

CONFIGURATION SYNTHESIS OF A VARIABLE COMPLEXITY TRUSS

Erik K. Antonsson, Ph.D., P.E.

Fabien Nicaise

Tomonori Honda

Abstract

One of the key challenges of engineering complex systems is to rapidly explore the range of possible solutions. As systems become more complex, the time required to generate candidate solutions can take prohibitively long, forcing the engineer to make decisions based on poor or missing data. Genetic Algorithms are one method to help explore the design space to synthesize useful designs.

This paper explores the use of Genetic Algorithms (GA) for designing structures for spacecraft. The basic GA is modified to include multiple levels of search to help reduce the threshold required to vary the topology of the structure. The method is validated on a simple, easily verifiable design task.

Finally, the method is applied to the structure of the NASA Mars Science Laboratory Descent Stage. This is a large structure that supports many elements with many constraints, precluding simple solutions.

Keywords: Design synthesis, adaptive design methods, Genetic Algorithms

1. Objectives

Spacecraft must efficiently perform a multitude of tasks while meeting stringent performance requirements, such as fitting inside the fairing of a launch vehicle. Balancing these requirements can be extremely challenging. As spacecraft become larger, more complicated, more capable, and more integrated, this challenge is becoming ever more difficult.

This paper presents a method for synthesizing design concepts for payload structures like the NASA Mars Science Laboratory (MSL) Descent Stage structure. The method is first applied to a simple test case to verify functionality. Then the method is applied to the Descent Stage to synthesize the structure.

The Descent Stage structure supports the equipment required to land the MSL rover as well as the rover itself during the entire mission. Designing such a structure requires balancing the load bearing capability of entry, descent, and landing hardware; ensuring that hardware fits inside the aero shell; and leaving enough volume for the rover itself.

2. Methods

The basic approach for formal design synthesis is

1. Determine a metric to compare different designs
2. Generate design configurations.
3. Evaluate the performance of each configuration.
4. Retain the best performing designs.

The metric used is based on aggregated preference from the Method of Imprecision. We chose Genetic Algorithms as a base algorithm for synthesis [2,3], because the novel configuration for the MSL Decent Stage requires variable complexity¹. Conventional convex optimization techniques do not work for this preference landscape because there may exist many local maxima. To implement a modified Genetic Algorithm to synthesize truss structures, the following items must be defined: Genetic structure, fitness landscape (which is equivalent to the metric), and operators for the genetic structure.

2.1. Fitness Definition

The first step of a Genetic Algorithm optimization is to define gene structure to be used. For each truss, there are two arrays of data, one for storing node information and another for storing connectivity (link) information. The Node information consists of a 3-D vector of real numbers representing the Cartesian coordinate of the node. The node number for each node is implicitly kept by the index. The connectivity array keeps the two end node numbers, cross-sectional area, and the material identification number. (There is separate data structure that keeps material properties for different materials that are usable by any link.) The number of nodes and links can be different from individual to individual and the number of nodes is not limited. (The number of nodes is controlled using survival of fittest, i.e., it will stabilize during the GA run to an optimal number of nodes.) In addition, the constraints and loads applied at the prescribed nodes are kept as another vector array. The input file for the code includes these prescribed nodes, constraints, force vectors, and material properties.

The second step is to define the fitness landscape. The main concern is that the landscape must provide selective pressure, an incentive for the Genetic Algorithm to converge to at least one set of feasible solutions. A feasible solution to the design problem here is any rigid truss that satisfies geometric constraints (in the form of keep out zones) and supports given load(s). To achieve this fitness landscape, the fitness level is divided into four regions:

1. The truss is a mechanism, not a structure
2. The truss is a structure, but does not support the load(s)
3. The truss is a structure and supports the load, but does not satisfy the keep out constraints.
4. The truss meets all the prescribed criteria.

As the fitness of the truss improves, and ascends through the levels, the truss becomes a feasible truss.

The following are fitness levels given to trusses in first three levels. If the truss is a mechanism, it will obtain a fitness using following formula:

$$fitness = \frac{1 \# \text{ of dof removed}}{4 \text{ total \# of dof}} \quad [1]$$

¹ The number of nodes and links are allowed to vary

where *dof* is degree of freedom. This is essentially a measure of how well the solution is constrained. Values are always in the range [0,.25).

