



An Abaqus-Matlab Tutorial for Jointed Systems

International Committee on Jointed Structures Seminar Series

Nidish Narayanaa Balaji

Department of Aerospace Engineering, IIT Madras

March 26, 2025

Table of Contents

- 1 Introduction
- 2 Outline of the Steps
 - Relative Coordinates Pipeline
- 3 Nonlinear Analysis in MATLAB
- 4 Outro and Recommendations

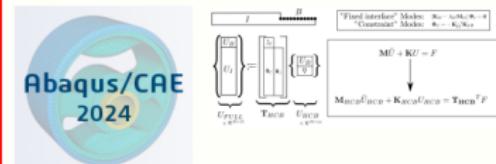


Detailed Instructions are hosted through Github
in a repository named Nidish96/Abaqus4Joints

Acknowledgements

- Prof. Matthew Brake
- Dr. Justin Porter, Maeve Karpov
- Prof. Matt Allen

Abaqus Workflow



$\underline{\underline{M}}$, $\underline{\underline{K}}$, \underline{F}_b , $\underline{\underline{R}}$

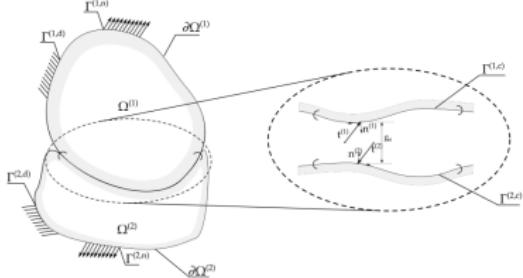
Mass, Stiffness
Bolt load vector
Selected rows for Recovery

DoFs: $3N_{int} + N_{gen}$

The research needs of the joints community are quite unique and specific

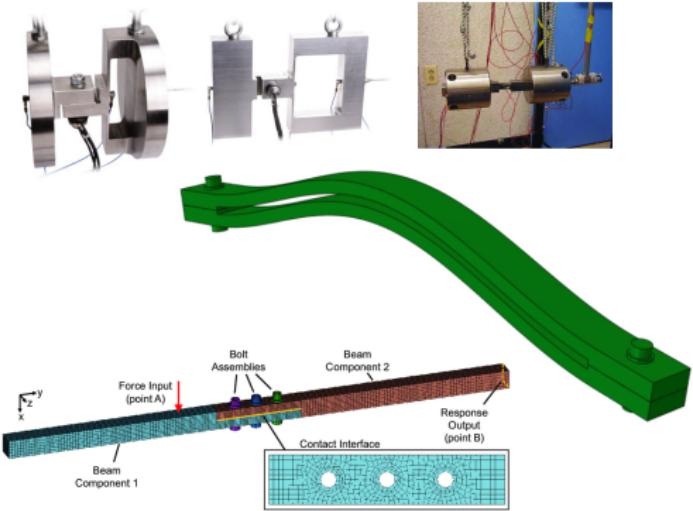
1. Introduction

The Generic Contact Problem



Joints Benchmarks ([see jointmechanics.org](http://jointmechanics.org))

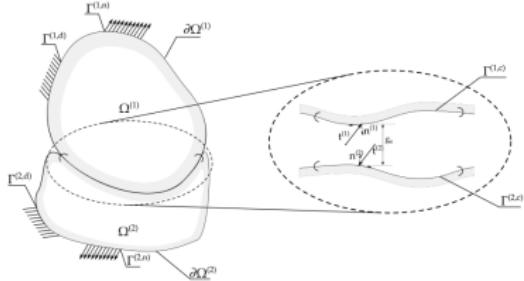
(see jointmechanics.org)



The research needs of the joints community are quite unique and specific

1. Introduction

The Generic Contact Problem

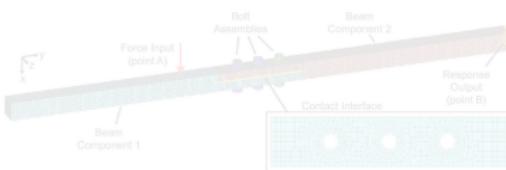


Joints Benchmarks (see jointmechanics.org)



