The Deployable Structures Library

Prepared for:

The American Modelica Conference 2018

Prepared by:

Cory Rupp, Ph.D.
cory.rupp@ata-e.com

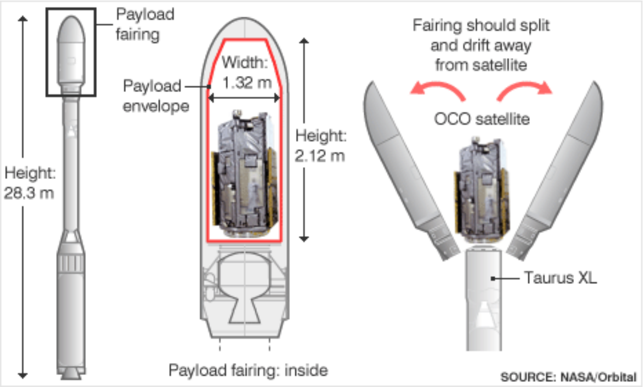
Date:

October 10, 2018

1597 Cole Boulevard, Suite 375, Lakewood, CO 80401
13290 Evening Creek Drive, Suite 250, San Diego CA 92128
(858) 480-2000
www.ata-e.com
ata-engineering
@ATAEngineering

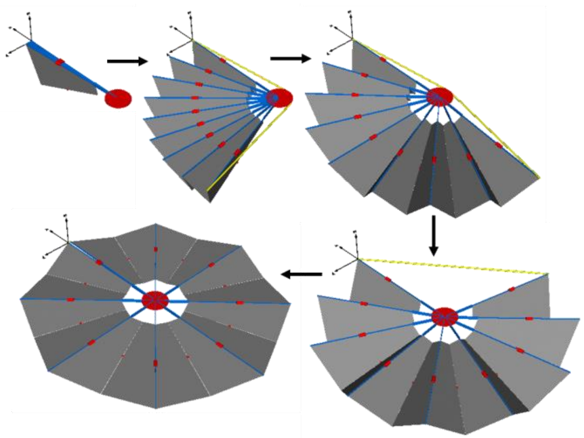
Deployable Structures Are Difficult To Model

- Nearly all spacecraft have deployable structures
  - Solar arrays, antennas, sensors
  - Volume in launch vehicle fairing is limited
- Deployment while in orbit is the #1 risk
  - If it doesn't deploy, nothing else matters
- Most of these structures have unique deployment mechanisms that are difficult to model
  - Often impossible with standard modeling tools
  - Need for adaptable modeling tool
- Flexible multi-body dynamics is also necessary to provide design guidance
  - Modal analysis, structural deflections



Modelica Enables Modeling Deployable Structures in Early Design Phases

- Immature designs require flexibility in component selection, sizing, and function
  - Greatly aided by Modelica parameterization
- Some problems (e.g., solar array scaling) require performing topological changes
  - Change the number of panels or sections
  - Also parameterizable in Modelica (with recompilation)
- Unique mechanisms can be modeled
  - Deployable boom, lanyard on a spool, latches, etc.



UltraFlex/MegaFlex (NGIS)

ISS solar array

ISS

ATA ENGINEERING, INC.

Figures courtesy of NASA

The Deployable Structures Library (DeployStructLib)

- Compliments the MultiBody library of the Modelica Standard Library
- Contains specialized modeling blocks often used in deployable structures:
  - Variable length flexible beam models a deploying boom
  - Tension-only/compression-only springs
  - Release and stop/lock mechanisms
  - Flexible cloth modeling capability
- Uses a top-level property definition workflow typically of other engineering software
- Usage examples included

