer is still $x=y=P_{max}/4$ as long as x_{max} is greater than or equal to $P_{max}/4$, otherwise $x= x_{max}$ and $y= P_{max}/2- x_{max}$. Now consider an additional requirement that the length to width ratio cannot be less than 2. Now the problem becomes even more complicated. While this problem is simple and easy to solve, it would eventually become nearly impossible to solve by hand if constraints or requirements continued to be added. At this point we would turn to computational optimization.

Formal Optimization

The most widely used computational optimization schemes use numerical, so-called formal optimization. The objective function (or functions) and constraints in a formal optimization problem are described mathematically and initial guesses for the optimization variables are made. The objective function is evaluated and then gradients of this function with respect to the optimization variables are calculated and used to guide the variables to a "better" solution. Each time this procedure is performed, better and better solutions are obtained until the optimum solution is found. Because the above fence problem is well defined mathematically, it lends itself well to formal optimization methods.

Structural Optimization

Efforts in structural optimization have traditionally focused on mathematically based, formal optimization methods. Consider the following problem, which is given as an example in the MSC/NASTRAN Design Sensitivity and Optimization User's Guide. A cantilevered beam with a vertical end load is to be designed using a rectangular cross section. The optimization problem is to minimize the weight of the beam while limiting vertical deflection and insuring structural integrity. In this case, the objective function is the weight. The obvious constraints on the problem are: 1)The stress, σ cannot exceed the material yield stress; and 2) The deflection, δ cannot exceed a certain prescribed amount. What is not so obvious about this problem is that this beam has a potential torsional buckling failure mode. In the design of this beam, an engineering "rule-of-thumb" is typically used to guard against this type of failure. For this example, the rule-of-thumb is that the height-to-width ratio should not exceed 12 to prevent torsional buckling. For this problem, the objective function (weight) and constraints (σ, δ, h/w) are easily described mathematically and the problem lends itself very well to formal optimization techniques such as those provided in the MSC/NASTRAN Design Sensitivity and Optimization Module.

Benefits of Formal Optimization

The cantelevered beam example demonstrates several benefits to using formal optimization techniques for structural optimization. First, virtually any continuous physical dimension of a problem can be optimized. This is true as long as the user can quantify the impact of each dimension on possible failure modes. For example, the beam in the example problem has a simple rectangular cross-section, but there is no

Formal Optimization Benefits

- Any continuous variable can be optimized
- Any mathematically defined objective function or constraint is allowed
- Techniques to estimate when an optimized solution is found are part of the mathematical formulation
- Design variable linking enhances manufacturability of the optimized design

reason that an I-beam or C-beam could not be modeled with different formulations for stress, deflection and torsional buckling. Also, once any set of optimization parameters (i.e. objective function and constraints) are identified, they can be worked into the optimization. That is, formal optimization is relatively flexible. Another benefit is that their rigorous mathematical formulations include methods that attempt to determine when optimized solutions are reached.

Another benefit of formal optimization is called design variable linking, which has actually grown out of one of its weaknesses. In formal optimization, a few hundred design variables is considered to be a “large” optimization problem. But what if a structure is composed of several hundred discrete components and you wish to optimize each component for eight or nine different design parameters (such as panel height, facesheet thickness, etc.)? This leads to several thousand design variables which is beyond the practical limits of formal optimization. Design variable linking solves this problem by linking design variables to one another, thus reducing the number of independent variables. Techniques for design variable linking include using model symmetry, allowing only linear variations of plate thickness or limiting groups of structural components to a single design parameter value. For example, suppose it is desirable to maintain continuous corrugation spacing from one component to the next. This variable could be “linked” in such a way that one or maybe a few optimum corrugation spacings were allowed for the entire structure. The benefit of design variable linking is that it will often lead to more practical or more manufacturable designs.

Shortcomings of Formal Optimization

While appropriate for solving certain problems, formal optimization has weaknesses. First, even in the above simple example, there is a great burden on the analyst/designer to determine not only the possible failure criteria, but also how they are to be modeled. The analyst must be able to see all of the possible failure modes and then somehow come up with design rules to account for them. The height to width ratio criteria in the example could be argued to be too conservative in some situations (resulting in over-design) and insufficient in others. Also, if a designer wants to try an I-Beam concept for this problem, he must have expert knowledge of the new concepts unique failure modes and generate a completely different set of optimization constraints.