Common theme:

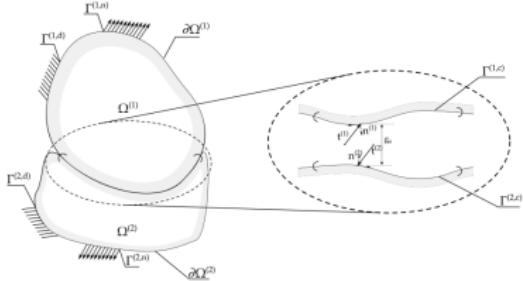
Linear subcomponents joined through a non-linear contact interface undergoing small deformation vibrations



The research needs of the joints community are quite unique and specific

1. Introduction

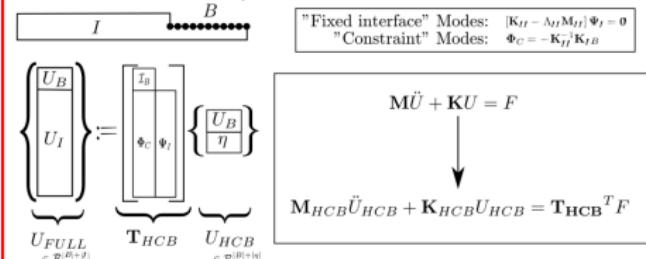
The Generic Contact Problem



Joints Benchmarks (jointmechanics.org)

Common theme:
Linear subcomponents joined through a non-linear contact interface undergoing small deformation vibrations

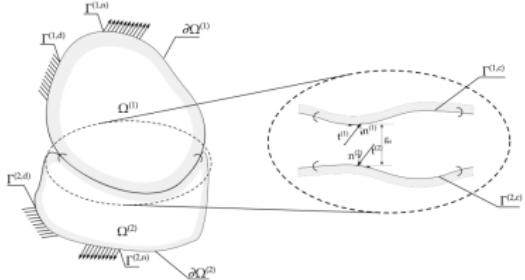
The HCB/CMS Methodology



The research needs of the joints community are quite unique and specific

1. Introduction

The Generic Contact Problem

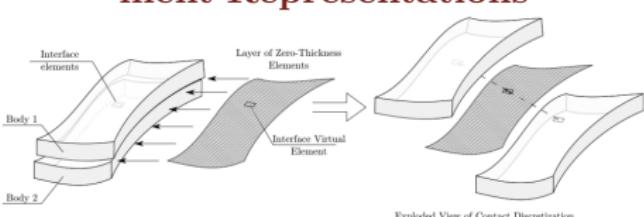


Joints Benchmarks (jointmechanics.org)

Common theme:

Linear subcomponents joined through a non-linear contact interface undergoing small deformation vibrations

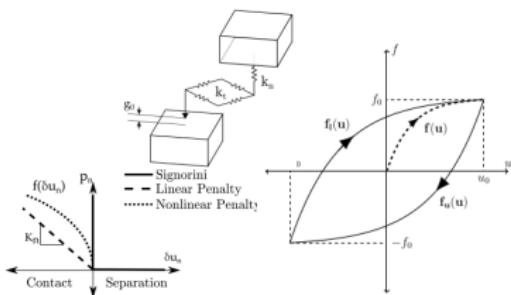
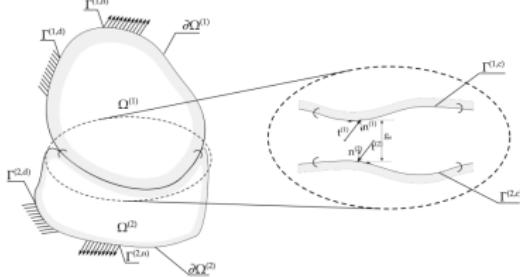
Interface Virtual Element Representations