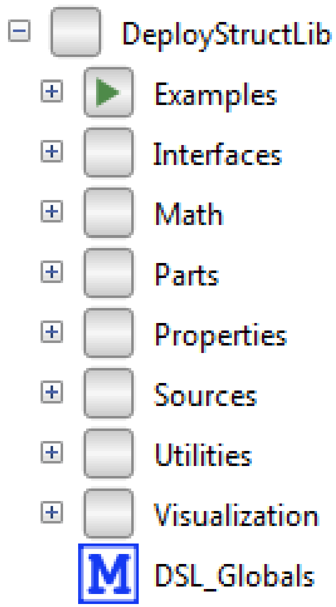
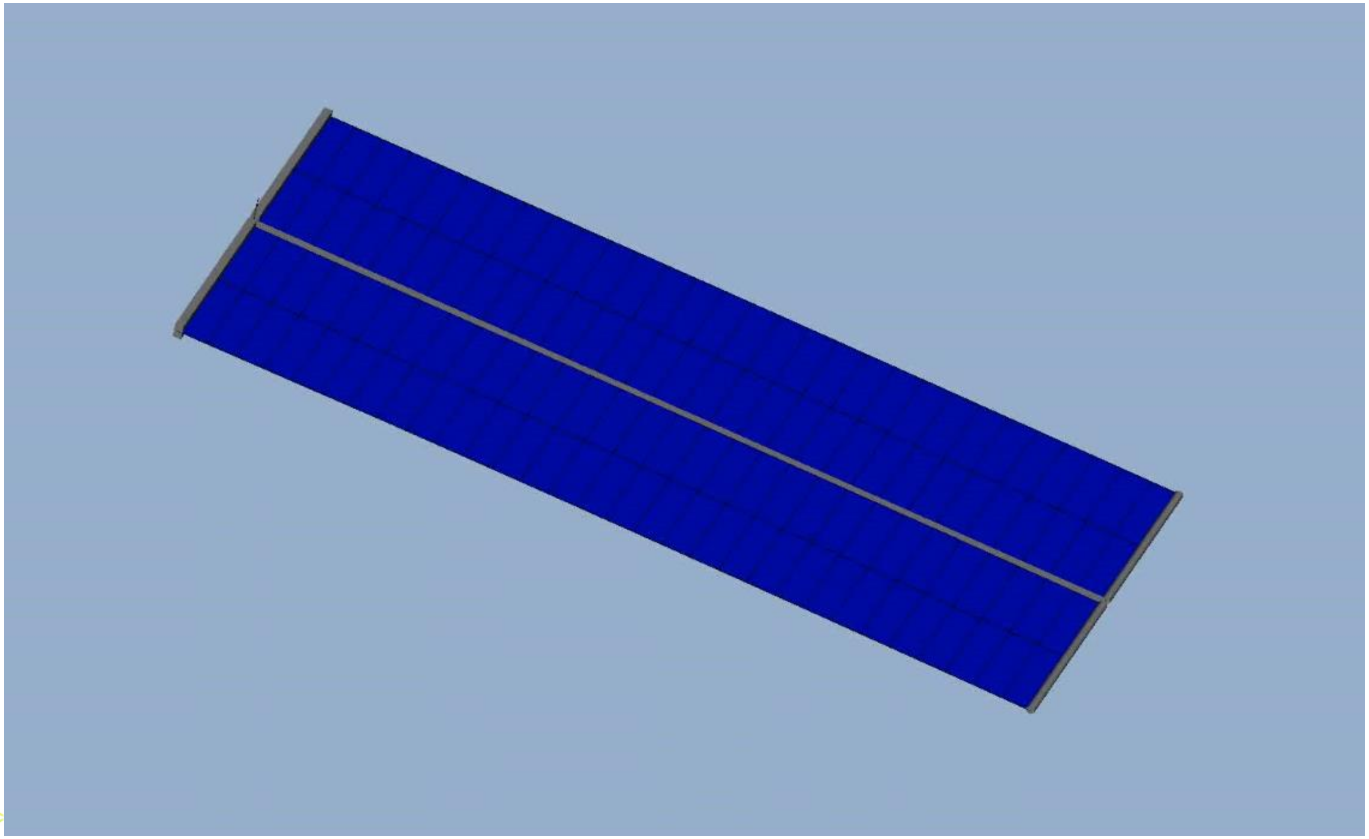


Figure courtesy of NASA

ATA ENGINEERING, INC.

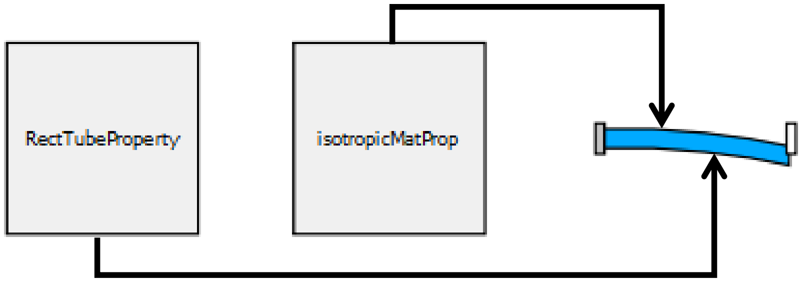
DeployStructLib Development Driven by Difficult-to-Model Structures

- During solar array deployment, several analysis challenges exist
  - Deploying mast changes stiffness and inertia continuously over time
  - Solar blanket must unfold
  - Blanket is tensioned at the end of deployment, adding stress stiffening



