The use of EGADS
(and other Solids-based Geometry-Kernel APIs)
for Meshing

Bob Haimes
haimes@mit.edu
Aerospace Computational Design Lab
Department of Aeronautics & Astronautics
Massachusetts Institute of Technology

Overview

- Background
- ESP & EGADS
- Geometric Expressions
- Topology – The Boundary Representation (BRep)
  - Manifold vs. Nonmanifold
  - Parsing the BRep
  - Tolerances
- The EGADS API
- Important Concepts for Meshing
  - Point Queries
  - Surface Degeneracies
  - Watertight BRep meshing
- Closing Remarks

Background – What is a Geometric *Model*?

Discrete

Usually a collection of linear elements (triangles) that together can *approximately* express shapes

- **STL** (STereoLithography) is the standard file format
  - Output from laser scanners
  - Input to 3D printers
- Heavily used in animation / gaming
- Can be modified via:
  - *Subdivision Surfaces* for smoothness – creases?
  - Free Form Deformation (FFD)
  - Applications such as Blender (https://www.blender.org)
- Grids can be considered this *form* of geometry

Background – What is a Geometric *Model*?

Analytic

Collections of *equations*, their coefficients & connectivity that can be used to generate shapes

- Standard file formats:
  - **IGES** – Initial Graphics Exchange Specification
    bag of geometric entities usually without connectivity
  - **STEP** – Standard for The Exchange of Product model data
    has the ability to transmit the complete *Model*
    difficult to write the code necessary to *read* these files
- Proprietary files:
  - Geometry Kernels – Parasolid, ACIS, OpenCASCADE, SMLib...
  - CAD Systems – Unigraphics, SolidWorks, Catia, Pro/ENGINEER
  - Use the appropriate API

Geometry should **not** be considered synonymous with CAD

Background

Why not focus on Discrete?

- Parametric Geometry
  - Discrete requires some form of *Free Form Deformation*. The parameters are the position of Control Points (or ganged CPs) in the FFD box. Cannot support topological changes.
  - With Analytic (and *Constructive Solid Geometry* modeling) the model can be constructed using a parameteriziation that a designer can intuitively understand.
- Control over shape – how close to a NACA airfoil is good enough?
- Surface quality
- Manufacturing – when is a circle a circle?

Having both coupled Analytic & Discrete representations is useful

Background

The Analytic Geometric *Model*

- Geometry: points, curves, surfaces
- Topology: hierarchy, connectivity, orientation and bounds/limits
  - Provides an unambiguous partitioning of space where floating point cannot – due to tolerances
- The Boundary Representation (BRep)

- Perfect for Object Oriented software design
- Older *Solid* Modeling interfaces hide *Objects* with integer tags
- `EGADS` is *Object-based* – allows mixing programming languages

Engineering Sketch Pad (ESP)

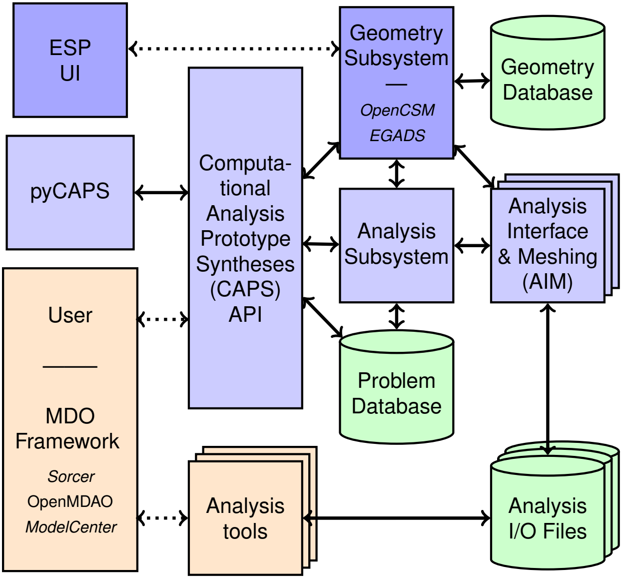
ESP is:

- a parametric geometry creation and manipulation system designed to **fully support** the analysis and design of aerospace vehicles (**aCAD**)
- a stand-alone system for the development of models
- can be embedded into other software systems to support their geometric and process needs

ESP is not:

- a full-featured mechanical computer-aided design (**mCAD**) system
- a system to be used for creating “drawings”
- an MDAO Framework

Background – Engineering Sketch Pad (ESP)



EGADS Overview

The Engineering Geometry Aircraft Design System (EGADS) is an open-source geometry interface to OpenCASCADE

- Reduces OpenCASCADE's 17,000 methods to less than 100 calls
  - Supports C, C++ & FORTRAN
- Provides *bottom-up* and/or *top-down* construction
- Full suite of geometric primitives
  - curve: line, circle, ellipse, parabola, hyperbola, offset, bezier, BSpline (including NURBS)
  - surface: plane, spherical, conical, cylindrical, toroidal, revolution, extrusion, offset, bezier, BSpline (including NURBS)
- Solid creation and Boolean operations (*top-down*)
- Provides persistent user-defined attributes on topological entities
- Adjustable tessellator (*vs* a surface mesher) with support for finite-differencing in the calculation of parametric sensitivities

EGADS Overview

System Support (now 64 bit ony)

- Mac OSX with **gcc**, **clang**, **ifort** and/or **gfortran**
- LINUX with **gcc**, **ifort** and/or **gfortran**
- Windows with Microsoft Visual Studio C++ and **ifort**
- No globals (but not thread-safe due to OpenCASCADE)
- Various levels of output (0-none, through 3-debug)
- Written in C and C++

EGADS Objects (**ego**s)

- Pointer to a C structure – allows for an *Object-based* API
- Treated as “blind” pointers (i.e., not meant to be dereferenced)
- **ego**s are INTEGER*8 variables in FORTRAN

Geometric Expressions

surface

- 3D surfaces of 2 parameters $[u, v]$
- Can have infinite extent
- Sample types: Plane, Spherical, Cylindrical, Revolution, Toriodal, Trimmed, Bezier, BSpline/NURBS, Offset, Conical, Extrusion
- All types abstracted to $[x, y, z] = f(u, v)$

pcurve – Parameter Space Curves

- 2D curves in the Parametric space $[u, v]$ of a surface
- Sample types: Line, Circle, Ellipse, Parabola, Hyperbola, Trimmed, Bezier, BSpline/NURBS, Offset
- All types abstracted to $[u, v] = h(t)$

Geometric Expressions

curve

- 3D curve – single running parameter (t)
- Can have infinite extent
- Sample types: Line, Circle, Ellipse, Parabola, Hyperbola, Trimmed, Bezier, BSpline/NURBS, Offset
- All types abstracted to $[x, y, z] = g(t)$

point

- 3D point in space – no parameter

Geometric Expressions

Geometric Entity Parametrization

- Arc-length based: most analytic forms
  Linear use of parameter space results in arc-length spacing
  Great for meshing!
- BSpline/NURBS – based on knot sequence
  - Has an order (polynomial degree)
  - Constructed by *approximation* uses piecewise arc-length spacing
    Otherwise the spacing may need to be carefully examined!
  - *Multiplicity of knots* reduces *Continuity* for each added knot

Continuity – C^x

The number of available derivatives

- Degenerate points (Poles of a sphere, Apex of a cone)
- BSpline/NURBS
  Can be C^0 in the interior (*multiplicity of knots* = degree)

BRep Topology

Topology	**Body Type**	**Geometry**	**Function**
container	*Solid*		
Shell	*Sheet*		
Face		**surface**	$(x, y, z) = \mathbf{f}(u, v)$
Loop	*Wire*		
Edge		**curve**	$(x, y, z) = \mathbf{g}(t)$
(CoEdge *)		**pcurve**	$(u, v) = \mathbf{h}(t)$
Node	*Point*	**point**	

