Rodymol:
Rotorcraft Dynamics Modeling Library
A Flexible Multi-Body Dynamics Modeling Toolkit

Benjamin Silbaugh[1]

[1] **Graduate Research Assistant**
silbaugh@umd.edu
Alfred Gessow Rotorcraft Center
Department of Aerospace Engineering
University of Maryland, College Park

October 10, 2012

UNIVERSITY OF
MARYLAND

What is Multibody dynamics?

- A generic methodology for solving equations of motion for an arbitrary *system*
- *Systems* are usually described by a set of primitive components and constraints
- Formulation can be characterized as "topology independent"
- Most generic approach is to numerically solve system as DAE
- Can eliminate constraints in some Holonomic systems and solve as ODE

Why should we care?

- Eliminates need to derive specialized EOM's for each system of interest
- Eliminates need to write specialized software for each system
- Exact treatment of finite rotations is easy
- Facilitates modular software design
- Facilitates modular verification and validation
- Facilitates distributed development and testing

What is Rodymol?

Rodymol: **Ro**torcraft **Dy**namics **Mo**deling **L**ibrary

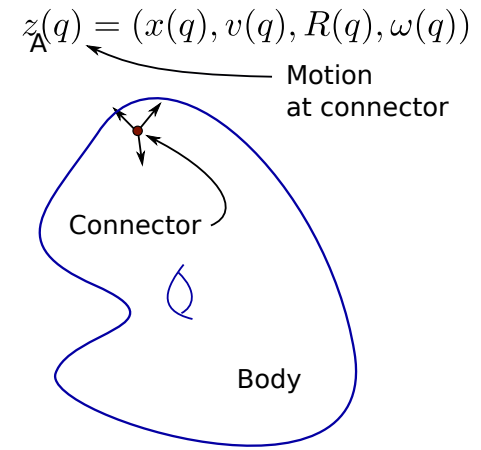
- A topology independent modeling methodology that allows both implicit and explicit treatment of constraints
- A high-level collection of software tools for modeling the evolution of rigid bodies, flexible bodies, joints, and parametric inputs
- A modern, highly organized, clearly written, C++ class library
- Can be used to create ad-hoc stand-alone programs
- Can be easily embedded into a computational framework
- Developed to support PhD research

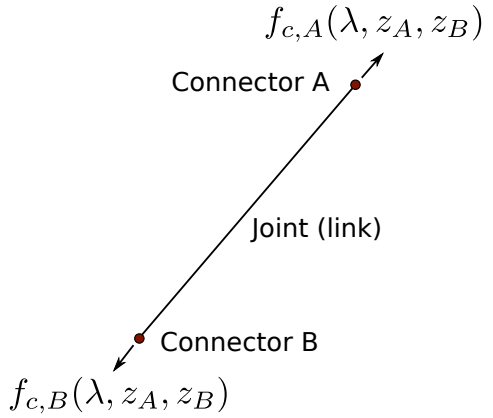
Outline

Global Equations of Motion

Multibody System EOM

- External forces:
  $f_e(t)$
- Internal + inertial force balance:
  $f(t) = \mathcal{F}(q(t), \dot{q}(t), \ddot{q}(t))$
- Constraint forces:
  $f_c(t) = \mathcal{F}_c(\lambda(t), z(t), t)$
- State of points/connectors on bodies:
  $z(t) = \mathcal{Z}(q(t))$
- Forces must sum to zero:
  $f(t) + f_c(t) + f_e(t) = 0$
- Auxiliary constraints must be satisfied:
  $\Phi(z(t), t) = 0$





Partitioning Scheme

Bodies (differential equations)

Internal + inertial force balance: $f = \{\mathcal{F}_i(q_i, \dot{q}_i, \ddot{q}_i, t)\}_{i=1}^{N}$

State vector: $q = \{q_i\}_{i=1}^{N}$

Motion at connectors: $z = \{\{z_p\}_{p=p_a(i)}^{p_b(i)}\}_{i=1}^{N}$

Force residual: $r = \{(f_i + \sum_{k \in K(i)} f_{c_k} + f_{e_i})\}_{i=1}^{N}$

Joints (algebraic equations)

Constraint forces: $f_c = \{\{f_{c_k}\}_{k=k_a(j)}^{k_b(j)}\}_{j=1}^{M}$

Constraint residual: $r_c = \{\phi_j(\{z_p\}_{p \in P(j)}, t)\}_{j=1}^{M}$

System Topology/Connectivity

Pointers linking constraint forces to bodies: $K(i) = \{k_\beta\}_{\beta=\beta_a(i)}^{\beta_b(i)}$

Pointers linking motion of connectors to joints: $P(j) = \{p_\alpha\}_{\alpha=\alpha_a(j)}^{\alpha_b(j)}$

Partitioning Revisited

Consider single partition

	ith Body Model	Joint Model
Vector of unknowns:	q	λ
Constitutive equation:	$f = \mathcal{F}(q, \dot{q}, \ddot{q})$	$\phi = \Phi(z, t)$
Coupling equation:	$z = \mathcal{Z}(q)$	$f_c = \mathcal{F}_c(\lambda, z)$
Residual equation:	$r = f + f_c + f_e$	$r_c = \phi$

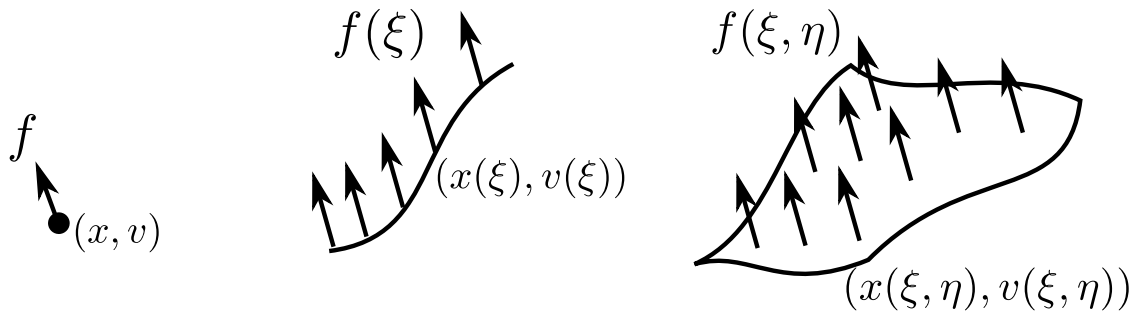
Hmm... bodies and joints have similar structure. Let's exploit this!