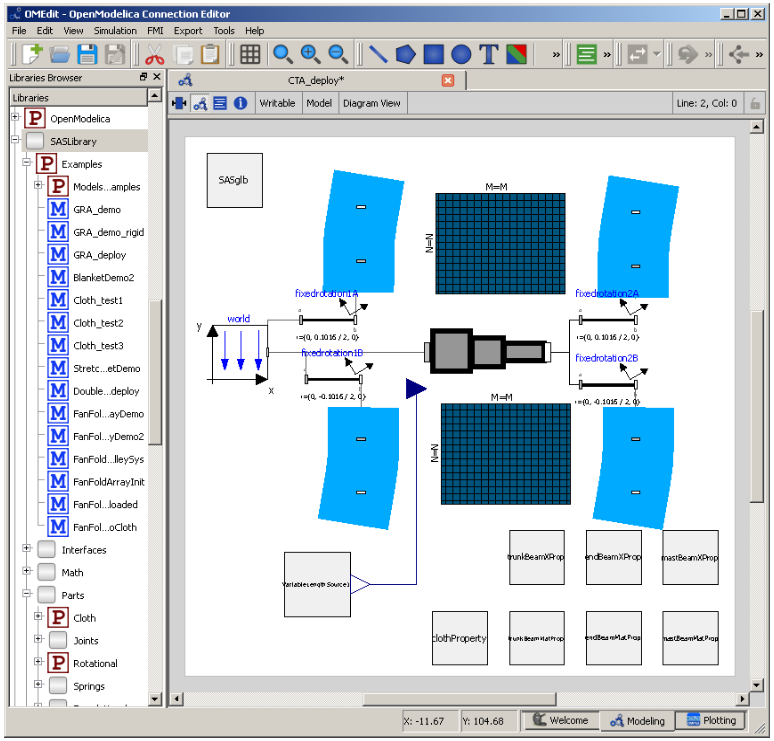
Property Definition Workflow

- Most structural engineering software (especially finite elements) define material, cross-sectional, and other properties at the top level
  - These are merely referenced by the entities that use them
- The DeployStructLib implements a similar scheme via Modelica record blocks
- Currently three types
  - Material properties
  - Beam cross-sectional properties (standard and EAGJ formulations)
  - Cloth properties (equivalent to an FE shell)
- Passed into modeling blocks as parameters



DeployStructLib.Parts

- Beams
  - Parameterized to model either a rigid or flexible Euler beam
    - Makes model building and debugging significantly easier
  - Flexible beam follows formulation of Schiavo *et al* (2006), with updates for property definition
  - Rigid beam follows the MSL BodyShape block, again with property updates



- VariableLengthBeam
  - Flexible beam that updates its stiffness as mass as it changes length
  - Geared toward slow-moving space structures (makes quasi-static assumptions)

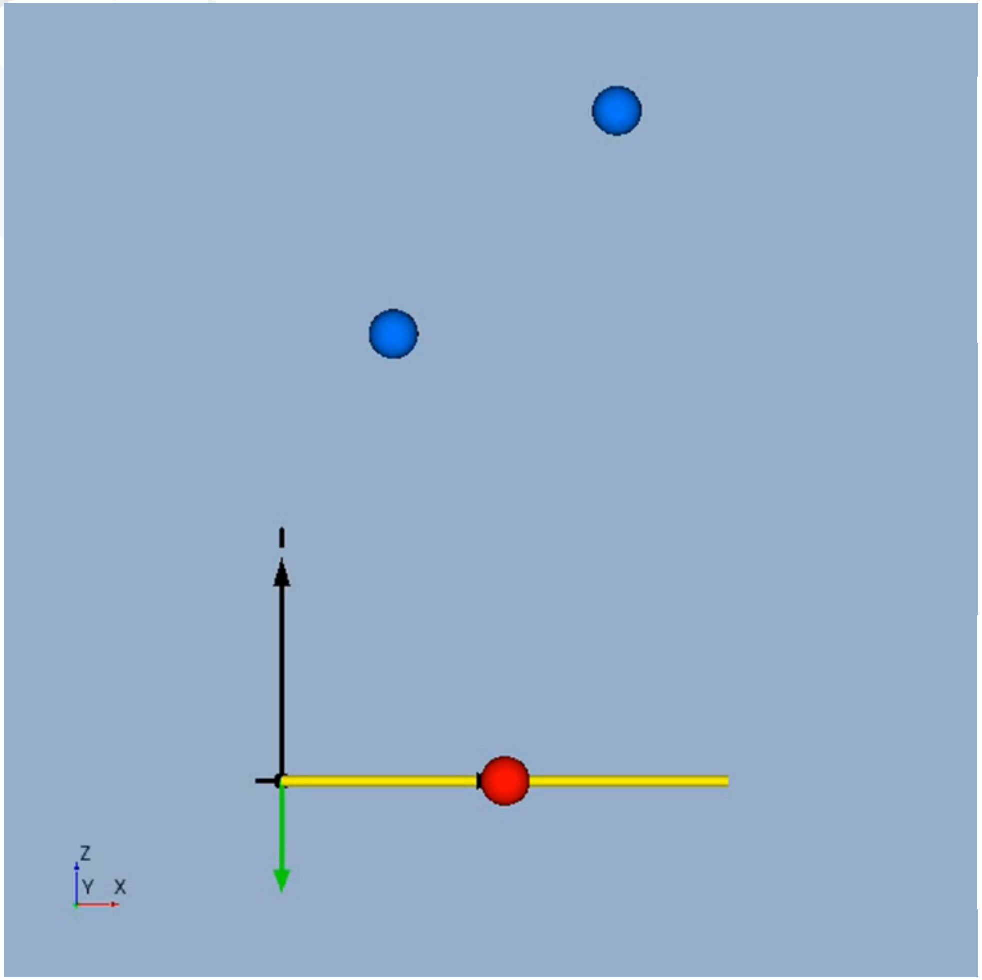
ATA ENGINEERING, INC.