- A *Solid* Body is closed and manifold
- A *Sheet* Body is either open and/or nonmanifold
- A *Wire* Body has no Faces and may be open

* A CoEdge is only required for a Loop that *bounds* a Face and is matched to an Edge (with the same parameterization)

BRep Topology – Terminology

Clashes with traditional Gridding nomenclature

Topology	**Alias**	**Mesh Term**	**My Term**
Shell			
Face		facet of a 3D element	facet
Loop	Wire Contour		
Edge		2D element side 3D element edge	side ?
CoEdge	Fin		
Node	Vertex		vertex

Note that I will further distinguish BRep terms by capitalization

This is a bit of a mess!

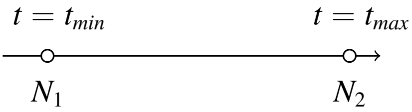
BRep Topology

Node

- Contains a point $[x, y, z]$

CoEdge/Edge

- Has a curve (2D or 3D)
- Has a t range (t_{min} to t_{max}, where $t_{min} < t_{max}$)
  Note: The positive orientation is going from t_{min} to t_{max}
- Has a Node for t_{min} and for t_{max} – can be the same Node

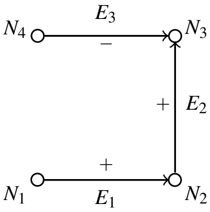


BRep Topology

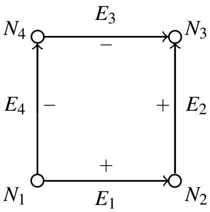
Loop – without a reference surface

Free standing connected Edges that will **not** be used to *bound* a Face

- An ordered collection of Edges with associated senses that define the connected Loop
- Can be open or closed (comes back on itself)



Open: $+E_1$ $+E_2$ $-E_3$



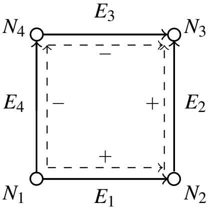
Closed: $+E_1$ $+E_2$ $-E_3$ $-E_4$

BRep Topology

Loop – with a reference surface

Connected Edges that *sit* on a surface

- An ordered collection of Edges & senses with corresponding CoEdges that define the $[u, v]$ surface trimming
- No intraEdge *self* intersections
- Types: open or closed (comes back on itself)



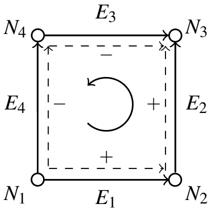
dotted lines indicate CoEdges/pcurves

BRep Topology

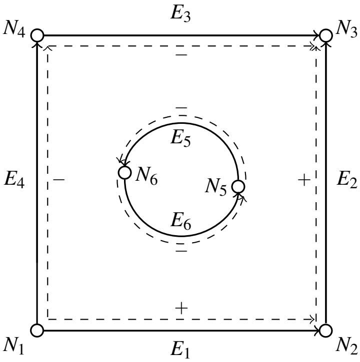
Face

- A surface bounded by one or more closed Loops
- *Usually* only one outer Loop and any number of inner Loops
- The Loops reference surface must match that of the Face
- The orientation of the Face based on surface's $U \otimes V$
    - An outer Loop traverses the Face in a right-handed manner
    - Inner Loops trim the Face in a left-handed manner
    - *Material* is to the left of the Edges going around the Loops