The Component Model

$$\text{Component Model} = \text{Body Model} \bigcup \text{Joint Model}$$

- Localize space and time discretization to each Body Model gives nonlinear algebraic system much like join model
- Flexible bodies may support either force or displacement BC's (body connectivity may be joint-like)
- May swap Joint Models with Body Models
- Define the Component Model as a generic representation of either a Body Model or a Joint Model

Connectivity Nodes



Point Node | Curve Node | Surface Node

- Component Models communicate by way of Connectivity Nodes
- Generalize Connectivity Node concept to include various topologies: point, curve, surface
- Multiple node topologies enables coupling between continua: flow fields, membranes, etc

Solution Algorithm

Initialize:

1. For each Component Model set initial conditions

For each time step:

1. For each Component Model:
   - 1.1 Set estimated state
   - 1.2 Compute value of displacement nodes
   - 1.3 Fetch displacement node data
   - 1.4 Compute value of force nodes
2. Compose global residual vector $r(q_{n+1})$
3. If converged stop
4. Compute correction Δq_{n+1} for global nonlinear algebraic system

Using Jacobian Free Newton Krylov method

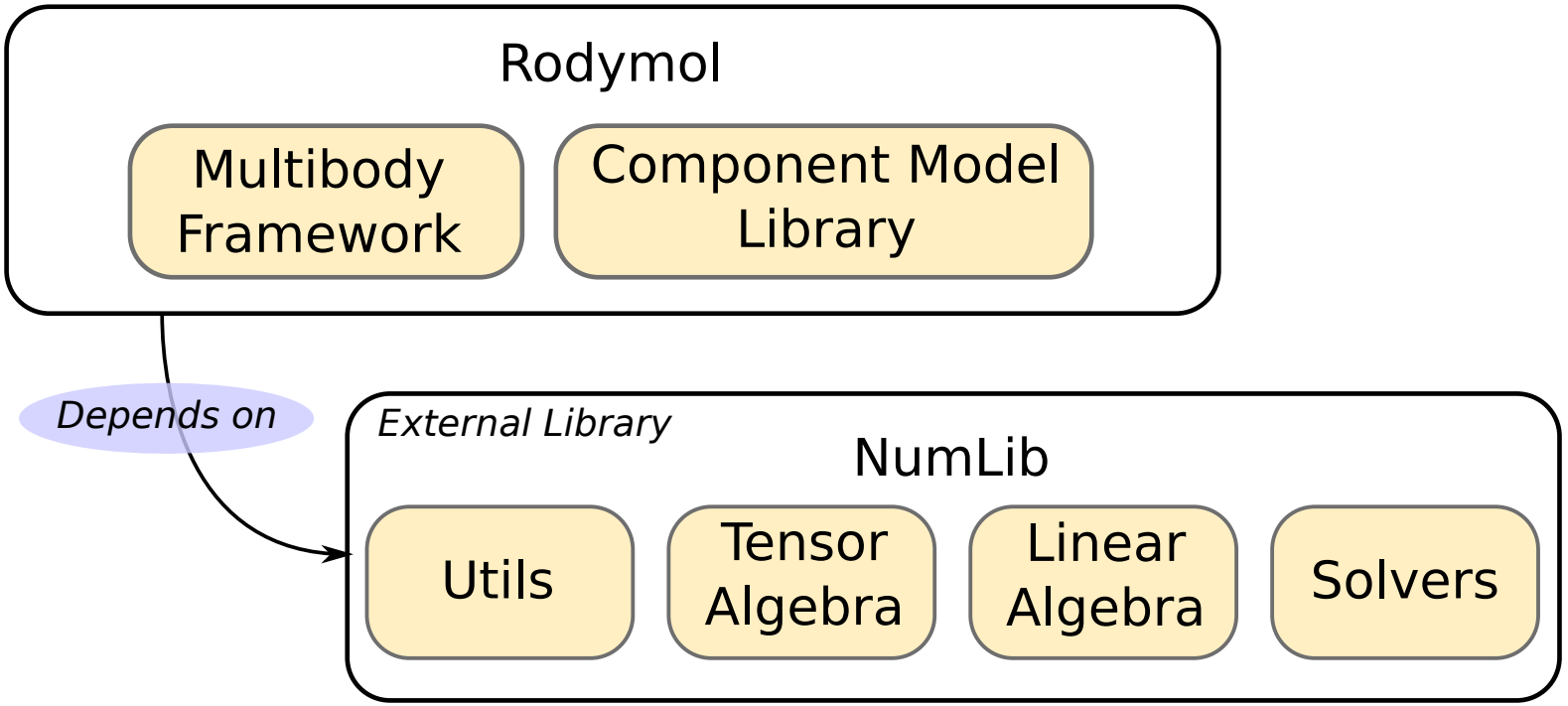
Outline

Overview

- Rodymol is a library of tools – not a stand-alone program
- Can explicitly link components together using client "glue code"
- Can use the Rodymol DSL interpretor to load configuration from disk

Effort required to create a stand-alone program is very small

Library Organization



- Rodymol is the "physics layer"
- Numlib is the "numerical methods layer"

World

Definition

The global parameters governing a simulation are collected into a singleton object called World.

Information provided by world:

- Current simulation time
- Time step size
- Gravity vector

World cannot be modified by system components

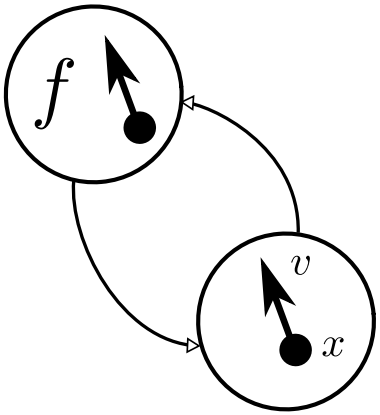
Connectivity Nodes

Definition

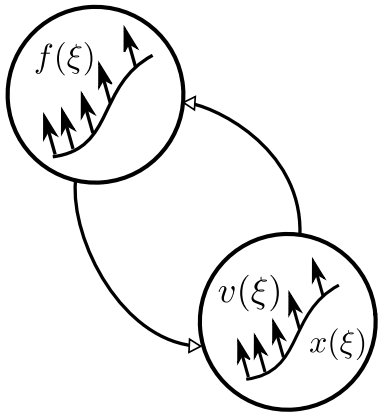
A Connectivity Node is an abstract representation of state or time dependent physical data associated with a particular set of material points.