Next, if the truss is a structure, it is checked to determine whether it passes through all of the load nodes and sufficient ground nodes. If it does not, then the fitness for the truss be given by:

$$fitness = \frac{1}{4} + \frac{1}{4} \frac{\# \text{ of load supportable}}{\text{total \# of load}} \quad [2]$$

again, the fitness is a measure of how well the truss supports the requisite loads. Values are always in the range [.25,.5).

Once a truss has evolved enough to support a load, it is checked to determine whether the geometric constraints, in form of keep out zones, are satisfied and the fitness is given by:

$$fitness = \frac{1}{2} + \frac{1}{4} \frac{\# \text{ of link violating KO zones}}{\text{total \# of links}} \quad [3]$$

this is a measure of percentage of links that are at fault. The values are in the range [.5,.75).

Only trusses that satisfy these basic constraints are evaluated for performance, such as natural frequency, mass, and factor of safety. Because calculation of these performances is comparatively expensive, these first three levels constitute a pre-check to reduce the computational time required.

In the last level, fitness is shifted to aggregated preferences, defined in the Method of Imprecision [4,5]. This shift compensates for the fact that the fitness should be at least 0.75 because a truss that reaches this stage of fitness evaluation at least is rigid load-supporting truss that satisfies the geometric constraints. The fitness for the truss will be given by:

$$fitness = \frac{3}{4} + \frac{1}{4} \mu_{agg} \quad [4]$$

where μ_{agg} is the aggregated preference, giving values in the range [.75,1).

The following performance variables for the truss are used: mass, lowest vibration frequency, and factor of safety for stress. These criteria are also commonly used by JPL when evaluating truss design. To be consistent with JPL's standard, MSC Nastran is used to calculate the lowest vibration frequency and factor of safety.

The preference for mass is given by:

$$\mu_m(m) = \begin{cases} 1 - \sqrt{m/M} & \text{for } m \leq M \\ 0 & \text{for } m > M \end{cases} \quad [5]$$

where $\mu_m(m)$ is the preference function for mass, m is the mass of the truss being evaluated, and M is maximum mass cut-off.

The preference for natural frequency has the form:

$$\mu_f(f) = \begin{cases} 0 & \text{for } f < f_{\min} \\ \frac{\text{Log}_{10}(f) - \text{Log}_{10}(f_{\min})}{\text{Log}_{10}(f_{des}) - \text{Log}_{10}(f_{\min})} & \text{for } f_{\min} \leq f \leq f_{des} \\ 1 & \text{for } f > f_{des} \end{cases} \quad [6]$$

where $\mu_f(f)$ is the preference function for natural frequency, f is lowest frequency mode of the truss, f_{\min} is minimum requirement for frequency mode, and f_{des} is desired minimum lowest frequency mode.

Also, the preference for stress is defined using factor of safety and has the similar form:

$$\mu_s(S_f) = \begin{cases} 0 & \text{for } S_f < 1 \\ \frac{\text{Log}_{10}(S_f)}{\text{Log}_{10}(S_{f_{des}})} & \text{for } 1 \leq S_f \leq S_{f_{des}} \\ 1 & \text{for } S_f > S_{f_{des}} \end{cases} \quad [7]$$

where $\mu_s(S_f)$ is the preference function for stress, S_f is factor safety of the truss, and $S_{f_{des}}$ is the desired factor of safety.

Finally, these preferences must be aggregated to final preference using formula developed by M. Scott and E. K. Antonsson [5]:

$$\mu_{agg} = \left(\frac{\omega_m \mu_m^s + \omega_f \mu_f^s + \omega_s \mu_s^s}{\omega_m + \omega_f + \omega_s} \right)^{1/s} \quad [8]$$

where μ_{agg} is the aggregated preference, ω_i are the respective weights, and s is the degree of compensation.

2.2. Fitness Smoothing

It is well known that the convergence of a Genetic Algorithm depends on the smoothness of the fitness landscape. Because of the choice of encoding, a change in value of one of the end nodes in the connectivity data could lead to drastic changes in the fitness value. This leads to poor convergence unless some measure is taken to improve the convergence.