surface normal
is out of the page

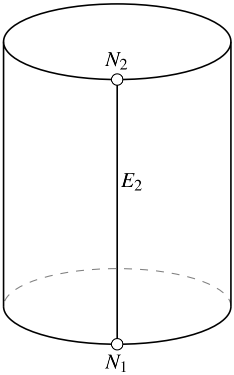


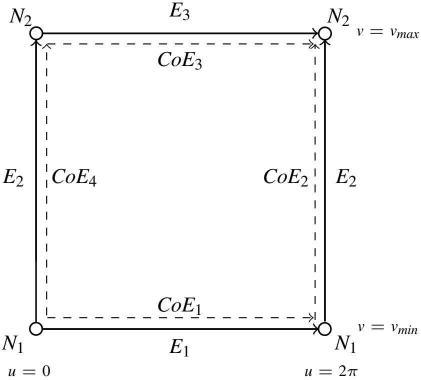
BRep Topology – Face Examples



- Outer Loop – right handed/counterclockwise: $+E_1$ $+E_2$ $-E_3$ $-E_4$
- Inner Loop – left handed/clockwise: $-E_5$ $-E_6$

BRep Topology – Face Examples



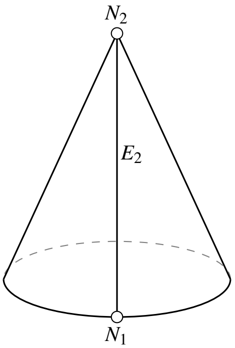


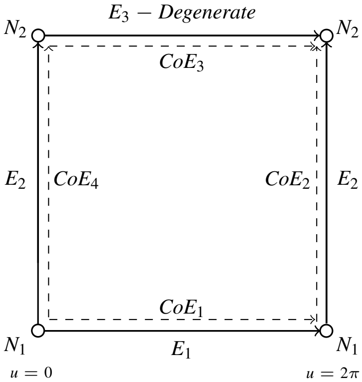
Unrolled periodic cylinder Face

Single Outer Loop – right handed/counterclockwise:

$+E_1$ $+E_2$ $-E_3$ $-E_2$

BRep Topology – Face Examples

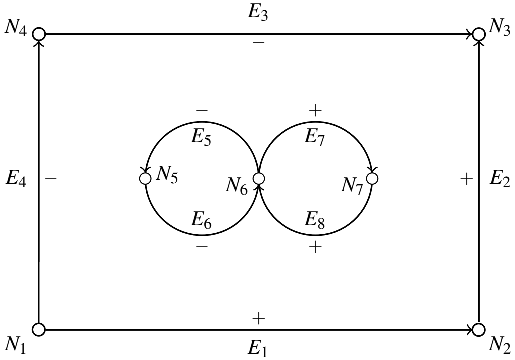




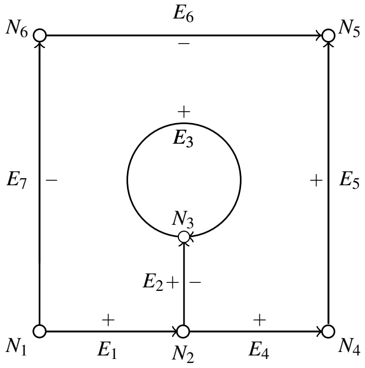
Unrolled Cone

E_3 and/or CoE_3 may not exist in the Topology

BRep Topology – Face Examples



- Outer Loop – right handed/counterclockwise: $+E_1$ $+E_2$ $-E_3$ $-E_4$
- Inner Loop #1 – left handed/clockwise: $-E_5$ $-E_6$
- Inner Loop #2 – left handed/clockwise: $+E_7$ $+E_8$



Single Outer Loop – right handed/counterclockwise:

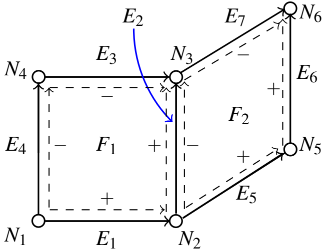
$+E_1$ $+E_2$ $+E_3$ $-E_2$ $+E_4$ $+E_5$ $-E_6$ $-E_7$

Note: CoEdge the same for both sides of E_2

BRep Topology

Shell

- A collection connected Faces with associated senses
- If sense is negative: may require considering the Loops reversed
- If closed: segregates regions of 3-Space