The research needs of the joints community are quite unique and specific

1. Introduction

The Generic Contact Problem



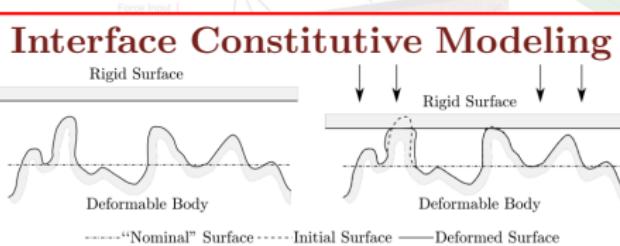
Joints Benchmarks (see jointmechanics.org)



Common theme:

Linear subcomponents joined through a non-linear contact interface undergoing small deformation vibrations

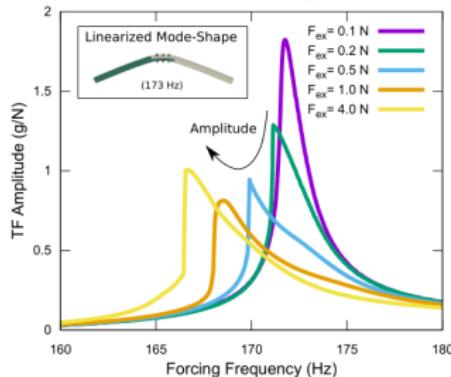
Interface Constitutive Modeling



The research needs of the joints community are quite unique and specific

1. Introduction

Nonlinear Responses

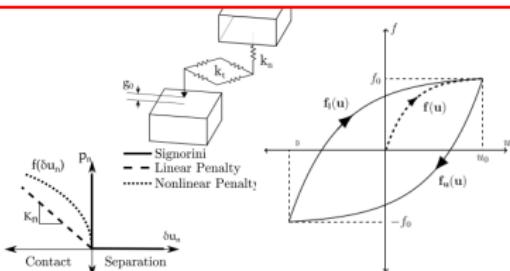


Joints Benchmarks (see jointmechanics.org)

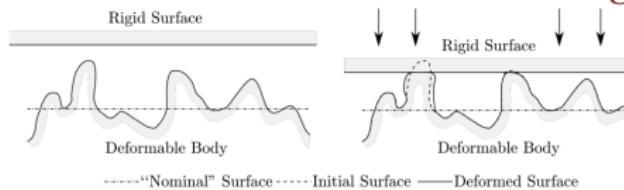


Common theme:

Linear subcomponents joined through a non-linear contact interface undergoing small deformation vibrations



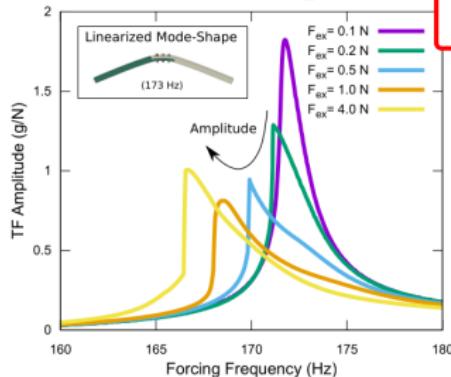
Interface Constitutive Modeling



The research needs of the joints community are quite unique and specific

1. Introduction

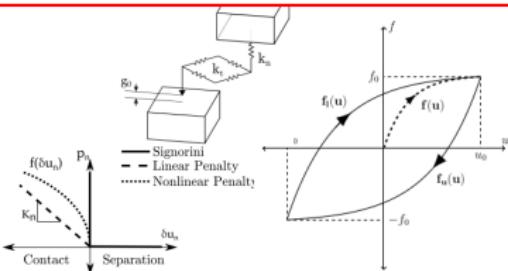
Nonlinear Responses



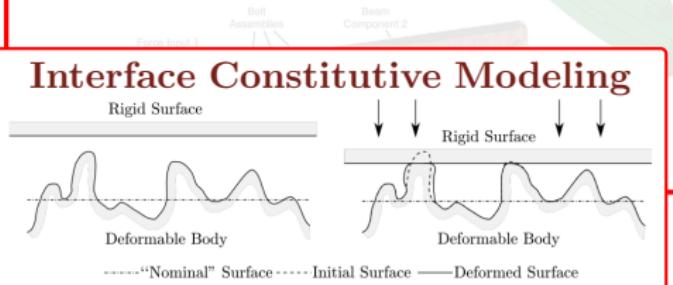
Goal: Off-Load as much as possible to **focus on joints**.