- Material point sets include: points, curves, and surfaces
- Physical data includes: forces, moments, displacements, rotations, velocities and angular velocities.
- Organized into conjugate motion-load pairs

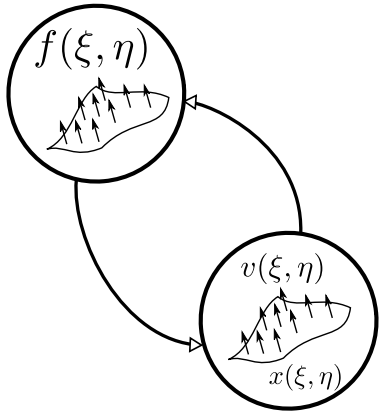
Point Node



Curve Node



Surface Node

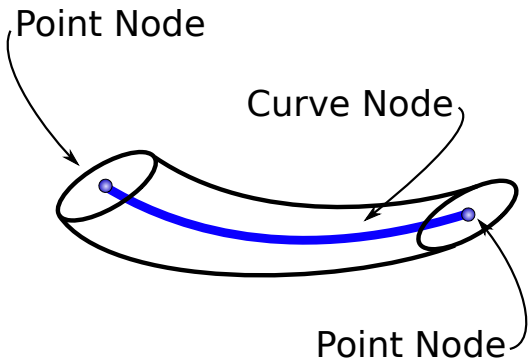


Component Model

Definition

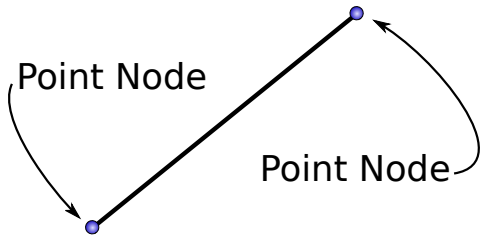
A Component Model is an implicit algebraic representation of a flexible body, rigid body, joint, or other algebraic constraint, combined with a set of connectivity nodes.

Examples:



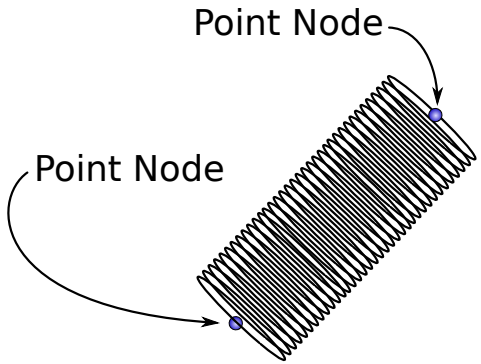
Flexible Beam

PDE



Massless Link

Algebraic Equation



Massless Spring

Force-Displacement map

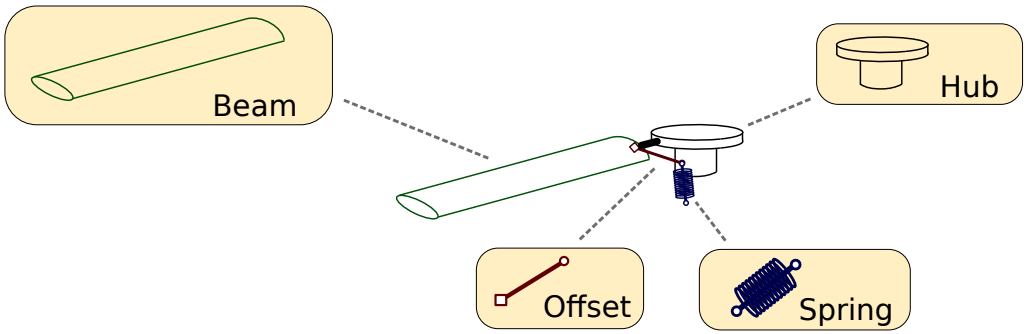
System Model

Definition

The System Model is a collection of Component Models, linked by way of their respective Connectivity Nodes, whose state and residual vectors are partitioned by the component models.

- Broadcasts time step messages to Component Models
- Partitions and forwards global state vector inputs to Component Models
- Collects and assembles Component Model residual vectors into global vector

Example:



Solver

Definition

The implicit time marching algorithm is encapsulated in an object call the Solver.

- Manages the solution of implicit algebraic equations at t_{n+1}
- Advances the time level when solution at t_{n+1} is known

Solution Streams

Definition

Component Models contain a collection of objects, called Solution Streams, in which solution data is written during the time stepping process.

- Solution Stream may be configured to "pipe" data to any output source; e.g. Writing data to disk, graphical output, etc.
- Solution Streams are configured by Client code–not Component Models

Implementation Notes

Written entirely in C++

- Allows Object Oriented Programming (OOP) design
- Mature compilers available for all platforms of interest
- ISO regulated, backward compatible
- Binding with C language trivial (mostly)

Abstract Base Classes and Virtual Functions are used

- Run-time polymorphism is needed since user inputs are not known at compile time

RTTI is used

- RTTI is required to obtain handles to component models from a system model
- Error handling uses exceptions, which means RTTI used anyway

Embedding Rodymol

Initializing Rodymol

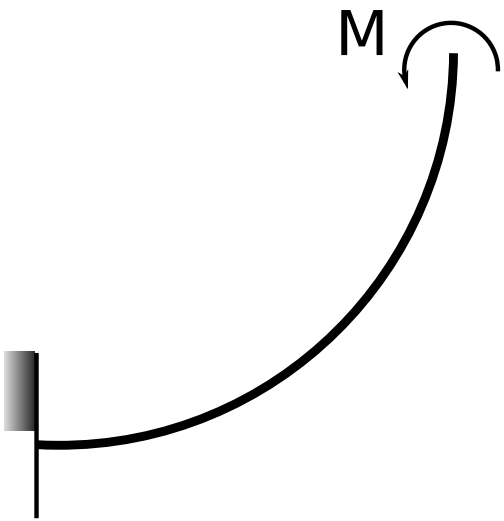
```
// Instantiate the system model...
ComponentModel* ptr = CompModelFactory::instance()
                                    .create("SystemModel", config_dir);
SystemModel* system = dynamic_cast<SystemModel*>(ptr);
// Setup solution streams...
system->configSolnStream(SolnStreamManager(output_dir));
// Create and link a solver ...
SolverIVP solver = new SolverIVP(*system);
// Optionally, get handles to driver components (example: rotor hub)...
ComponentModel* hub = SystemModel->componentModel(hub_name);
```

Executing a time step

```
// Set state of driver components, if any...
...
// Execute a time step...
solver->step();
// Optionally, get data from driver/observer components...
...
```