Face #1 Loop: $+E_1$ $+E_2$ $-E_3$ $-E_4$

Face #2 Loop: $+E_5$ $+E_6$ $-E_7$ $-E_2$

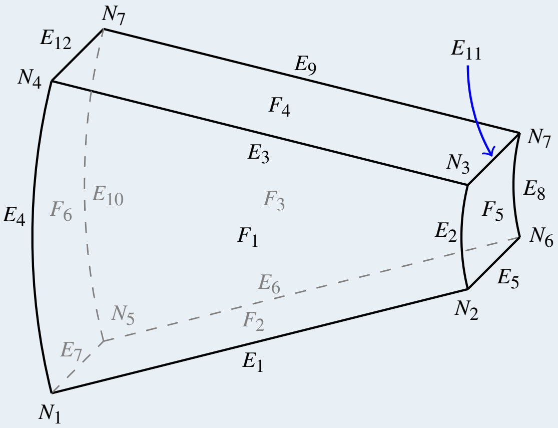
Manifold vs. Nonmanifold

The top level *container*

- Single BRep – *Bag of Topology*
  - Requires close examination of children when parsing the BRep
  - Can handle *floating*/superfluous entities
    i.e., entities may have children that do not *bound*
- Collection of (possibly connected) Bodies
  - Point Body
  - Wire Body
  - Sheet Body (open and/or nonmanifold)
  - Solid Body
    - A manifold collection of one or more closed Shells (with associated senses)
    - There may be only one outer Shell and any number of inner Shells
    - Edges (except Degenerate) are found exactly twice (sense = $\pm$)

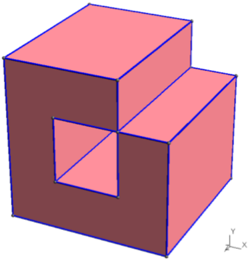
Manifold vs. Nonmanifold

Simple *Solid* Body example

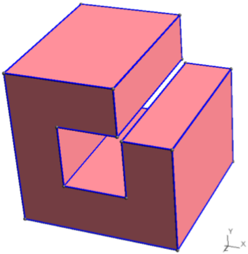


8 Nodes, 12 Edges, 24 CoEdges, 6 Loops and 6 Faces

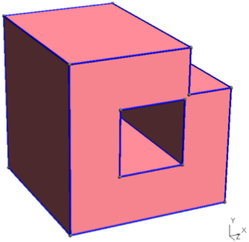
Manifold vs. Nonmanifold



nonmanifold



manifold



manifold

figure stolen from “An introduction to Geometrical Modelling and Mesh Generation: The Gmsh Companion” by Christophe Geuzaine, Emilie Marchandise & Jean-François Remacle – used without permission!

Parsing the BRep

Given a BRep

Top Down Queries – Given a topological entity:

- What is the *attached* geometry?
- What are the topological children?

Helper Functions

Allow for *jumping* up and down the topological hierarchy

- Who is the parent?
- Who are my neighbors (share children)?
- Skip generations:
  What are the Nodes on this Face?

The BRep & Tolerances

Why do we need Tolerancing?

BReps are not closed at machine precision

- *"Dirty CAD"* – Oxymoron especially for a *Solid*
- Due to surface/surface intersections
  No general analytic solution – point sampling & fitting
- Smoothness requirement

Tolerancing Models – Not transmitted in STEP files

- Absolute
- Relative – at the BRep level
- Relative – individual Entity (EGADS & OpenCASCADE)

Local tolerance can always be found by evaluations at bounds

The EGADS API

- Function names begin with "EG_" – or –
  "IG_" for the FORTRAN bindings
- Functions almost always return an integer *error code*
- *Object-based* – procedural, usually with the first argument an **ego**
- Signatures usually have the inputs first, then output argument(s)
- Some outputs may be pointers to lists of *things*
  `EG_free` needs to be used when marked as "freeable"
- **ego**s have:
  - *Owner*: Context, Body, or Model
  - Reference Objects (objects they depend upon)
- When a Body is made, all included Objects are copied – not referenced