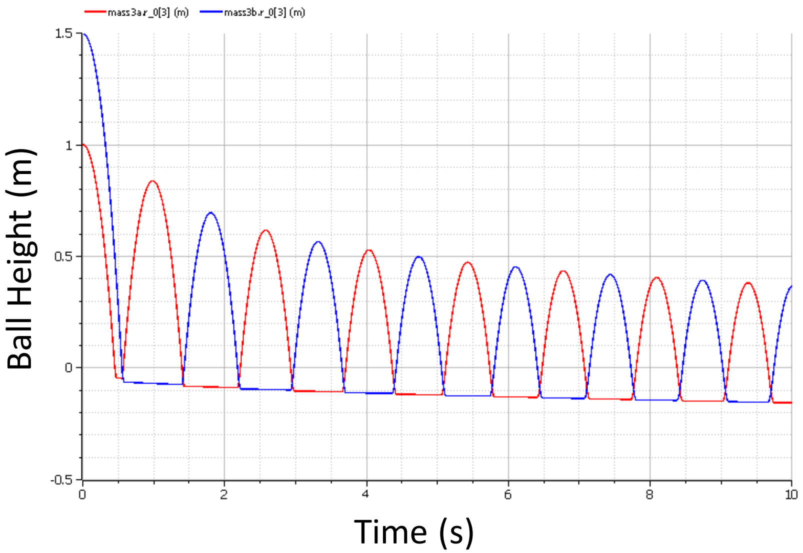
DeployStructLib.Parts.Springs

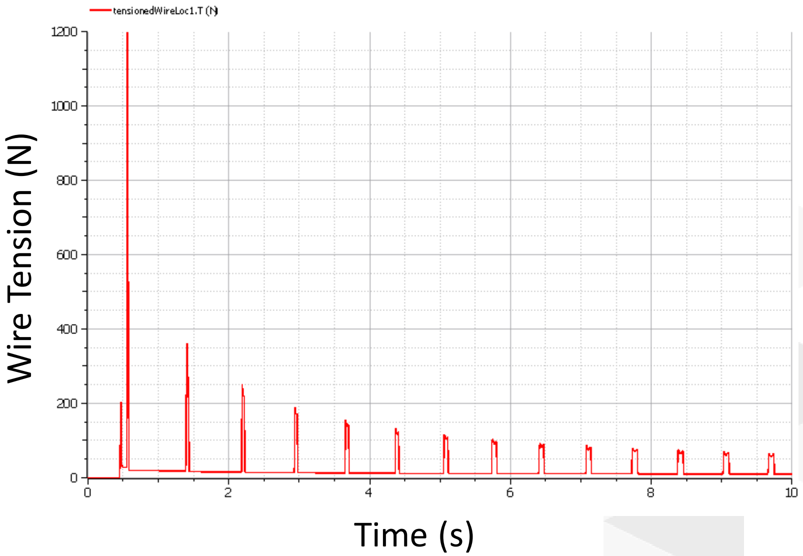
- Locks, stops, and barriers
  - Prevent MultiBody joints from moving too far
  - Translational and rotational versions
  - Nonlinear springs using Modelica semiLinear function
- Tension-only, compression-only, and release mechanisms
  - MultiBody models to simulate straps, wires, kick-off springs, contact, etc.
- Tensioned wire
  - Not in the library yet, but soon
  - Supports spooling and multiple interfaces (i.e., routing)
  - Example: Bouncing balls on a wire

Bouncing Balls on a Wire

Wire connected to damped spool gradually goes slack







A Fix for Kinematic Loops: Weak Joints

- Deployable structures often use mechanisms to create mechanical advantage, typically resulting in kinematic loops
- Often difficult to identify loops *a priori*
  - Especially for non-power users of Modelica
  - Even harder to set up problem properly with cut joints
- DeployStructLib introduces a set of weak joints to selectively break kinematic loops
  - Constraint equations are written in weak form:

```
r_rel_a = Frames.resolve2(frame_a.R, frame_b.r_0 - frame_a.r_0);
frame_b.f = -Frames.resolve2(R_rel, -c_constraint * r_rel_a);
```

  - Rather than strong form:

```
frame_b.r_0 = frame_a.r_0;
frame_b.f = -Frames.resolve1(R_rel, frame_a.f);
```

  - It would be better if Modelica had a way to force an equation to be in residue form
    - Similar to the equalityConstraint function
- Not ideal, but it saves headaches by getting the model running

ATA ENGINEERING, INC.