Common theme:

Linear subcomponents joined through a non-linear contact interface undergoing small deformation vibrations



Interface Constitutive Modeling

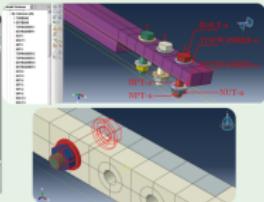
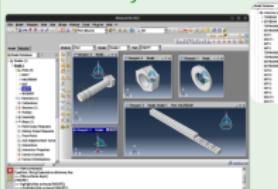


An Overview of the Modeling Pipeline

2. Outline of the Steps

Pre-Processing

Assembly



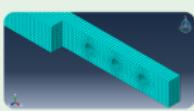
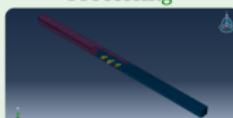
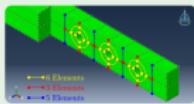
Constraints



Bolt Prestress as Static Load



Seeded Meshing, Processing



Component Mode Synthesis

$$\boxed{I} \quad \bullet\bullet\bullet \quad B$$

$\left\{ \begin{array}{c} U_B \\ U_I \end{array} \right\} \leftarrow \left[\begin{array}{c|c} I_0 & \\ \Phi_C & \Psi_I \end{array} \right] \left\{ \begin{array}{c} U_B \\ \eta \end{array} \right\}$

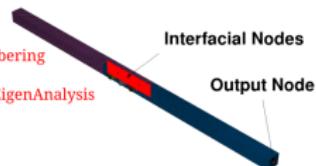
$U_{HCB}^{\text{FULL}} \quad \mathbf{T}_{HCB} \quad U_{HC}^{\text{CUT}}$

"Fixed interface" Modes: $[K_{IJ} - \Lambda_{IJ} M_{IJ}] \Psi_I = 0$
 "Constraint" Modes: $\Phi_C = -K_{IJ}^{-1} K_{IB}$

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{F}$$

$$\mathbf{M}_{HCB}\ddot{\mathbf{U}}_{HCB} + \mathbf{K}_{HCB}\mathbf{U}_{HCB} = \mathbf{T}_{HCB}^T F$$

- + Reorganize NodeSet numbering
- + Choose recovery nodes
- + Conduct Fixed Interface EigenAnalysis