The EGADS API – *Ownership*

Deleting Objects

- Use the function “EG_deleteObject” (or “ig_deleteobject”)
- The Object must be reference *free* – i.e. not used by another
  - Delete in the opposite order of creation
  - If in a Body, delete the Body (unless the Body is in a Model)
- “EG_deleteObject” on a Context does not delete the Context
  - Deletes all Objects in the Context that are not in a Body
  - Use “EG_close” to delete all objects in a Context (and the Context)

Another Rule

- A Body can only be in one Model
  - Copy the Body of interest, then include the copy in the new Model

Meshing Considerations

Point Queries – Evaluations

- Derivatives – Most all APIs provide 1^{st} and 2^{nd}
  - Curves: at $t \Rightarrow X, \frac{dX}{dt}$ and $\frac{d^2X}{dt^2}$

    tangency & curvature*
  - Surfaces: at $[u, v] \Rightarrow X, \frac{dX}{du}, \frac{dX}{dv}, \frac{d^2X}{du^2}, \frac{d^2X}{dv^2}$ and $\frac{d^2X}{dudv}$

    normal $(\frac{dX}{du} \otimes \frac{dX}{dv})^{\dagger}$ & curvature*
- Continuity – Derivative unavailable for less than C^2
  - Degenerate points (Poles of a sphere, Apex of a cone)
  - BSpline/NURBS with *multiplicity of knots*
  - Appropriate derivatives are returned as 0.0

* Note: Some APIs have functions that return Radius of Curvature(s) and the associated direction(s)

† Note: The Face normal may be opposite that of the surface

Meshing Considerations

Point Queries – Inverse Evaluations

curve/Edge: $t = \mathbf{g}^{-1}(x, y, z)$
surface/Face: $[u, v] = \mathbf{f}^{-1}(x, y, z)$

Can be accomplished with 1^{st} and 2^{nd} derivatives from Evaluation

- Optimization (Newton-Raphson)
  - Minimize distance to requested position – needs start location not projection in a particular direction
  - stopping criteria – only as accurate as this ϵ
  - needs clear line-of-site to target (wing near TE on wrong side)
  - can get stuck in local minima
  - periodicity – result may need to be adjusted by the period
- Costly when robust (requires many seed locations)
- Some APIs can limit to Edge/Face Bounds

Best to be avoided (if possible)

Meshing Considerations

Point Queries – Contained within Predicates

Answers the question: Is the specified location in this entity?

Useful meshing queries:

- Is this $[u, v]$ in the Face?
  - Is this $[u, v]$ in the Face's valid parametric box?
  - Is this $[u, v]$ trimmed away or in a hole?
- Is this X contained within the *Solid*?
  - Performed by ray-casting and counting crossings
  - Can be expensive
- Ambiguous (within the tolerance and) near bounds

It may be better to answer these queries in a discrete setting

Meshing Considerations

Surface Degeneracies

If the surface meshing algorithm has smoothness assumptions, then knowledge of degenerate locations is critical!

- Degenerate points (Poles of a sphere, Apex of a cone)
  - You can depend on these points being Nodes
  - Geometry Kernels with Degenerate Edges mark these locations
  - Zero derivatives are returned for an Evaluation at these points
- BSpline/NURBS with *multiplicity of knots* at C^1 or C^0
  - Much harder to deal with!
  - Will probably also have Edge curves with *kinks*
  - Some Geometry Kernels have functions that split Faces/Edges at C^1 and/or C^0 locations, which does not change the geometry but places all *kinks* at the topological bounds
  - If you are constructing the model – **DON'T DO THIS!**

Watertight BRep Meshing

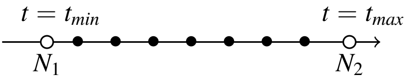
Dimensional Strategy

First mesh all of the Edges and then the Faces:

- Set the boundary vertices from lower in the topological hierarchy
- Mesh the interior
- Assume that C^0 locations are at bounds