Modeling Solar Blankets with the Cloth Block

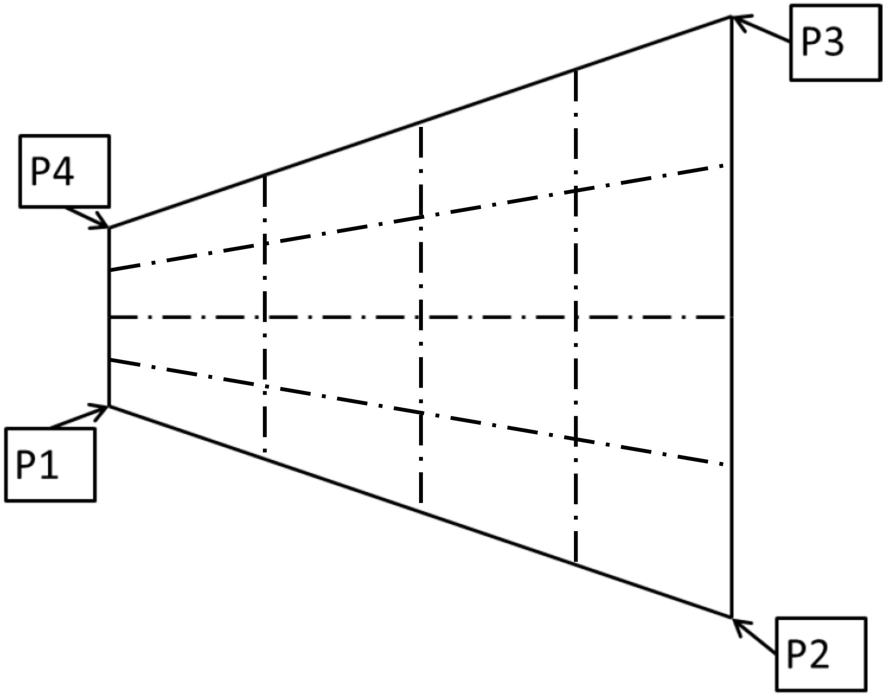
- A solar blanket consists of solar cells attached to a thin membrane that acts like a heavy fabric when deployed
- Requires solving a geometrically nonlinear problem
- DeployStructLib implements a new finite element formulation to efficiently model such structures
  - Geometrically nonlinear without updates to the stiffness matrix
  - Mass modeled with lumped masses
  - See paper for derivation
- Uses a new "Location" mechanical connector:

```
connector Location "Location of the component with one cut-force"
SI.Position r_0[3] "Position vector from world frame to the
  connector frame origin, resolved in world frame";
flow SI.Force f[3] "Cut-force resolved in world frame";
end Location;
```

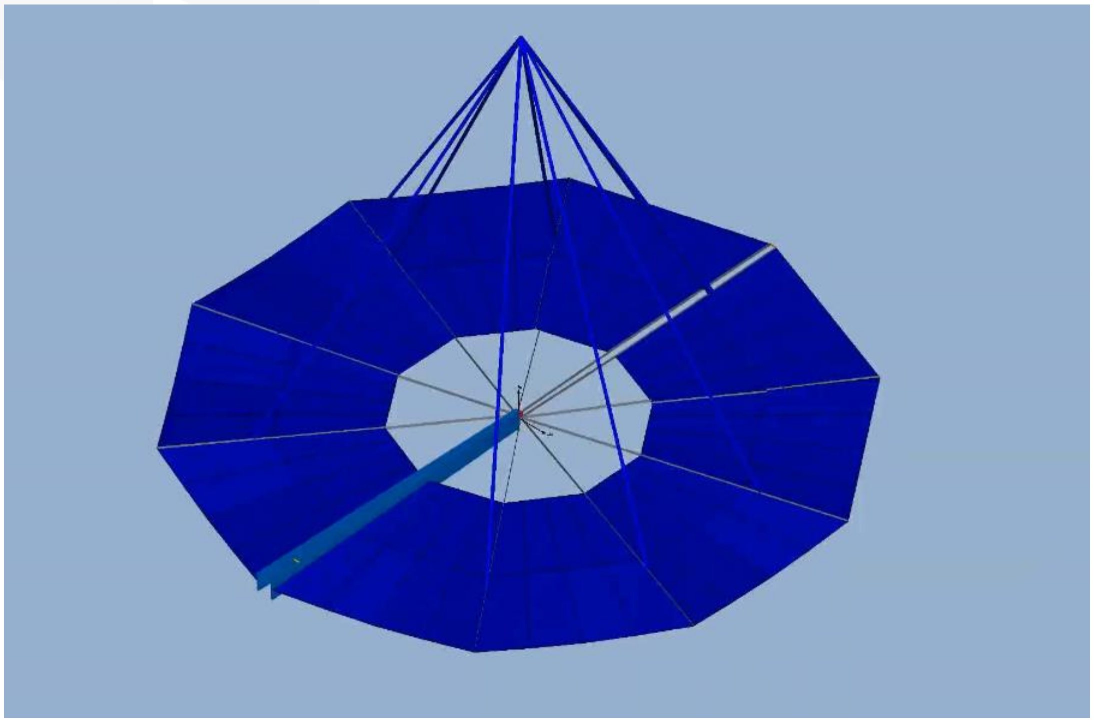
ATA ENGINEERING, INC.

