A Novel Modeling Platform for Characterization and Optimal Design of Micro-Architected Materials

Scott W. Godfrey\(^1\) and Lorenzo Valdevit\(^2\)
University of California at Irvine, Irvine, CA 92697-3975

Recent advances in multi-scale manufacturing enable fabrication of hollow-truss based lattices with dimensional control spanning seven orders of magnitude in length scale (from ~50nm to ~10cm), offering tremendous potential for multifunctionality. Topology optimization is essential to realize the full potential of these micro-architected materials. This paper presents a novel optimal design and modeling platform, consisting of four interconnected tools: (i) a geometric modeling algorithm; (ii) a meshing algorithm; (iii) an optimal design code; and (iv) a communication interface with a commercial Finite Elements program (Abaqus). The powerfulness of the proposed platform is demonstrated for the optimization of specific stiffness in pyramidal hollow micro-lattices.

I. Introduction

Recent advances in multi-scale manufacturing enable fabrication of hollow-truss based lattices with dimensional control spanning seven orders of magnitude in length scale (from ~50nm to ~10cm) [1-3]. Fig. 1 illustrates the architecture for hollow Nickel lattices, manufactured by HRL Laboratories with a novel process combining Self Propagating Polymer Waveguide polymerization and electroless Nickel deposition [2, 3]. These micro-architected cellular materials (which can be polymeric, metallic, ceramic or hybrid) offer tremendous potential for multifunctionality, achievable through a synergistic combination of architectural optimization and base materials selection (the ability to deposit the base material in the form of films with thickness lower than a micron allows exploitation of well-established mechanical size effects at the nanoscale [4]). The hollow nature of these lattices and their small dimensions complicate the mechanical analysis, by emphasizing the role of the nodal geometry (e.g., fillet radii, thickness variations) on strength and stiffness. The implications are that (a) standard geometry modeling and meshing algorithms are often inadequate for modeling of micro-architected materials, and (b) topology optimization must be implemented in a Finite Elements framework, as accurate objective functions and/or constraints are often not available in analytical form. This paper presents a novel modeling platform, consisting of four interconnected tools: (i) a geometric modeling algorithm; (ii) a meshing algorithm; (iii) an optimal design code; and (iv) a communication interface with a commercial Finite Elements program (Abaqus). The

---

1 Graduate Student, Mechanical and Aerospace Engineering Department, AIAA Student Member.
2 Assistant Professor, Mechanical and Aerospace Engineering Department, AIAA Member.
II. Description of the computational platform

The modeling platform that was developed comprises three key components: (i) a geometric modeling and meshing algorithm, (ii) a data management system and (iii) an interface for automatic communication with a commercial Finite Elements package (Abaqus, in the current implementation). The entire platform is almost completely parameterized and automatized, with minimal input required from the operator. Input parameters are the cellular lattice topology (e.g., a periodic pyramidal lattice), the space of design parameters (e.g., the allowable ranges of each dimension, possibly interdependent), the objective function and the constraints. The code may be designed to sample the entire parameter space (as in its current implementation) or include a family of optimizers (both discrete and gradient-based); in both cases, the code identifies the specific designs to be modeled, automatically generates complex geometries and meshes as appropriate, creates input files for Abaqus execution, controls the Abaqus job submissions, extracts results from the Abaqus output files and handles the results. Naturally, this platform is ideally suited to optimize lattices for situations where close-form analytical expressions of objectives and constraints are unavailable (hence the adoption of Abaqus for calculation of both), but more simple problems (whether mechanical or functional) are equally possible. The main components of the code are described below.

A. General Code Architecture

The architecture of the entire code is schematically depicted in Fig. 2. Colored boxes represent cohesive system components, or modules. In the current implementation, the Data Management system is focused on the mechanical analysis of periodic lattice structures and an appropriate geometry engine is attached. Alternative engines can be added with minimal implementation effort. Double-boxes represent process boundaries and are effectively single executable programs. Single square boxes denote complicated, multi-object systems and rounded boxes represent single objects; stacks represent arrays. The TCP/IP boundary allows for distributed execution; processes are not required to be on the same system or even nearby each other. Simulations are client-driven so any client that can see the server on a network can participate, provided resources are sufficient. The vast majority of our simulations have been performed on a single Quad-Opteron server communicating with itself locally.