Discretizing Edges

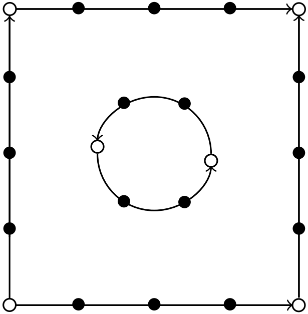
- Set the start and end Node positions (can be the same)
- Evaluate at t_{min} & t_{max} and compare to Node positions gets local (endpoint) tolerances
- Use a scheme to distribute vertices along the Edge (t or other) do not add vertices at the ends within the Node's local tolerance



Watertight BRep Meshing

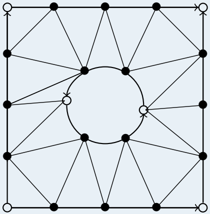
Discretizing Faces

- Foreach Loop that bounds the Face:
  - Collect the Edge discretization *tail to head-1* or *head to tail-1* based on the Edge's sense in the Loop
  - Use the Edge vertex t value and the CoEdge/pcurve to get $[u, v]$
  - Makes a 2D closed set of line segments & 3D bounds for the Face



Discretizing Faces – continued

- Any closed 2D region can be triangulated
  No additional vertices are needed



- Use a scheme to enhance the initial triangulation
  - Query in X and insert in $[u, v]$
  - When a triangle side in on an Edge/Node care should be taken:
    Compare X with the surface/Face evaluation at $[u, v]$
    Do not add vertices within that local tolerance

Closing Remarks

- Comprehensive EGADS programming examples can be found at: $ESP_ROOT/doc/EGADS/Tutorial
- Other EGADS examples can be found in the ESP Distribution: $ESP_ROOT/src/EGADS/examples
- OpenCSM
  - The parametric *engine* on top of EGADS (like SolidWorks is to Parasolid)
  - Basically *Top Down* but can support *Bottom Up* builds by UDPs
  - Tire UDP – $ESP_ROOT/data/training/session8/udpTire.c
- CAPS AIMs are also EGADS applets
- A note on OpenCASCADE:
  Though EGADS was originally a *thin* layer over OpenCASCADE many of the methods that did not work well have been replaced. Next on the list – the SBOs!

Closing Remarks – Future Directions

PAGODA

Develop a distributed/threaded geometry system to support solver meshing, adaptation, and sensitivities for analysis and design

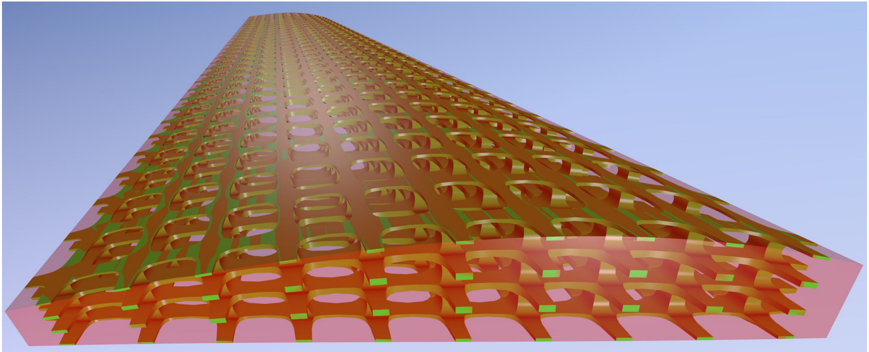
- `EGADSlite` for HPC environments & threaded SBOs

DARPA's TRADES Program – Jan Vandenbrande, PM

The TRAnsformative DESign program aims to advance the foundational mathematics and computational tools required to generate and manage the enormous complexity of design

- *Augmented Design Through Analysis and Visualization – Facilitating Better Designs and Enhanced Designers*
- *Design Responding to Engineering Analysis in support of Manufacturing – DREAM*
  - Fully couple conceptual optimization to the following phases
  - Embrace volumetric representations (VReps) in design

Closing Remarks



Software available at:

http://acdl.mit.edu/ESP