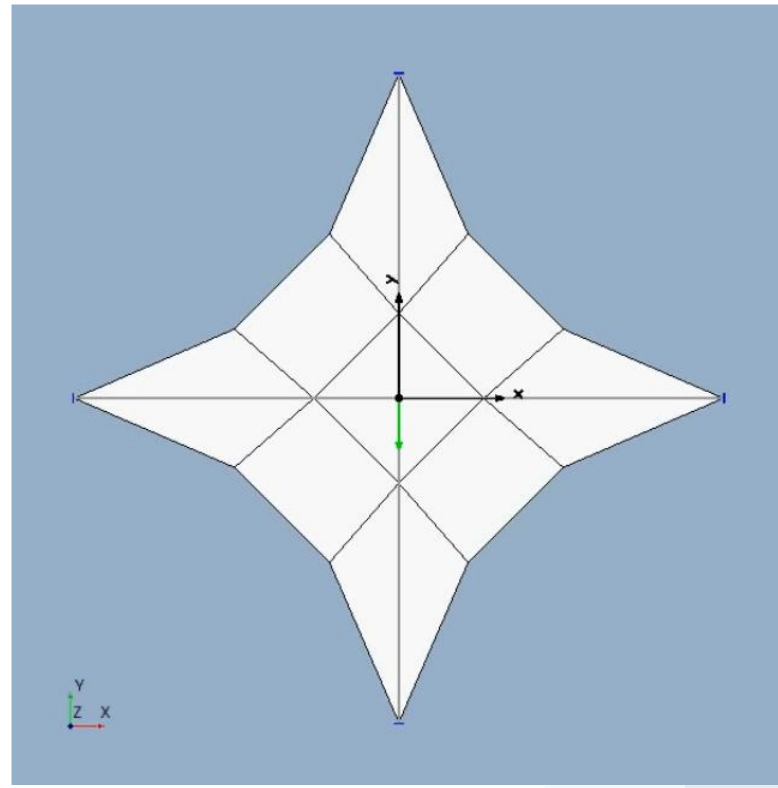
Cloth Modeling

- Cloth undeformed shape defined by four parameter point locations
- Initial location also defined by four points
  - Options for folding patterns and discretization
- Initialization performed via pre-compiled "C" code functions
  - Sets stiffness matrix and mass values as parameters
  - No reason to have Modelica compile each time



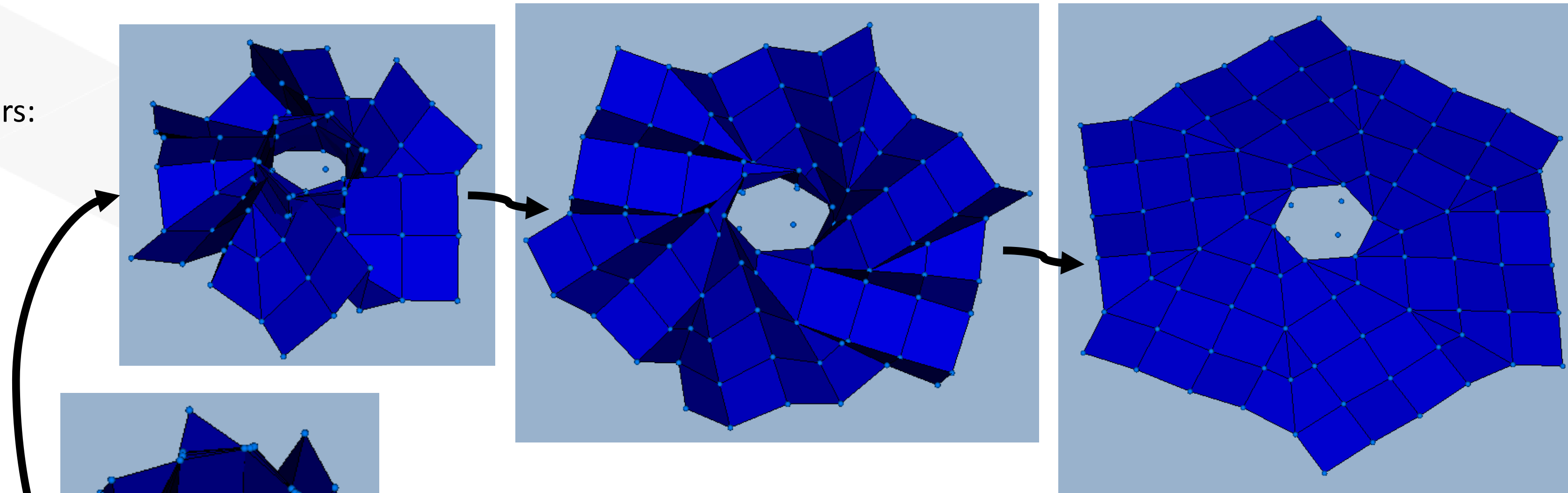
MegaFlex Solar Array and Solar Sail Deployment Animations