Accordingly, the following three loop approach has been developed, in which each sub-loop will have a better possibility of convergence than brute force optimization over the global fitness landscape (see Figure 1). The innermost loop only optimizes over the cross-sectional area of the truss elements if the truss satisfies the minimum requirement, (i.e., fitness is at least 0.75.) The middle loop optimizes the connectivity of the truss. The outer loop optimizes over the node locations. Thus, the smoothing of the fitness landscape is intended for design regions where the truss is at least rigid, load bearing, and satisfies the geometric constraints.

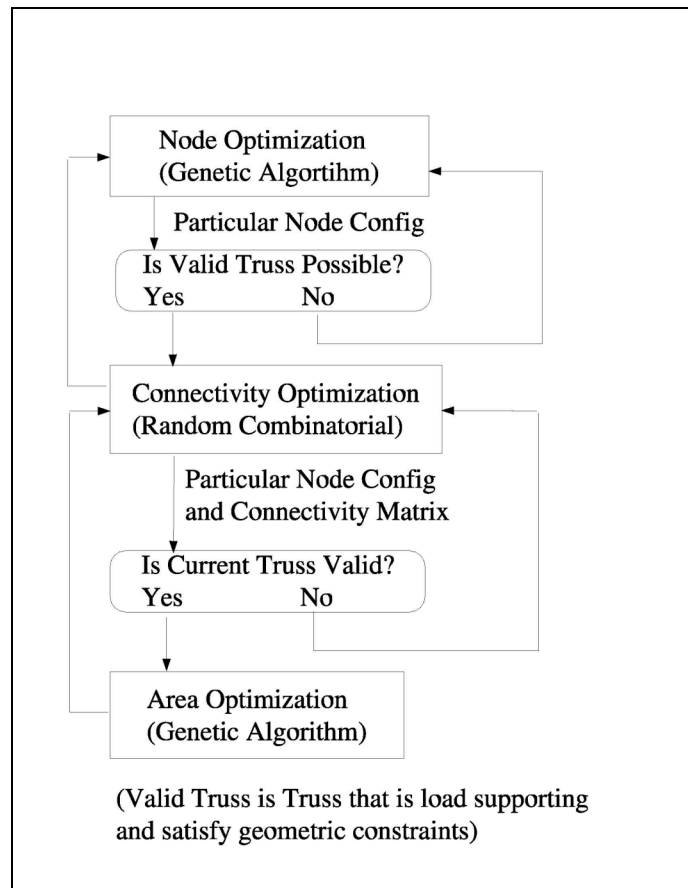


Figure 1 - Modified Genetic Algorithm Loop

At the inner loop stage, the objective (fitness) function is a function of cross-sectional area only. As long as the preference function is continuous with respect to the performance variables, this objective function will be a continuous function of area. To optimize this area, a simple Genetic Algorithm is used with the following mutation and crossovers. Area mutation is achieved by choosing new areas from a Gaussian distribution centered around the current area with a standard deviation of 20% of the current area. Cross-sectional area crossover is achieved by randomly selecting the area for each link from the areas of the two parents. The population size for area optimization is usually 5 to 10, and iteration is usually for 4 to 8 generations. The population and generation size is small because the main function of this inner loop is to reduce the chance of eliminating a truss because of poor area choices. Also, when the outer loops start converging, this inner loop will be less necessary.

At the middle loop stage, pure random combinatorial optimization is used to optimize the connectivity matrix. However, before the connectivity is optimized, a new truss with all links not in violation of the keep out zones is created. This truss is used to determine if, given the nodal configuration, it is possible to create a load-supporting truss that satisfies the geometric constraints. If it is not possible, then the connectivity optimization is skipped for that particular truss. If it is possible, then several links are randomly turned on or off to create new trusses. The best one is chosen from 5-10 trusses generated.

In the outer loop, node locations are optimized. If the two inner loops synthesize good-quality alternative configurations, then the fitness landscape should be continuous with respect to the nodal coordinates. To optimize the node locations, a Genetic Algorithm with following mutation and crossovers is used. Node mutation starts by randomly selecting how many nodes to add or subtract using a discrete normal distribution with mean of 0 and

standard deviation of 1. When a node is to be removed, it is randomly selected from the free nodes (nodes that are neither base nodes nor load nodes.). When a new node is to be added, it is randomly chosen from the entire space. Finally, all nodes are allowed to shift location with probabilities obtained from a Gaussian distribution with appropriate parameters. Crossover is performed by random selection of free nodes from each of the two parents.