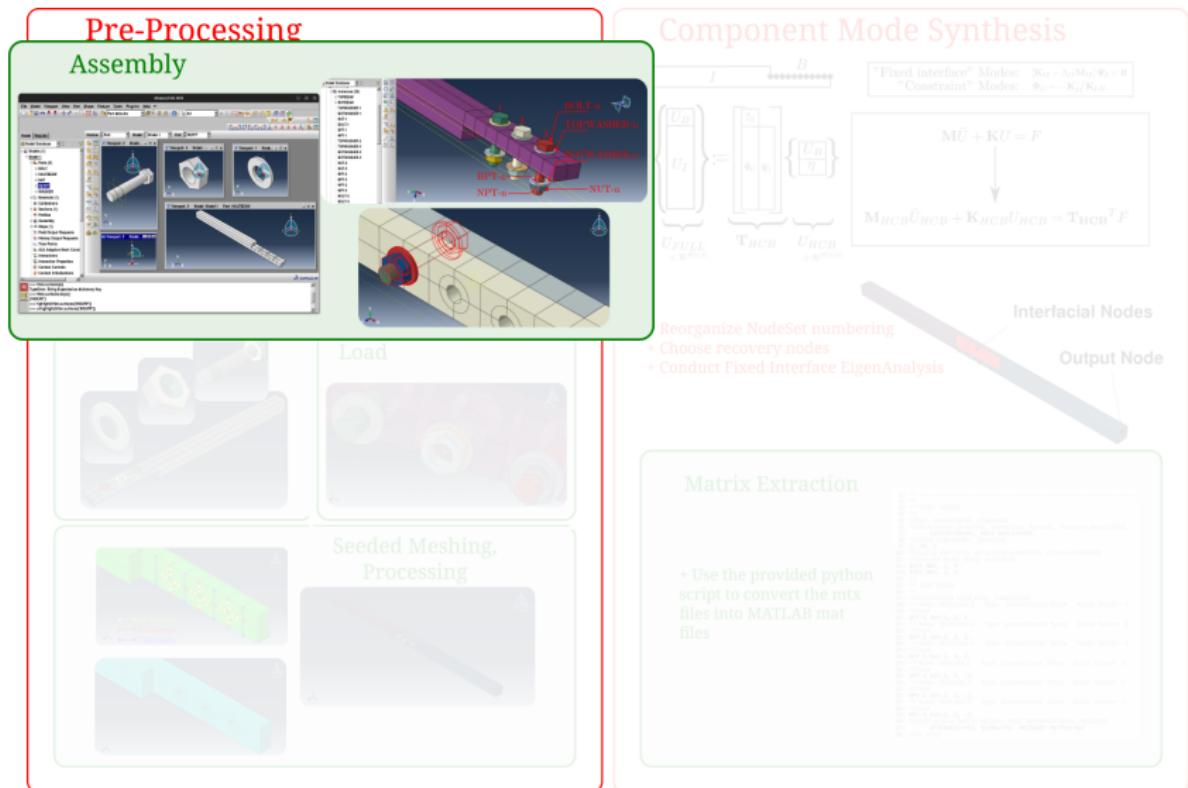


Matrix Extraction

- + Use the provided python script to convert the mtx files into MATLAB mat files

An Overview of the Modeling Pipeline

2. Outline of the Steps

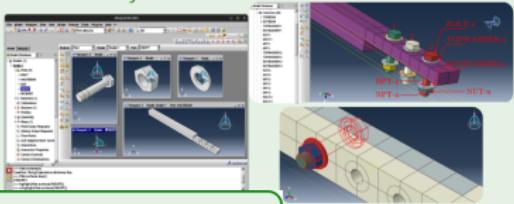


An Overview of the Modeling Pipeline

2. Outline of the Steps

Pre-Processing

Assembly



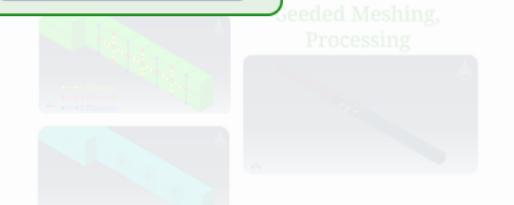
Constraints



Bolt Prestress as Static Load



Seeded Meshing Processing



An Overview of the Modeling Pipeline

2. Outline of the Steps

Pre-Processing

Assembly

Constraints

Bolt Prestress as Static Load

Processing

Component Mode Synthesis

"Fixed interface" Modes: $\dot{q}_{fix} = \Delta q_{fix}, \Psi_f = 0$
 "Constraint" Modes: $\Psi_c = 0, K_c/K_B$

$$\begin{aligned} I &= \dots \\ U_I &= \dots \\ U_{full} &= \dots \\ T_{HCB} &= \dots \\ U_{HCB} &= \dots \\ \bar{U} &= \dots \\ \bar{F} &= \dots \\ M\bar{U} + K\bar{U} &= \bar{F} \\ M_{HCB}\bar{U}_{HCB} + K_{HCB}\bar{U}_{HCB} &= T_{HCB}^T\bar{F} \end{aligned}$$

+ Reorganize NodeSet numbering
 + Choose recovery nodes
 + Create boundary conditions