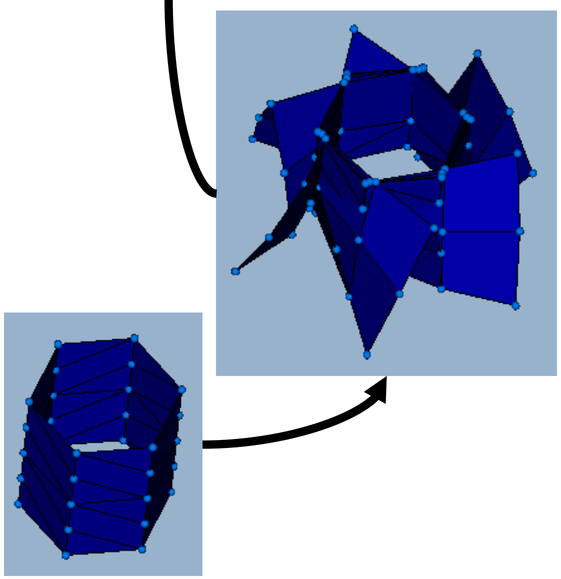




Origami Solar Array Deployment

Origami parameters:
M=6
H=2
R=2

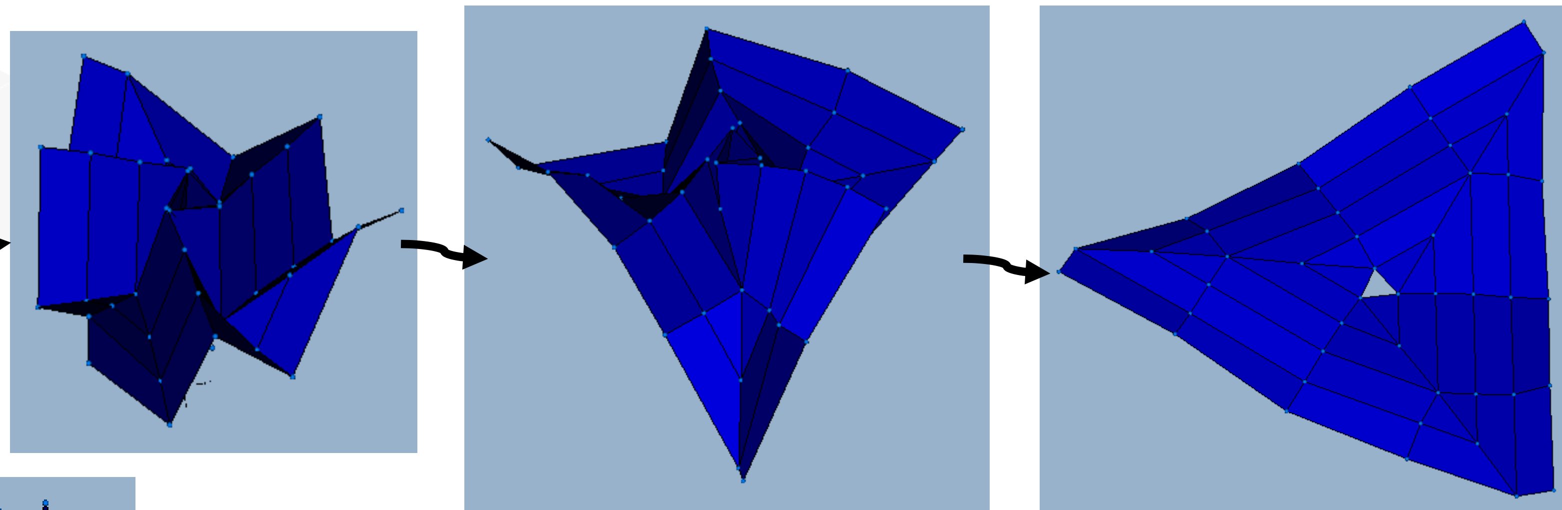


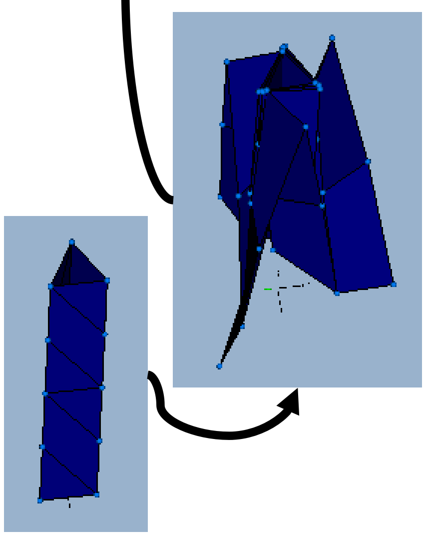


- Fully parameterized model allows for automated changes to material, structural, dimensional, and topological properties
  - Deployment simulation fixed at center, tip forces pull open