Outline

Elastica



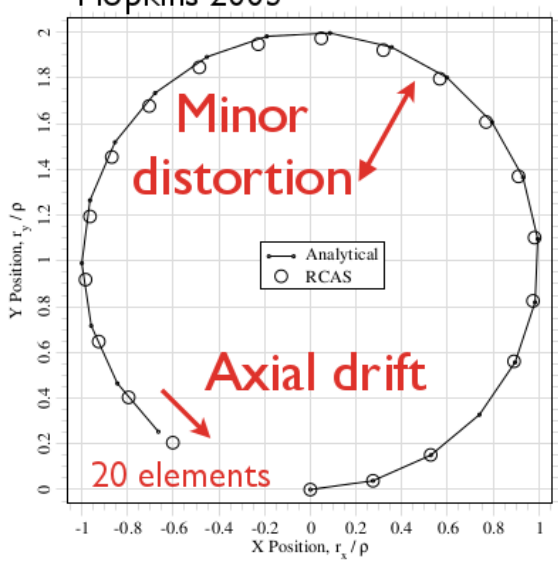
$$EI\frac{d\theta}{ds} = -M$$

$$\rho = -\frac{M}{EI}$$

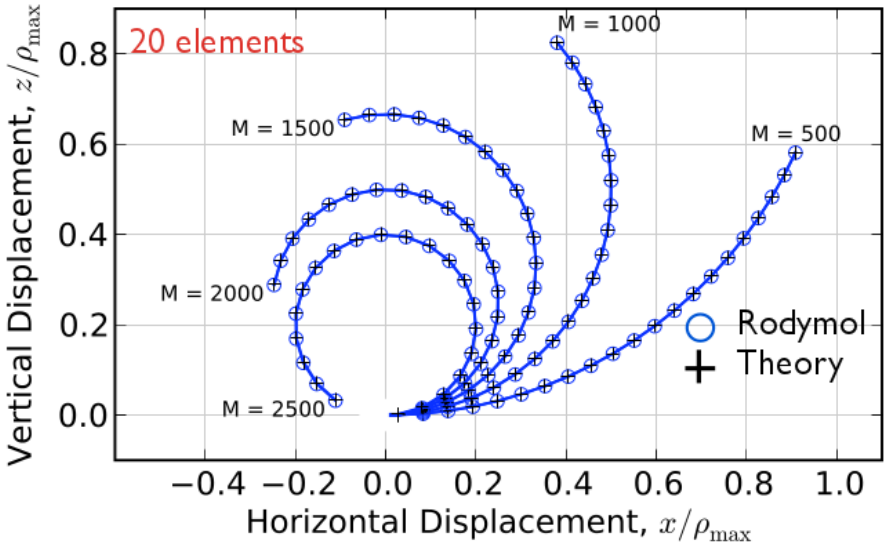
$$x(s) = \rho \sin(s/\rho)$$

$$y(s) = \rho(1 - \cos(s/\rho))$$

Hopkins 2003

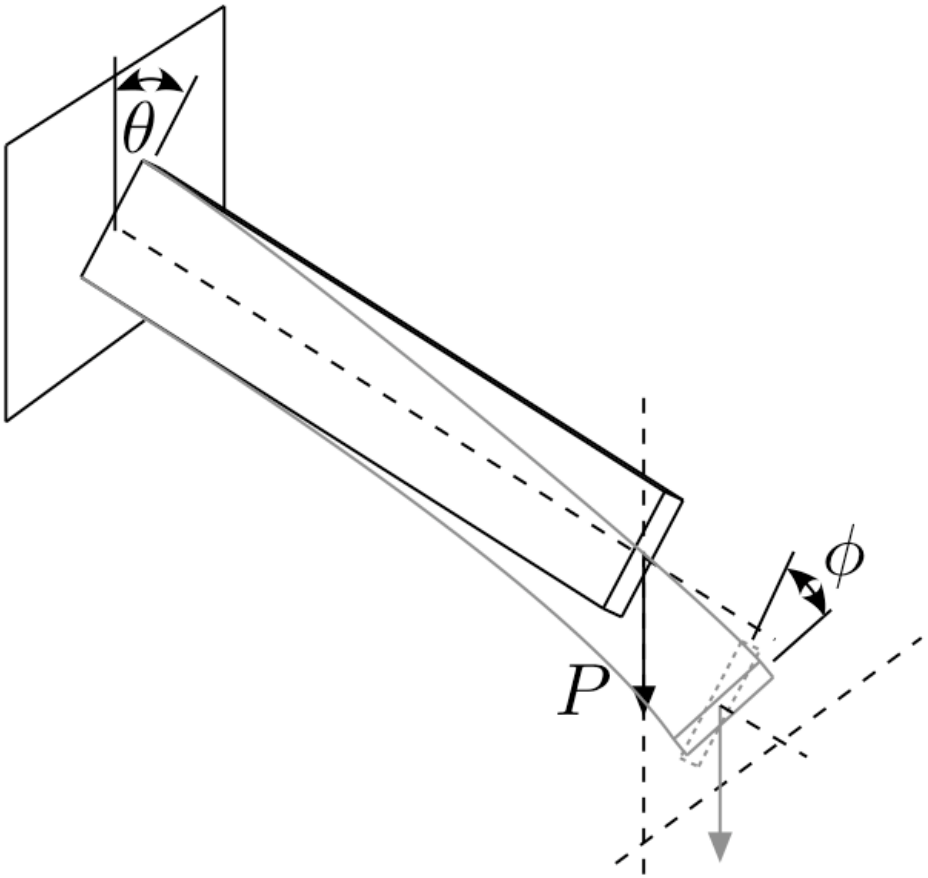


Rodymol



Outline

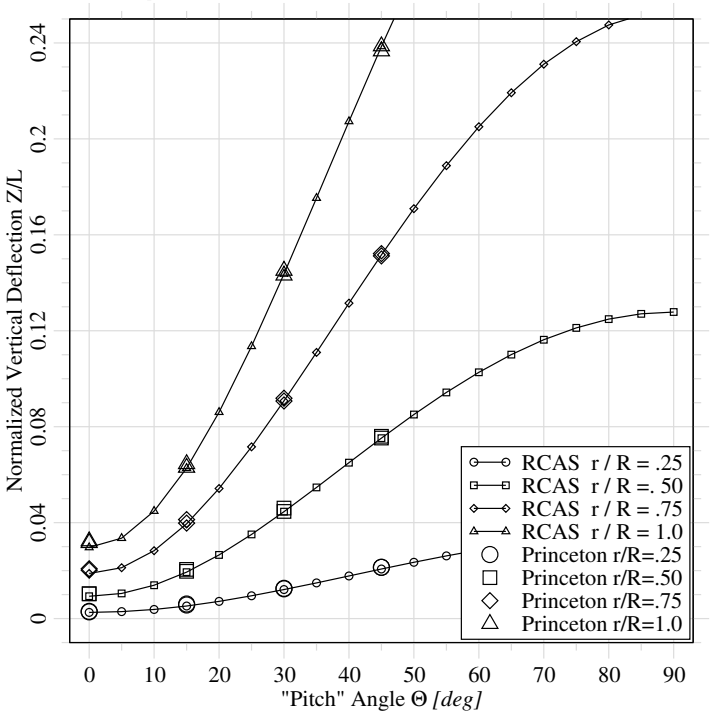
Princeton Beam