3. Results

3.1. Test Case

Several truss structures were synthesized capable of supporting the required load. These trusses exhibited major topological change during their evolution.

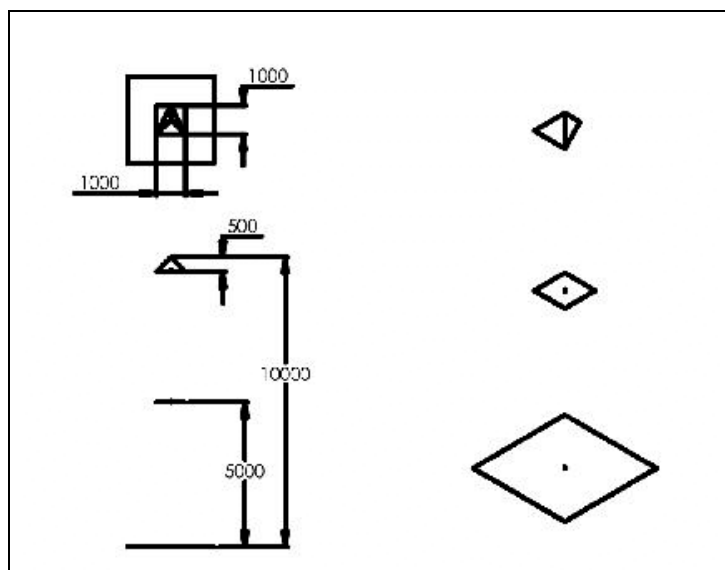


Figure 2 - Basic set-up of the test case.

Figure 2 shows a truss that consists of a mounting plane, where the truss can be grounded, a load application point, and a planar "keep-out" zone. The goal is to connect the ground plane to the load while avoiding the keep-out zone. The keep-out zone is placed in such a way that the trivial solution is disallowed. For this simulation, only the centerline of the member has to clear the keep-out zone.

The fitness function for the Genetic Algorithm is described in the above section.

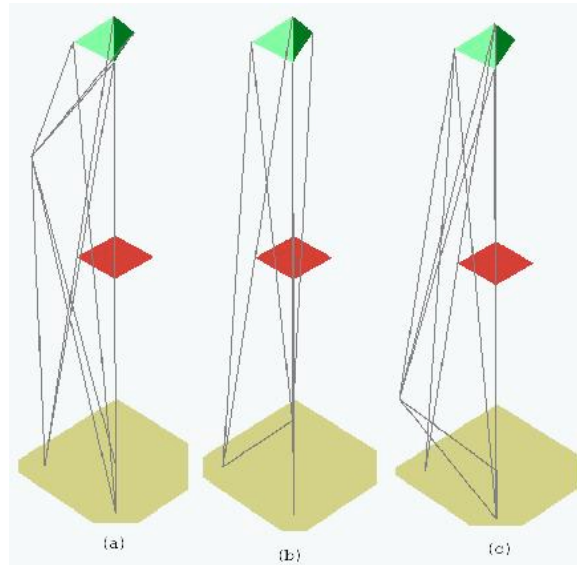


Figure 3 - Generating a solution to the test case, showing progress at generations (a) 8, (b) 26, and (c) 32.

A sample of the evolution is shown in Figure 3. These highlight the changes to the topology of the structure during the evolution. Notice the initial configuration is a simple triangular truss and passes through the keep-out zone. Subsequent individuals then explore the possibilities by adding new nodal points and links. Figure 4 shows the final configuration of a slightly modified test case.