Data Management System

The Database Manager provides a user interface for interacting with loaded databases and preparing simulations for operation. Each database can be an arbitrarily complex object, whose definition depends on the type of analyses being performed. For systematic exploration of the parameter space (which is the operation described in sec. III), the databases are created with a number of constants describing the structure (topology, angle, fillets, defect type, defect magnitude, material), while leaving two dimensionless unknowns to be swept: \( \frac{L}{D} \) and \( \frac{t}{D} \). Each of these databases contains an array of Database Part objects, each of which corresponds to a discrete \( \frac{L}{D} \) value. Each Database Part object includes a quadratic...
optimizer that calculates t/D to match a prescribed relative density to a high degree of precision. With each t/D value, a Database Entry is constructed, which is a fully described design corresponding to a particular density within the database. Each Database Entry is also a ‘STORM Particle’ object by inheritance; this provides a standard interface by which the STORM system can request it to perform a variety of tasks including creation and execution of Finite Elements jobs. Each Database Entry object has an arbitrary number of Database Boundary Property objects associated with it. Each Boundary Property contains the formal description of a particular class of simulations relating to that design, e.g., boundary conditions, lattice dimension, element type, property of interest (e.g., compression modulus, shear modulus…). Simulation results reside here, including stresses, strains, reaction loads, etc.

The current implementation of the Database Management System systematically and efficiently samples the entire design space and provides multi-objective optimal design tools only in post-processing mode. In other words, the Database Management System submits FE job requests serially and not based on the results of previously submitted jobs. Modest alterations have been considered for serving closed-loop optimizers residing both in the local process and as remote clients.

**Geometry Engine**

The geometry kernel is composed of a specialized volume processing system for generating blended tubular lattice structures of arbitrary configuration composed of members of arbitrary shape. The output translators take the raw geometric data and simulation descriptions and form them into input appropriate for specific analysis types (e.g., generates input files for ABAQUS analysis). Additionally, an object-based automatic-programming interface generates job-specific analysis scripts for processing the simulation results. Intentionally, the system is not ‘device specific’ and with modest effort could support any number of output formats; this theme persists throughout.

**Network Interface Management System (STORM)**

The STORM Server is a highly threaded object that manages inbound clients (e.g., the commercial FE package ABAQUS) and acts as an interface layer between the Data Management System and the simulation application. When being prepared for operation, each Database triggers all its designs (Database Entry objects) to generate a set of related analysis jobs (STORM Job objects). Each job corresponds to a discrete simulation, and contains information on its identity (session and job number, type) as well as pointers in memory for where to find and where to put relevant information (e.g., the design of interest, boundary conditions, and results). To enable more complex simulations and dependencies, each job may link to other jobs and other resources on disk.

The STORM Server communicates with Finite Elements Analysis packages through TCP/IP protocol. Administrative interaction with the STORM Server is enabled by a STORM Network Controller, which merely acts as a remote user interface. Dedicated modules can be written to optimize the interaction with commercial packages. For the case of the FE package ABAQUS, two modules are implemented: (i) The STORM CAE Client, written in Python and executed within CAE, is responsible for communicating availability to the STORM Server, receiving job data, initiating and monitoring simulations, then analyzing and returning results to the server. (ii) The STORM HYDRA Client is a multi-role tool for managing ABAQUS CAE, including an interface for programming the clients behavior, initiating an appropriate number of client processes, and the monitoring and respawning of new clients as necessary. This also allows for rudimentary manual load-balancing and license management of the system when job types of different performance characteristics are available for execution.

The STORM Server handles and processes all STORM jobs. Once the STORM client (e.g., ABAQUS) confirms resource availability (in terms of computational power and licenses), the STORM Server selects one job from the queue and, through the STORM Particle interface, retrieves the Database Entry associated with it (containing all the geometric information), adds the requested Database Boundary Properties (lattice dimension, boundary conditions, etc…), and invokes the Geometry Engine (described above) to create a suitable mesh and an a properly formatted input file for ABAQUS analysis. The input file is then sent to the ABAQUS Client via TCP/IP, and the result of the analysis is stored backed into the Database Entry. This process is schematically represented by the arrows in Fig. 2.

**B. Geometric Modeling and Meshing**

This component of the modeling platform (contained in the Geometry Kernel) is responsible for parametric generation of solid and/or shell geometries and their adequate meshing for Finite Elements analysis. Conceptually, this operation is carried out in five subsequent steps: (1) Definition of lattice points; (2) Definition of nodal connectivity (members locations); (3) Member profiling; (4) Generation of fillets and smooth member connections; (5) Meshing and assembly of the lattice (Fig. 3).