The requirement that the optimization variables be continuous is also restrictive. What if the designer wants the software to determine the optimum structural concept (I-Beam vs. box-beam) or material (steel vs. aluminum)? Even numeric variables may be hard to optimize using formal optimization. Consider trying to optimize a composite ply layup using formal optimization. Two optimization variables for a composite layup are the number of plies and the fiber angle of each ply. Assuming appropriate objective functions and constraints are determined, the optimizer might return a solution like 2.42 plies with ply angles of 27.2° and 58.2°. The resulting 2.42 plies is obviously unacceptable, as it makes no physical sense. Is it appropriate then to use 2 plies or 3? The resulting ply angles *could* be reasonable, but manufacturing constraints would usually limit the fiber angles to more round numbers like 30°, 45° or 60°. Which of these angles is appropriate? Having to answer questions like this makes *automatic* optimization very difficult.

Formal Optimization Shortcomings

- Modeler must determine failure criteria and analysis method
- Only continuous, numeric optimization variables can be used
- Occurrence of local or false optima
- Number of degrees of freedom / optimization variables are limited

Formal optimizers also have the inherent problem of returning local optima. According to optimization theory, the gradients that are calculated and used by the optimizer approach zero in the vicinity of the optimum solution. However, the gradients can also approach zero for non-optimum solutions. These "false" solutions are called "local optimum" solutions. It is not easy for a formal optimization scheme to automatically determine if a local optimum has been found.

Finally, formal optimization techniques are computationally intensive. Small problems like those described above are trivial, but what about designing an entire airframe? For large optimization problems, individual panel level details (like local buckling criteria) must be ignored in favor of simpler strength analyses. In addition, design variable linking *must* be used to reduce the number of variables to a manageable level. Although listed as a benefit above, design variable linking can be restrictive if the user wishes all components to be optimized independently. Even when simplifying assumptions are made and the number of variables is reduced; there is an exponential relationship between the number of independent degrees of freedom (i.e. model size and number of design variables) and the amount of CPU time and memory required for optimization. Problem size is therefore limited and very large problems become unsolvable.

Discrete Optimization

While formal optimization techniques are valuable for certain problems, **automatic optimization of a total structural system** requires a different approach. Discrete optimization is one approach that aids in automating the optimization procedure. With this approach, a system could be reduced to discrete panels and beams with many permutations on the panel and beam concepts generated based on user-defined optimization bounds (such as minimum and maximum gage thickness). After filtering out impractical or impossible designs, each concept is evaluated for all possible failure modes using an extensive list of physics based, closed form analyses. The lightest of the solutions that pass all structural integrity checks is the optimum design.

Discrete Optimization Benefits
(as implemented in **HyperSizer**)

- Designer not required to determine failure criteria
- Can handle non-numeric optimization variables like material or structural concept
- Can handle discrete optimization variables such as number of plies
- No occurrence of local or false optima
- Efficient; No limitation on problem size

Benefits of Discrete Optimization

Discrete optimization is able to address a number of the concerns raised for formal optimization. First, there is no requirement for the designer to determine appropriate failure criteria. This means that the designer/analyst has no need to determine applicable rules-of-thumb to account for structural failure modes. The discrete optimization implemented in HyperSizer checks all possible failure modes for every panel and beam concept by default. The process HyperSizer uses is similar to what an analyst would normally have to do manually. First, the structure is analyzed using Finite Element Analysis (FEA) and then a series of closed form analyses is performed to predict failure modes such as panel local buckling, crippling or ply-by-ply material strength. The difference is that the closed form solutions are performed automatically for every failure mode, every concept, and every location on the structure.

Another benefit of discrete optimization is that any physical or manufacturing constraint can theoretically be optimized in a design. Parameters such as panel or beam concept can easily be optimized. The method can also optimize material selection. Even further, individual properties of a material, such as temper or manufacturing technique can be addressed. With discrete optimization, virtually any combination of material or concept is possible. The only limitations are the imagination of the software developers to give the users variables to play with and efficiently written code to evaluate all possible combinations in a reasonable amount of time.

In addition to being able to optimize non-numerical quantities such as material selection, discrete optimization is also better able to optimize for discrete parameters such as the number of plies in a composite layup. The designer/analyst has the ability to specify acceptable layups, both in number of plies and fiber angle. Therefore, the optimizer would never return a result such as 2.42 plies because this was not specified up front as an acceptable design. This would also mean that the optimizer would not return impractical solutions, such as fiber angles of 27.2° and 58.2° because those solutions would have been disallowed before the optimization took place.

Discrete optimization has no problems associated with local optima. The user can be assured that the solution returned by the optimization procedure is the absolute minimum weight structure given the optimization bounds and permutations selected. Discrete optimization is also much less computationally intensive than formal optimization. There is no exponential relationship between model size and CPU time. Entire structural systems can be designed and analyzed on an Intel based, Windows NT workstation.