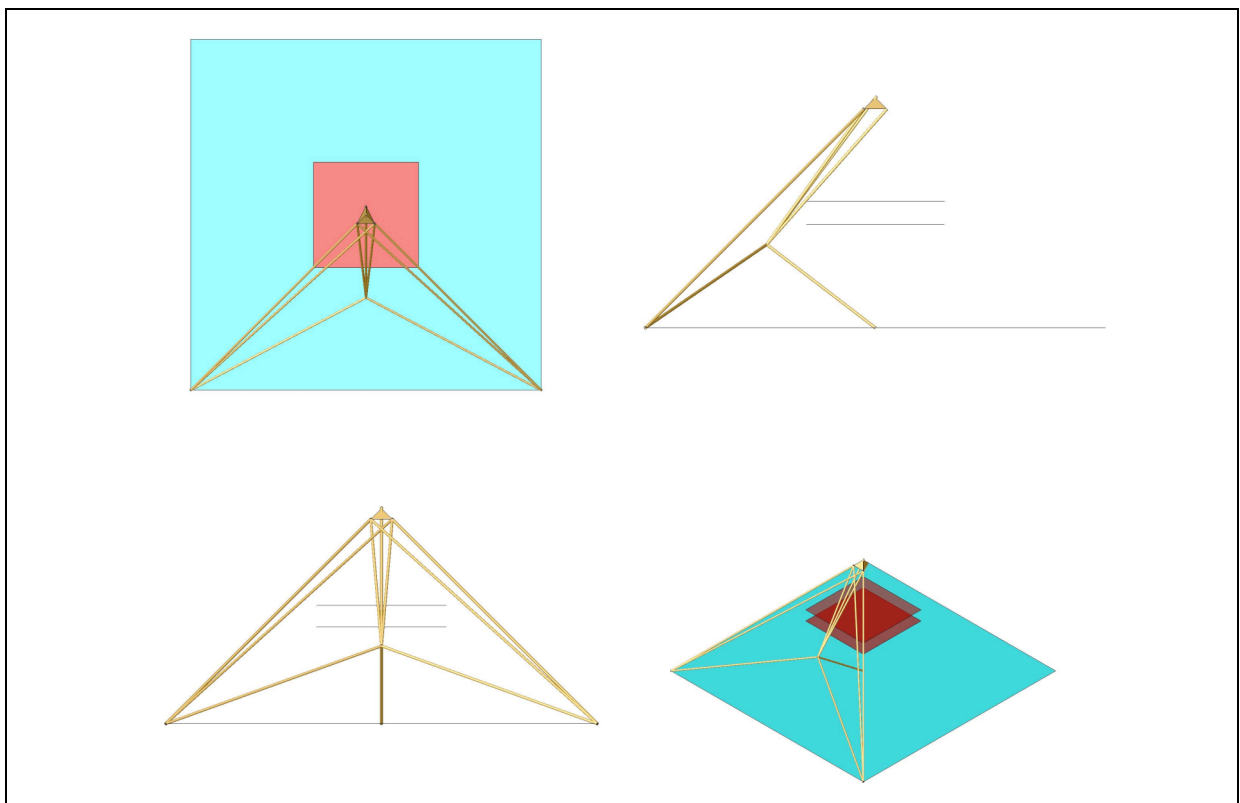


Figure 4 - Final Configuration

3.2.Design Case

Having validated the method on a simple case, the method was then applied to a real world type of problem.

The NASA Mars Science Laboratory (MSL) is a JPL-led mission to land a large-scale rover on Mars to explore and analyze the terrain. Through various design decisions, this mission will not use the airbag landing made famous recently, opting instead to use a controlled-descent and soft landing. This is made possible by the use of a Descent Stage that will fly the rover through the atmosphere, drop it at its destination, and fly away.

However, to do its job well, the descent stage must carry a lot of equipment. There need to be thrusters to provide the deceleration during the decent. There also need to be tanks of propellant and oxidizer to feed the thrusters. Computers are needed to sense the environment and control the vehicle accordingly. There needs to be an interface to the rover. All this equipment needs to fit within a prescribed aero-shell to protect it during entry. And there needs to be an interface to the cruise vehicle that carries it from Earth to Mars. Last, but certainly not least, there needs to be a structure to hold all these pieces together.

It is with this last item that is the subject of this example: a structure to support all the requisite equipment, not pass through any of that equipment, and fit within an aero-shell while maintaining an adequate margin of safety and keeping the mass low. Figure 5 shows all the requisite elements that need to be mounted. The goals of the design of the structure are fairly simple: support all the elements while keeping a natural frequency above 28 Hz and the mass low.