American Institute of Aeronautics and Astronautics
1. Definition of lattice points and nodal connectivity

Lattices are defined by two sets of objects, Nodes and Truss Members, contained within an ‘overseer’ Truss Structure object that directs operations. A stick-and-ball molecular model analogy can be used to imagine the structural implementation (Fig. 3a-b). Nodes are seeded periodically through a Cartesian 3D space; each node object only contains its local spatial information and the list of connecting truss members. Truss members are attached to pairs of nodes in order to bound the structural volume and therefore are the root expressions of connectivity. At this level, they are merely geometric connectors, and do not contain any information on the shape and size of the cross-section. The lattice topology is defined by specific algorithms; new algorithms (and hence new topologies) can be continuously added without substantially altering the structure of the code. Without loss of generality, this paper will present a regular, periodic pyramidal lattice as an example. It is important to emphasize that lattice periodicity is not required, enabling the investigation of functionally graded structures.

2. Member profiling

Truss members are assumed to be hollow and are parameterized with a cylindrical coordinate system, \((r, \theta, \lambda)\), with \(\lambda\) the normalized axial coordinate along the truss member length (\(\lambda = 1\) is the nodal distance). The surface of the truss members can be expressed as: \(r(\theta, \lambda) = C(\theta) M_0(\theta, \lambda) M_1(\theta, \lambda) \ldots M_n(\theta, \lambda)\), where \(C(\theta)\) is a simple curve describing the primitive shape of the cross-section, and \(M_i(\theta, \lambda)\) are surface modifiers that introduce arbitrarily complex tapers or distortions along the surface. This allows very systematic analyses of the imperfection sensitivity of hollow lattices. Global (lattice-wide) cross-section variations are possible, although their implementation is more cumbersome. For shell element modeling, the surface is defined at the mid-point of the member thickness, as common in beam modeling. For solid element modeling, actual volumes are generated. At the end of this step, a member might look as depicted in Fig. 3c.

3. Generation of fillets and smooth member connections

Truss members have been described independent of each other and overlap in space at the nodes. To detect surface intersections, each truss member queries its nodes for all adjacent neighbors. The angular coordinate, \(\theta\), is sampled regularly. For each discrete value \(\theta_n\), an optimization routine is invoked to identify the intersection points around the two end nodes. This procedure will produce a surface \(C^0\) continuous at the intersections, provided the surfaces intersect only on single curves (i.e., they are non-reentrant). Actual lattices contain fillets at the nodes (with radii possibly varying in space around the node), and exhibit geometries that are at least \(C^1\) continuous. More sophisticated geometric tools can be employed that guarantee \(C^1\) continuity: one such algorithm (which seeks an intermediate surface simultaneously tangent to both truss members) has been implemented in the code. The mathematical details are omitted for the sake of conciseness (see [5] for details). The final product of this step is a truss member (or a set of truss members, for lattice topologies with non-identical members) with proper end shapes for lattice assembly (Fig. 3d).

4. Meshing and lattice assembly

The complete geometric information of each member, as obtained in the previous step, is combined with the basic lattice description (lattice connectivity) to generate the entire lattice. Each member is regularly meshed within the regions of the \(C^1\) continuous surface (whether with solid or shell elements), and the portion of the fillet that pertains to each member is separately meshed. The mathematical details are omitted for conciseness (see [5] for}

\[\text{Figure 3. Sequence of steps in the geometric modeling platform. (a) Lattice points definition. (b) Truss members definition (nodal connectivity). (c) Truss member profiling. (d) Generation of fillets and smooth member connections. (e) Assembly and meshing.}\]
details). Mesh control parameters (mesh density, bias, etc…) are defined by the operator as initializers. A detail of the final result is depicted in Fig. 3e, for shell elements meshing. An analogous solid elements mesh is alternately available.