Shortcomings of Discrete Optimization

Of course, no method is perfect, and discrete optimization also has weaknesses. First, having no rigorous mathematical formulation, it is rather difficult to quantify whether an actual optimum solution has been found. In other words, while the lowest weight solution given a set of optimization bounds will be returned, there is no way of ascertaining that there are not lower weight solutions outside of those bounds. Of course, while formal optimizers have mathematical criteria for determining whether an optimum has been reached, those methods can still return local optima. Second, variables, including those that are continuous, are treated as if they were discrete. This means that the optimum value of a continuous variable can actually lie between two discrete values. If the user desires that continuous property values be closer to optimum, the optimization

Discrete Optimization Shortcomings

- No way to quantify whether an optimized solution has been reached
- Actual optimum solutions of continuous variables can occur between discrete values
- Design variable linking is not implemented

bounds must be *refined.* HyperSizer allows a user to very easily refine optimization bounds for continuous variables. Due to the continuous nature of the design variables (i.e. *x* and *y*), discrete optimization may not be the best choice to solve the fence problem from the first section.

Another shortcoming of discrete optimization is the absence of design variable linking. Remember that design variable linking originally came about because of the need in formal optimization to reduce the number of independent variables. Discrete optimization as implemented in HyperSizer has no such limitation. To enhance the manufacturability of optimized designs, the linking of certain design variables would be useful. For example, if a user wished to maintain component to component continuity of stiffened panel corrugation spacing, automatic design variable linking would be very helpful.

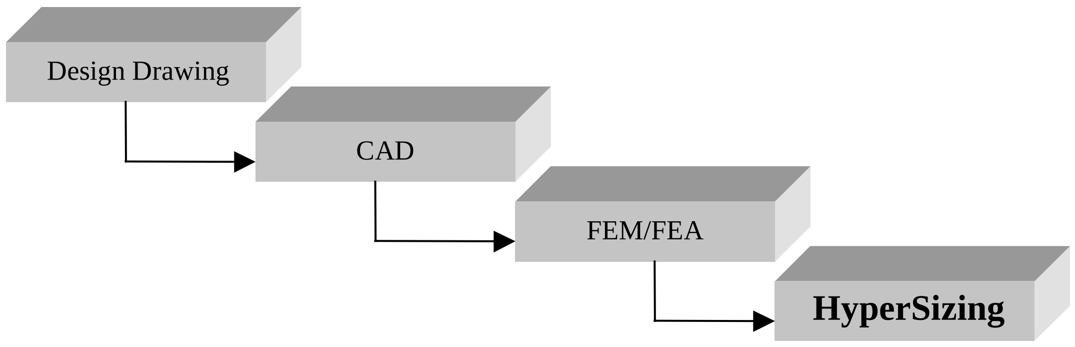
HyperSizer and the Future

The methods discussed in the discrete optimization section are implemented in the structural analysis and optimization package called HyperSizer. Building on over 10 years of research at the NASA Langley Research Center, HyperSizer is able to automatically optimize total structural systems and provide structural designers and analysts with a powerful, user-friendly tool which will dramatically increase productivity.

At Collier Research Corporation, we believe HyperSizer to be an extremely powerful tool but we also recognize that it has weaknesses. We have a full-time development team addressing these issues and working to improve the technology and the interface. In the very near future, HyperSizer will include the capability to automatically refine optimization variables to improve the optimization of continuous variables. For example, the optimization bounds on a continuous variable like facesheet thickness may start out as 0.5" to 2.75". A statistical distribution of optimum solutions might identify a refined optimization range of 1.25" to 1.75", which would in turn be used as bounds for a new discrete optimization. By automating this procedure, continuous variables (e.g. the length and width of the fence problem) could effectively be optimized using discrete optimization with a minimum of user interaction. In addition, an automated variable refinement scheme could include convergence criteria that allow the user to determine when an optimized solution has been achieved. The development of discrete design variable linking is also under way. Similar to the continuous design variable linking in formal optimization, this capability will allow a designer to automatically enhance the manufacturability of an optimized design by ensuring continuity of certain variables between discrete components.

One day soon, we believe that automated structural ‘Sizing’ (i.e. HyperSizing) will become an industry standard in structural design. Its efficiency and ability to run on a relatively inexpensive Windows NT workstation will allow every designer and analyst to perform detailed structural design from their desktops.

HyperSizer provides the next logical step in structural design automation in the progression:



As automated sizing becomes less a convenience and more of a necessity, HyperSizer will be leading the way.