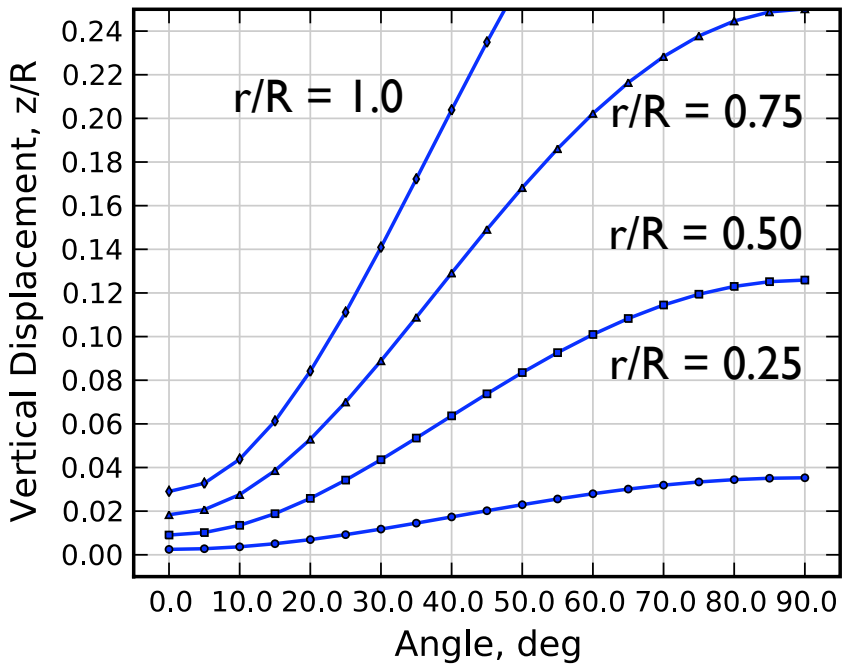
- Static large deformation test
- Uniform 7075-T651 Al beam
- Dead load applied to tip
- Coupled bending-torsion
- Measured displacements and twist

Princeton Beam: Vertical Displacements

Hopkins 2003



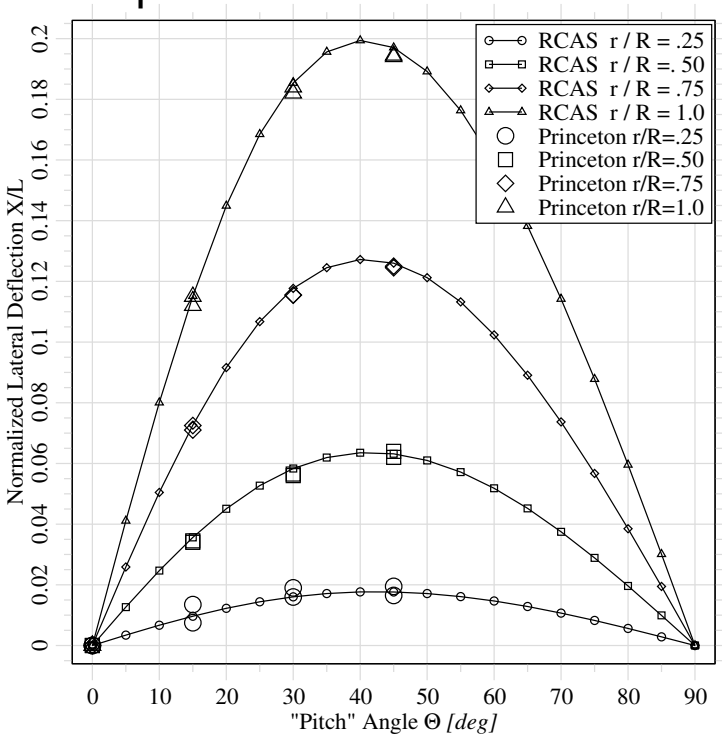
Rodymol



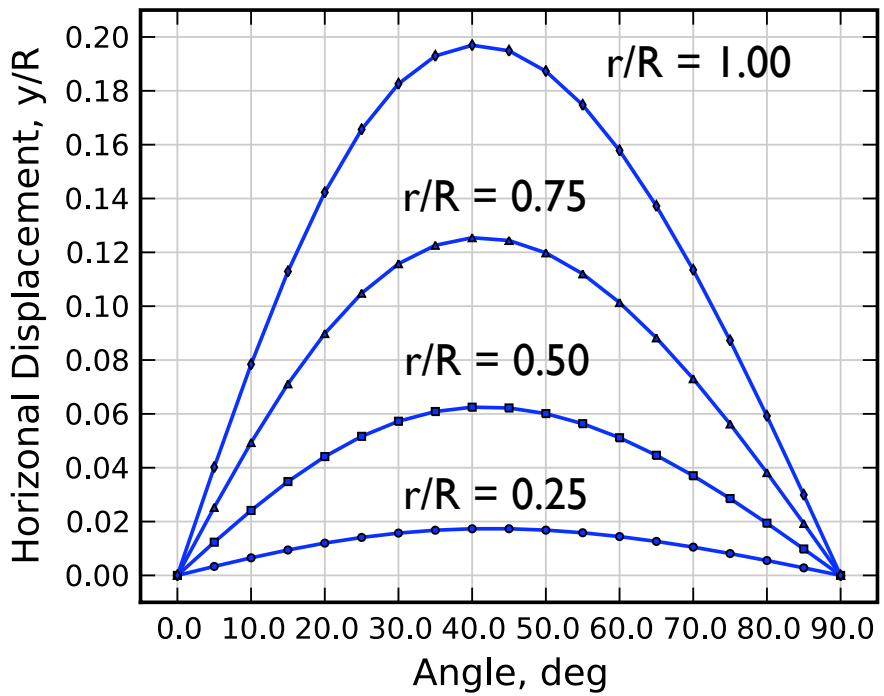
Good qualitative agreement with RCAS and experiment