III. Specific stiffness optimization in micro-architected materials

A. Problem Statement and lattice topology

This section demonstrates the capabilities of the modeling platform described in Sec. II with a relatively simple mechanical optimization problem. The structure of interest is a hollow lattice with the unit cell topology depicted in Fig. 4. The geometric parameters required to fully define the lattice are the lattice angle, $\theta$, the cell size, $L$ (related to the truss member length by $L = 2t\cos\theta$), and the truss member diameter, $D$, and thickness, $t$. In non-dimensional form, the geometric parameters are $[L/D, t/D, \theta]$. To maximize relevance, it is assumed that the lattice is fabricated with the novel Self Propagating Polymer Waveguide approach [6]. All geometric parameters are bounded as follows, consistently with the limitations of this manufacturing approach [7]:

$$
0.00001 < \frac{t}{D} < 0.2 \\
2 < \frac{L}{D} < 16 \\
45^\circ < \theta < 75^\circ
$$

To mimic the manufacturing process as closely as possible, truss members are connected by fillets with a spatially variable radius ($0.1 < r_f / D < 0.5$). 2D projections of 3D unit-cell meshes generated by the Geometry Kernel for specific parameter choices are depicted in Fig. 5.

The objective function to be maximized is the compressive elastic modulus, $\bar{E} = E / E_s$ (with $E$ and $E_s$ the elastic moduli of the lattice and the base material, respectively), over a range of relative densities, $\bar{\rho} = \rho / \rho_s$ (with $\rho$ and $\rho_s$ the mass densities of the lattice and the base material, respectively).

Figure 4. Geometry for the hollow-truss pyramidal lattice and variables definition.

Figure 5. Generated meshes for three particular choices of geometric parameters: (a) represents a well generated lattice, whereas (b) and (c) are at the boundary of the geometric feasibility region.
B. Density Calculations

Simple geometric considerations provide an analytical value for the relative density of the hollow lattice depicted in Fig. 4 [1]:

\[ \rho_{\text{hollow}} = \frac{2\pi}{\cos^2 \theta \sin \theta} \frac{D}{\ell} \frac{t}{\ell} \]  

(1.2)

This equation is only accurate to first order in $D/\ell$ and $t/D$ and neglects any mass clipping at the nodes. For comparison, the relative density of a solid lattice of the same topology (accurate to first order in $D/\ell$, and neglecting mass clipping at nodes) is given by:

\[ \rho_{\text{solid}} = \frac{\pi}{2\cos^2 \theta \sin \theta} \left( \frac{D}{\ell} \right)^2 \]  

(1.3)

The Geometry Kernel described above (Sec. II.B) can be used to generate lattices of desired dimensions (both hollow and solid), and calculate their relative density by solid modeling numerical approaches rather than by elementary first-order geometry. This allows modeling real node effects, including mass overlaps and the existence of finite fillet radii, and provides useful information on the range of validity of the analytical expressions.

A comparison of numerical and analytical density calculations is provided in Fig. 6, for both solid and hollow lattices. The angle is fixed at $60^\circ$ in both cases, for the sake of illustration, although the entire range $45^\circ < \theta < 75^\circ$ was modeled. For hollow lattices, the member thickness is fixed at a very low value, $t/L = 0.0001$ (representative of thicknesses easily achieved with a combination of self propagating polymer waveguide polymerization and electroless metal deposition [2, 3]). Numerical results are provided for four values of the fillet radius ($r_f/D = 0, 0.5, 1, 1.5$). Analytical predictions are reported as dashed lines (Eq. (1.3) in Fig. 6a and Eq. (1.2) in Fig. 6b). A number of results are worth emphasizing:

(a) For solid lattices (Fig. 6a), the analytical model is accurate up to relative densities ~10%, whereas for thin-walled hollow lattices with stubby members ($D/L=0.25$) the analytical model overpredicts the density by as much as 100% - even at densities as low as 0.1%. This result has important practical implications on the modeling of lightweight hollow lattices: whereby Eq. (1.3) can be comfortably used to model nearly all solid lattices of technological importance ($\rho < 10\%$) [8], Eq. (1.2) should be used with extreme caution in modeling hollow lattices.

(b) The analytical prediction for solid lattice density (Eq. (1.3)) provides unphysical results at aspect ratios, $D/L>0.37$; conversely, the solid modeling remains accurate over the entire space of parameters.