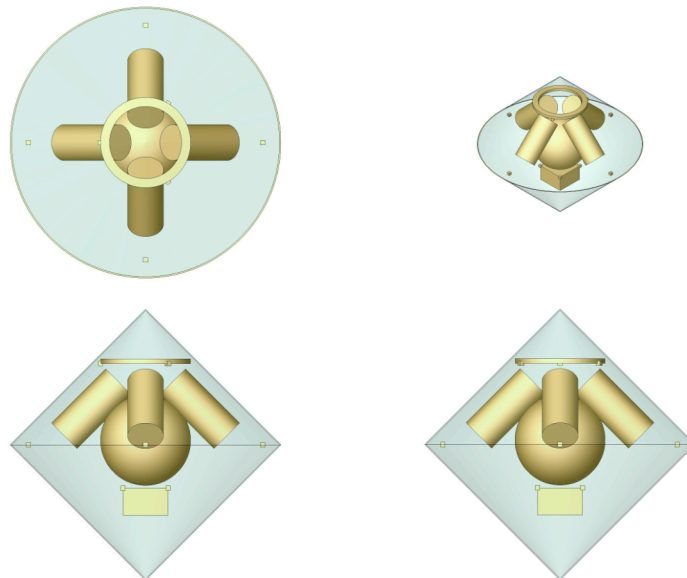


Figure 5 - Set-up for the MSL design case. (Clockwise from top-left view: Top view, Isometric, Side view, Front view.)

Figure 6 shows the results of a run synthesizing the structure using the method described. Notice that all the elements are supported and no strut goes through any other strut or piece of equipment.

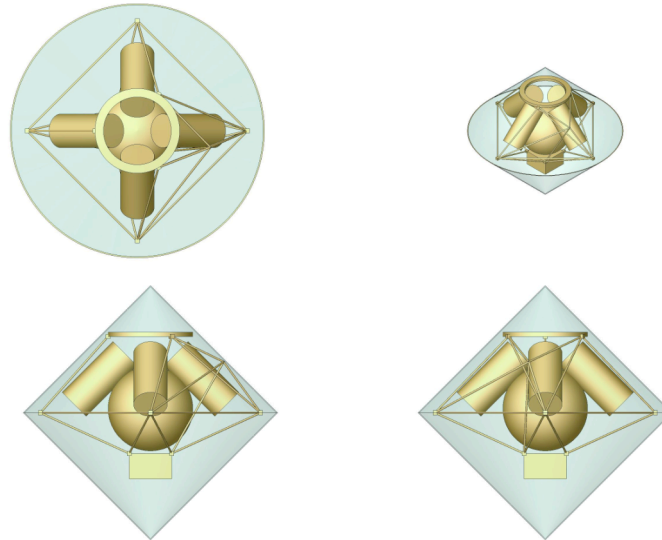


Figure 6 – Evolved solution for the MSL design case. (Clockwise from top-left view: Top view, Isometric, Side view, Front view.)

4. Conclusions

In this paper we have shown how evolutionary methods can be applied to the design of spacecraft structures. The method shows flexibility and can eventually be extended to include other engineering disciplines. Future work will focus on optimizing whole systems, rather than single functional areas. This will eventually lead to better designed and better performing spacecraft in the future.

5. Reference

- [1] Erik K. Antonsson. Imprecision, Trade-Offs and Negotiation in Engineering Design. In *1998 NSF Design and Manufacturing Systems Grantees Conference*, Monterrey, Mexico, January 1998. NSF.
- [2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.
- [3] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.
- [4] Erik K. Antonsson and K. N. Otto. Imprecision in Engineering Design. *Special Combined Issue of the Transaction of the AMSE commemorating the 50th anniversary of the Design Engineering Division of ASME*. pp. 25-32 1995.
- [5] Michael J. Scott and Erik K. Antonsson. Compensation and Weights for Trade-offs in Engineering Design: Beyond the Weighted Sum. *jmd*, 2003. Submitted for review, November, 2003.

6. Acknowledgment

This material is based upon work supported, in part, by: NASA's Jet Propulsion Laboratory as part of the MSL Descent Stage Genetic Algorithm Optimization Study and a Moore Fellowship supported by the Gordon and Betty Moore Foundation.

Corresponding author:

Erik K. Antonsson, Ph.D.,PE
Engineering Design Research Laboratory
California Institute of Technology, Mechanical Engineering
M/C 104-44
1201 E. California Blvd
Pasadena, CA 91125
United States of America
(626)-395-3790
FAX (626) 583-4963
erik@design.caltech.edu