Princeton Beam: Horizontal Displacements

Hopkins 2003



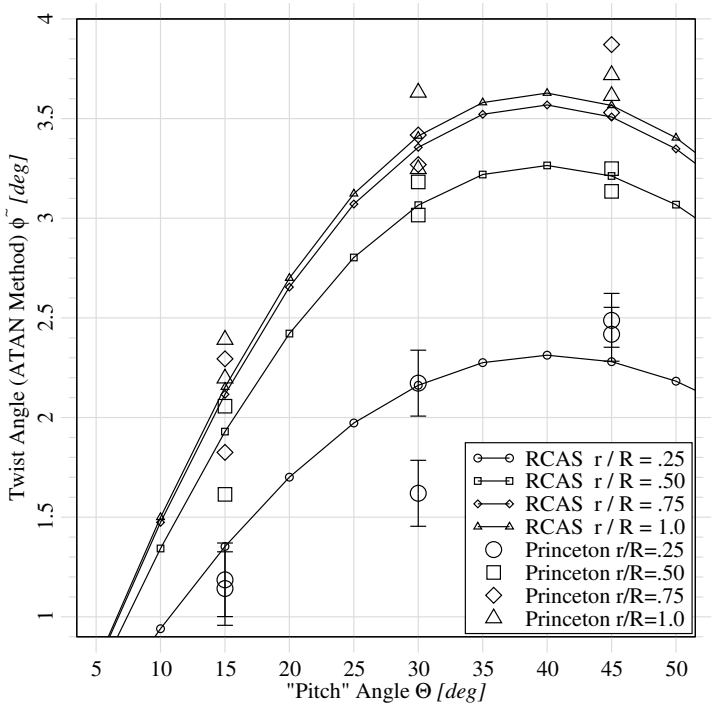
Rodymol



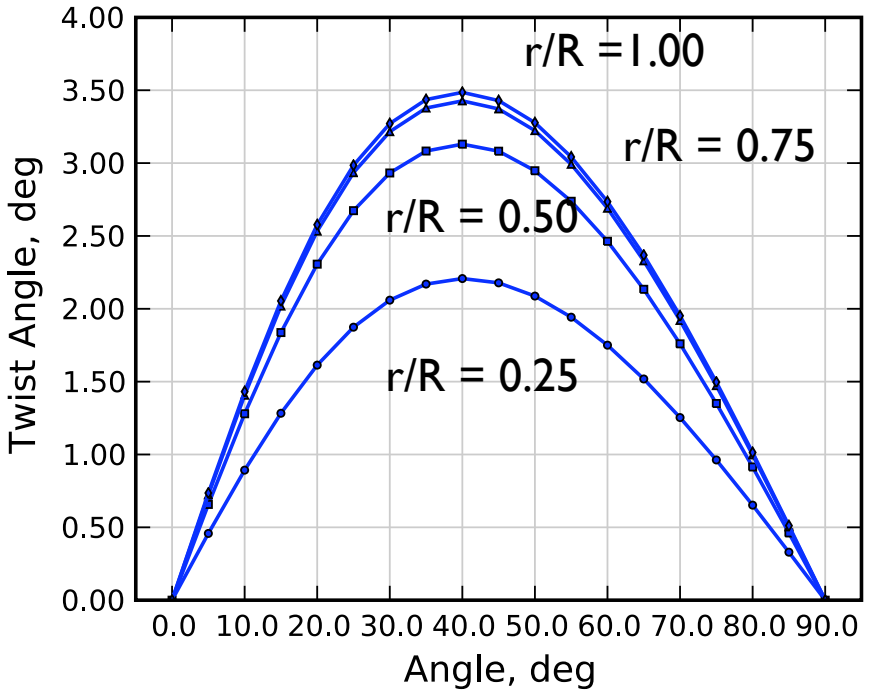
Good qualitative agreement with RCAS and experiment

Princeton Beam: Torsional Displacements

Hopkins 2003



Rodymol

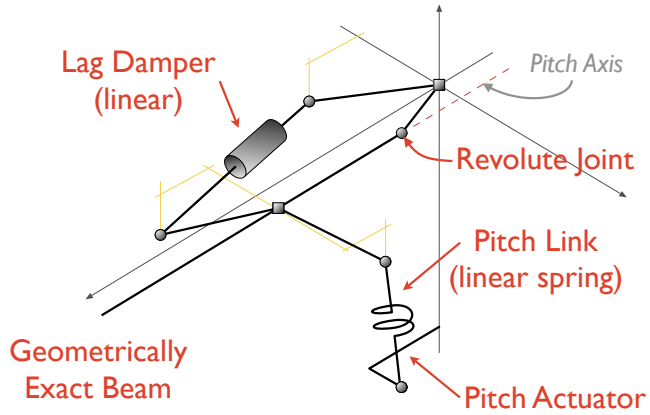


Good qualitative agreement with RCAS and experiment

Outline

UH60 Model Overview

- Blade modeled using two geometrically exact beams
- Beams joined at pitch horn and damper pick-up
- Currently using linear damper model
- Pitch link modeled as linear spring
- Controls imposed by "pitch actuator"
- Elastomeric bearing modeled as revolute joint
- Sweep modeled using "built-in shear"



Rotor properties based on UH60 Airloads Database

Flight Condition and Controls

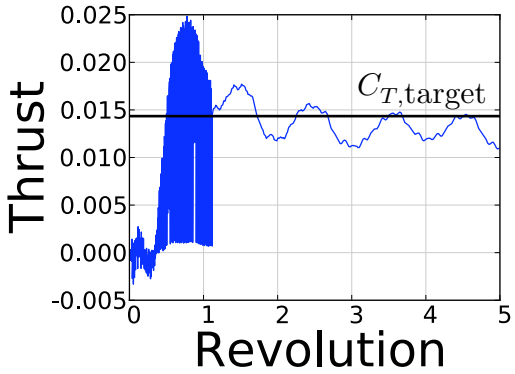
Flight Condition:

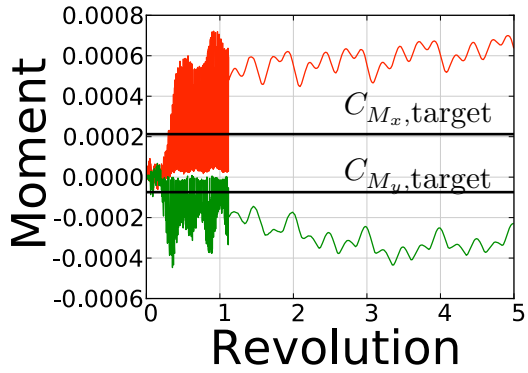
$$\mu = 0.368,\ C_T/\sigma = X,\ \gamma = 0°.$$