(c) In contrast to solid lattices (for which the density is an increasing function of the member stubbiness), for

![Figure 6. Relative density of (a) solid and (b) hollow lattices, calculated by solid geometric modeling, as a function of the truss member aspect ratio and the node fillet radius. For the sake of illustration, the angle is fixed at $60^\circ$ and the hollow member thickness is fixed at a very low value, $t/L = 0.0001$. First order analytical predictions (Eqs. (1.2) and (1.3)) are reported for comparison. Notice that for solid lattices, the analytical model is accurate up to relative densities ~10%, whereas for thin-walled hollow lattices with stubby members the analytical model overpredicts the density by as much as 100% even at densities as low as 0.1%.](image-url)
hollow lattices, the relative density has a peak at $D/L=0.25-0.35$, depending on the fillet radius. This is a real node effect, which cannot be captured with first-order analytical models, and can be rationalized as follows: For hollow lattices the volume can be simply expressed as the surface area multiplied by the surface thickness. As the diameter increases the surface area gradually increases and so does the error in the analytical model due to beams interacting at the nodes. At some point the increase in surface area of the hollow members is exceeded by the loss of surface area lost to overlap and clipping at the nodes and the total surface area declines, thus reducing the relative density.

(d) The fillet radius has a fairly minor effect on the relative density, for both solid and hollow lattices, although it affects the range of achievable member aspect ratios. For thicker walls, the effect of the fillet is expected to increase.

Figure 7 shows numerically calculated relative density contours for 60° hollow lattices. Extremely low aspect ratios ($L/D<3$) and large thickness to diameter ratios ($t/D>0.1$) are not geometrically feasible with the node generation approach described in Sec. II.B), as they would induce non-smooth fillets and/or extremely distorted elements nodal regions. Each point on the map denotes a fully defined and unique design. As all the feasible designs within the prescribed range of parameters are depicted on this map, Fig. 7 can be used as a design tool for lattice geometry generation.

C. Stiffness Calculations

The Young’s modulus along the $z$ direction for an infinite lattice with the topology depicted in Fig. 4 (whether solid or hollow) can be estimated by beam theory considerations. It is essential to notice that the lattice in Fig. 4 is not statically determinate; that is, if the nodes are modeled as pin joints, the structure is a mechanism and would be incapable of supporting any load. The implication is that truss modeling is inadequate to calculate the stiffness (and/or the strength) and beam modeling should be used instead. By using the Principle of Virtual Work [9], and accounting for stretching, bending and shear compliance, the stiffness can be written as:

$$E = \frac{E}{E_s} = \frac{3\pi \sin \theta \left( \frac{D}{\ell} \right)^3}{\cos^4 \theta \left( \frac{t}{\ell} \right)} \left( \frac{1}{1 + \frac{3}{2} \tan^2 \theta \left( \frac{D}{\ell} \right)^2 + \frac{3}{4} (1 + \nu) \left( \frac{D}{\ell} \right)^2} \right)$$

where $\nu$ is the Poisson’s ratio of the constituent material. Although this model captures the global mechanical behavior of the truss members (assuming that they are slender), it ignores the details of the nodal geometry, the mechanical deformation at the nodes and the local wall deformation. To fully capture all these effects and assess their importance over a wide parameter space, a very large series of geometrically non-linear Finite Elements simulations on very detailed geometric meshes is required. The modeling platform described in Sec. II is ideal for this task. The parameter space described in Eq. (1.1) was discretized in 5,600 designs and for each design a single unit-cell mesh was generated by the Geometry Kernel. Shell elements were used throughout, given the very low $t/D$ ratio for the lattices of interest. To represent an infinite lattice, the following boundary conditions were applied (refer to Fig. 4 for the coordinate system): (a) the nodes on the $-z$ face were constrained from moving in $z$ but allowed to move along $x$ and $y$; (b) the nodes on the $+z$ face were subject to a downward displacement $\delta_z = -0.00001H$ (equivalent to a strain of 0.001%); (c) the nodes on the $\pm x$ faces were all imposed to move by the same unspecified amount in the $x$ direction, and let free to move along $y$ and $z$; (d) the nodes on the $\pm y$ faces were all imposed to move by the same unspecified amount in the $y$ direction, and let free to move along $x$ and $z$; (e) finally, all rotations were prevented at all nodes on the $\pm x$, $\pm y$ and $\pm z$ faces. Notice that these boundary conditions assume that the
deformation field of a whole lattice has the same periodicity of the unit cell. Although this assumption is expected to be realistic for slender lattices with densities in the range 1-10% (where nodes are sufficiently strong to effectively clamp the truss members), it would probably overpredict the stiffness for extremely lightweight lattices, whose behavior is expected to be node dominated. (Future assessments will investigate alternative boundary conditions, and compare the numerical predictions to experimental characterizations, to fully capture the mechanical behavior of ultra-light lattices.)