Topologically Inconsistent Updates Analyzed With the Same Model

Origami parameters:
M=3
H=2
R=2

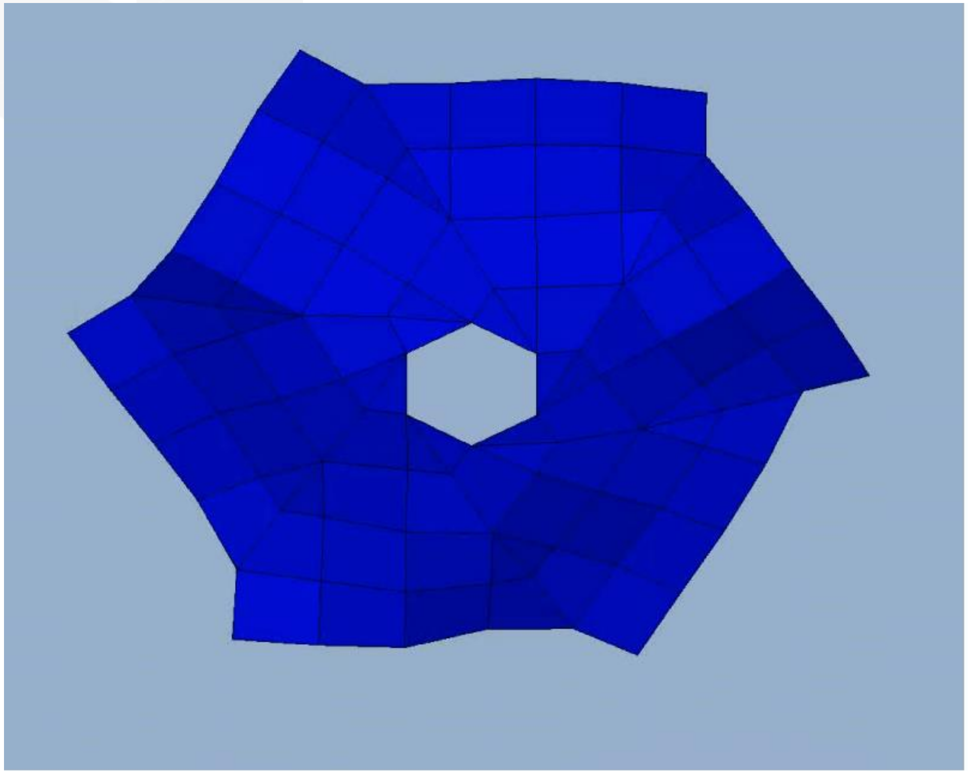




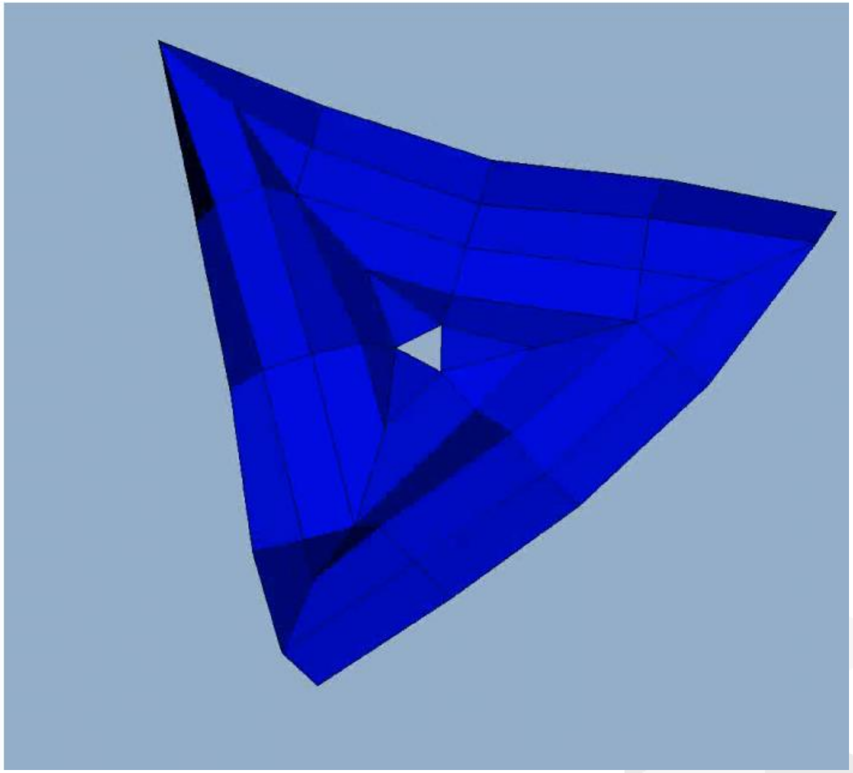
- Exact same model as previous example
- Only origami parameter M changed
  - Results in different origami layout and different structural topology

Origami Array Deployment – One Model, Many Designs

Origami parameters:
M=6, H=2, R=2



Origami parameters:
M=3, H=2, R=2



More info on this origami solar array design can be found in: Zirbel *et al*, *Journal of Mechanical Design*, 135, 2013.

Conclusions

- Available on GitHub
  - https://github.com/ATAEngineering/DeployStructLib
- Developed using OpenModelica
  - Help ensuring compatibility with other compilers would be much appreciated
  - Bug reports and suggestions are always welcome

- Work supported through a NASA Phase SBIR (NASA Langley)
  - Technical monitors: Geoff Rose and Richard Pappa
  - Contract: NNX14CL07C

ATA ENGINEERING, INC.

Contact Us

13290 Evening Creek Drive
Suite 250, San Diego, CA 92128

(858) 480-2000

info@ata-e.com

www.ata-e.com
www.ata-plmsoftware.com

@ATAEngineering

ata-engineering

ATA ENGINEERING, INC.