Controls:

	θ_0	θ_{1s}	θ_{1c}	
Flight Test	12.32	-9.79	4.68	← **Used**
OVERTURNS+UMARC	15.41	-9.13	4.30	

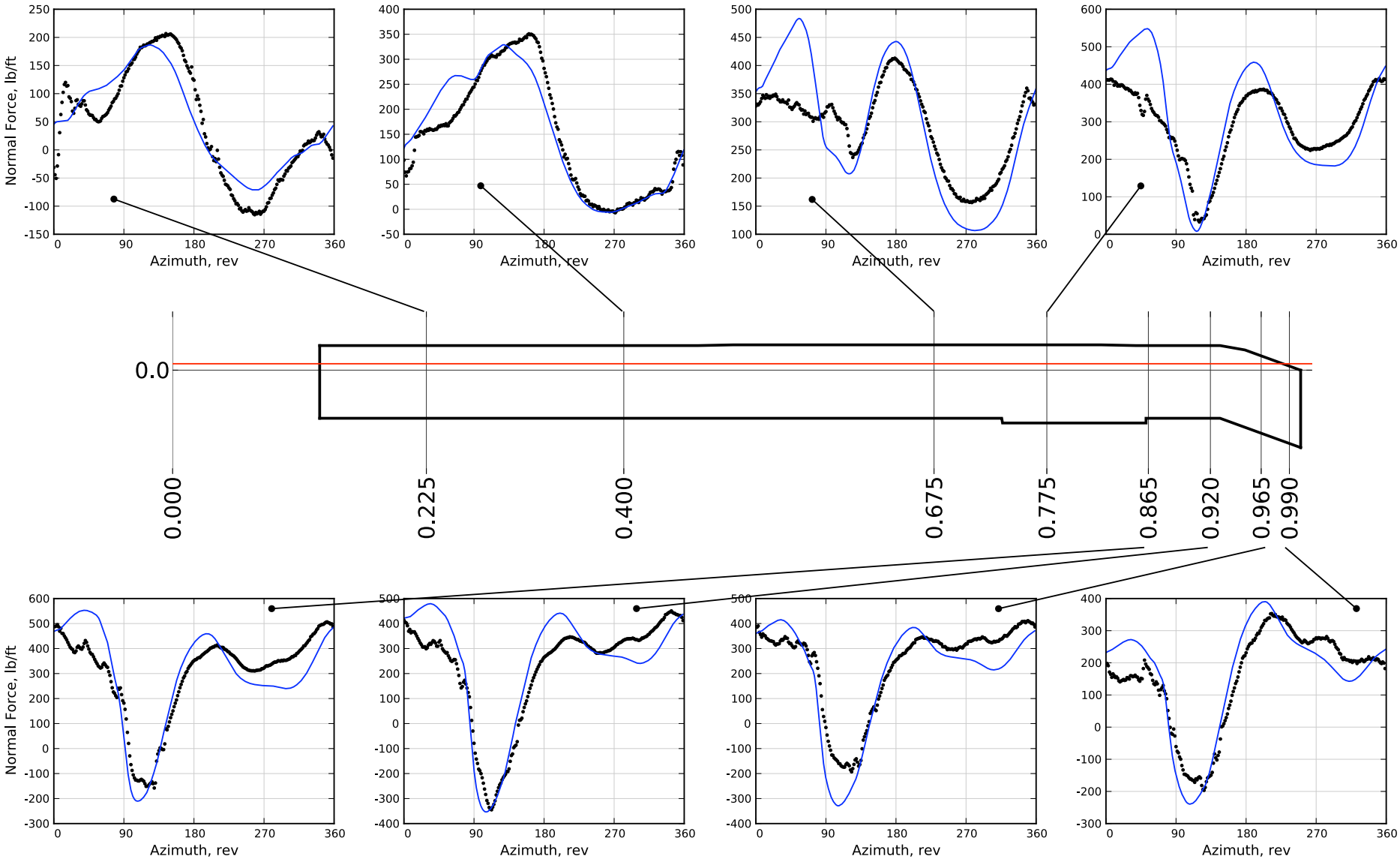
Thrust and Moment Histories:



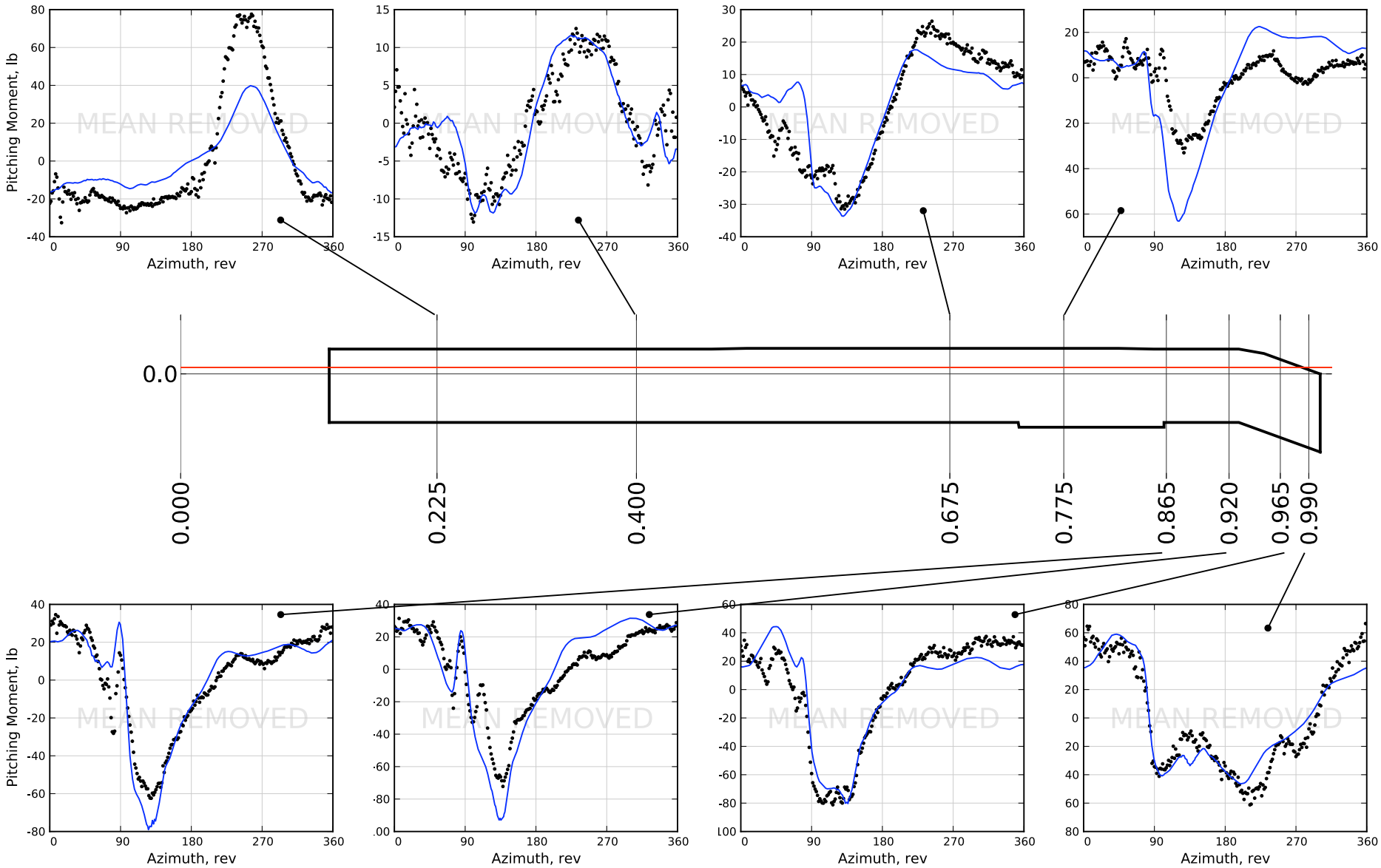


Rodymol+OVERTURNS results are not precisely trimmed

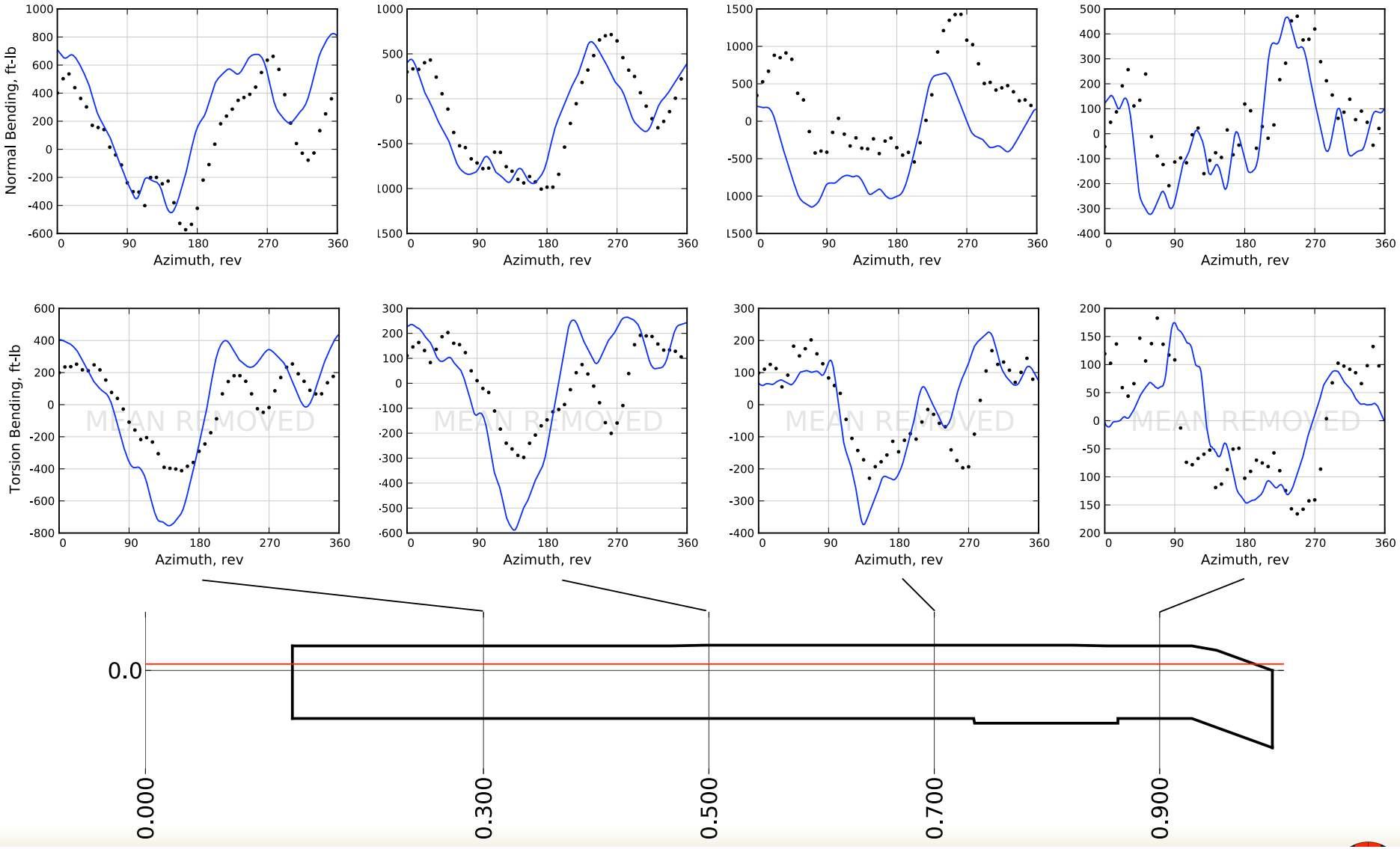
Normal Force



Pitching Moment



Structural Loads



Summary

Multibody Dynamics

- A topology independent methodology for solving the evolution equations of mechanical systems
- Facilitates the development of reusable, modular, software

Rodymol Methodology and Architecture

- Allows implicit or explicit treatment of constraints
- Structured around component model and connectivity node concepts
- Easily embedded in computational framework (CFD/CSD)
- Written entirely in C++

Rodymol Verification and Validation

- Excellent results for Elastica and Princeton Beam
- Good correlation for UH60 C11029 starting flight condition

Future Work

TODO List

- Add trimming mechanism (autopilot) for rotorcraft analysis
- Add mechanism for extracting mode shapes and frequencies
- Write theory manual
- Write user manual
- Look into open source licensing options

Wish List

- Add subsystem modeling capabilities
- Add shells and membranes to component model library
- Higher order reconstruction options for geometrically exact beam model

Questions?