Isotropic linear elastic properties were used in the simulations (with $E_s = 210$ GPa and $\nu = 0.3$) and a geometrically non-linear (large deformation) analysis was performed, to capture local wall deformations and rotations. The reaction force, $F_R$, on the bottom face was extracted from the simulations and the Young’s modulus in the $z$ direction was calculated as: $E = (F_R / L^2) / (\delta_z / H)$, with $L$ and $H$ the dimensions of the unit cell (Fig. 4).

FE simulation results for a particular lattice ($t = 500$ nm, $D = 500$ $\mu$m, $L = 4$ mm, $\theta = 60^\circ$) are depicted in Fig. 8 (a 2D projection is shown, although the simulation is 3D): Fig 8a shows the Von Mises stress distribution, whereas Fig. 8b illustrates the magnitude of the nodal rotations. The deformation is amplified 5,000 times for the sake of illustration. Notice that most of the deformation (and hence the stress) is localized around the nodes. Figure 8. Finite Elements results for uniaxial compression along the $z$ direction. The mesh and the entire input file were generated automatically with the modeling platform described in Sec. II. (a) Von Mises stress distribution; (b) Nodal rotation magnitude distribution. Note that stresses and rotations are primarily concentrated around the nodes, with very little strain energy along the member length.

Figure 9. Comparison of analytical (Eq. (1.4)) and numerical predictions for the relative elastic modulus of hollow lattices. (a) Constant relative density, $\bar{\rho} = 0.2\%$; (b) Constant truss member aspect ratio, $L/D = 8$. The lattice has an angle, $\theta = 60^\circ$ in both cases. In (a), notice that the analytical model predicts a modulus which is continuously increasing with the member stockiness, whereas the numerical results show a peak at $L/D=5$. In (b), notice that the analytical model substantially underpredicts the stiffness, with the exception of very low relative densities, $\bar{\rho} < 1\%$, where the converse is true.

8

American Institute of Aeronautics and Astronautics
nodes, with minimal strain energy stored along the bars length.

A notable strength of the modeling platform described in Sec. II is the exceptional computational efficiency. More than 5000 simulations were automatically performed to scan the entire design space. Results for hollow lattices with a 60° angle and relative densities between 0.01% and 10% are depicted in Fig. 10. Note that with this choice of boundary conditions, an optimal truss member aspect ratio, \( L/D \approx 4.5 \), emerges over the entire sampled range of relative densities. Also notice that at extremely low relative densities (\( \rho \approx 0.01\% \)), it is possible to vary the Young’s modulus of a lattice by more than an order of magnitude simply by modifying the aspect ratio, at a constant relative density. As the density increases, the effect of the truss member aspect ratio decreases. We should emphasize that these results apply to the particular choice of boundary conditions used in the simulations. A more detailed analysis of the effect of the boundary conditions is forthcoming.

IV. Conclusions

A novel modeling platform for the characterization and optimal design of cellular lattice structures was presented. The implementation is ideally suited to recently developed hollow-truss-based micro-architected materials, although the fundamental governing principles are extremely general. The usefulness and powerfulness of the suggested approach was demonstrated for the maximization of specific compressive stiffness in periodic pyramidal hollow-truss lattices. The protocol is fully automatic, yet does not require simplified mechanics-of-materials expressions for stiffness and density. The results reveal non-trivial optimal geometries that were not attainable with conventional optimal design approaches. The highly parallelized architecture of the modeling platform, combined with the relative simplicity of FE stiffness calculations, enabled the automatic executions of more than 5,600 FE analyses in a couple of weeks. A more detailed study of the effect of boundary conditions on the stiffness, validated by experimental mechanical characterization, is necessary to extract reliable quantitative information. Such a study is forthcoming. A relatively simple extension of the procedure described in this paper would entail the extraction of elastic buckling loads through linear perturbation of the stiffness matrix (eigenvalue extraction), thus allowing the generation of strength maps for buckling limited designs. This work is currently underway. More complex, non-linear limit load analyses would dramatically expand the computational time, requiring the implementation of a smart optimization module within the Database Management module.

Acknowledgments

The authors acknowledge a fruitful collaboration and continuous discussions with Tobias A. Schaedler, Alan J. Jacobsen and William B. Carter at HRL Laboratories. This work was supported by DARPA under the MCMA program managed by Dr. Judah Goldwasser (Contract# W91CRB-10-0305).

References