**"Pull the Bolts,
Push the Washers"**

Matrix Extraction

```

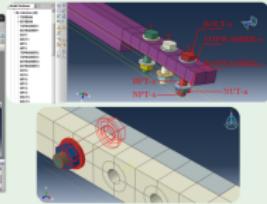
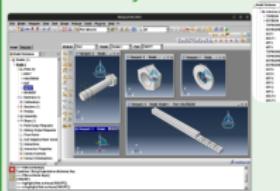
  + Use the provided python
  script to convert the mtx
  files into MATLAB mat
  files
  
```

An Overview of the Modeling Pipeline

2. Outline of the Steps

Pre-Processing

Assembly



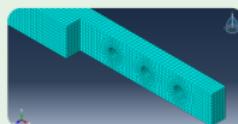
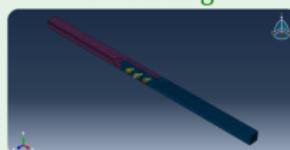
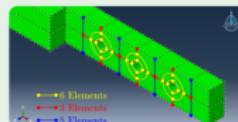
Constraints



Bolt Prestress as Static Load



Seeded Meshing, Processing

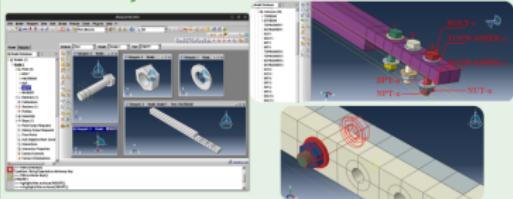


An Overview of the Modeling Pipeline

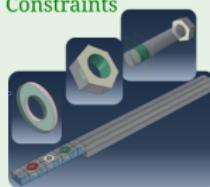
2. Outline of the Steps

Pre-Processing

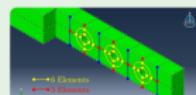
Assembly



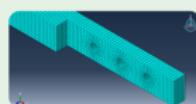
Constraints



Bolt Prestress as Static Load



Seeded Meshing Processing



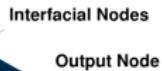
Component Mode Synthesis

$$\begin{aligned} I & \quad \text{"Fixed interface" Modes: } \mathbf{K}_{II} - \Lambda_{II}\mathbf{M}_{II}, \Psi_I = 0 \\ & \quad \text{"Constraint" Modes: } \Phi_C = -\mathbf{K}_{II}/\mathbf{K}_{IB} \\ \left\{ \begin{array}{l} U_B \\ U_I \end{array} \right\} := \left[\begin{array}{c|c} I_0 & \Phi_C \\ \Phi_C & \mathbf{I} \end{array} \right] \left\{ \begin{array}{l} U_B \\ \eta \end{array} \right\} \\ U_{HCB}^{\text{FULL}} & \in \mathbb{R}^{B \times n_B} \quad \mathbf{T}_{HCB} \quad \mathbf{U}_{HCB} \in \mathbb{N}^{B \times n_B} \end{aligned}$$

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{F}$$

$$\mathbf{M}_{HCB}\ddot{\mathbf{U}}_{HCB} + \mathbf{K}_{HCB}\mathbf{U}_{HCB} = \mathbf{T}_{HCB}^T \mathbf{F}$$

- + Reorganize NodeSet numbering
- + Choose recovery nodes
- + Conduct Fixed Interface EigenAnalysis



Matrix Extraction

- + Use the provided python script to convert the mtx files into MATLAB mat files

```

1: ** -----
2: 
3: ** STEP: HCB005
4: 
5: *Step, name=HCB005, elprint=0
6: *Substructure, matrix=COMBINED, visco=COMBINED
7: *mat=STRUCTURE, mass=STRUCTURE, type=EL, recovery matrix=YES,
8: *select Eigenvectors, generate
9: *load=0
10: *control, control=CONTROLS, visco=COMBINED
11: *load=0, initial def, sorted=NO
12: BOTS_NOS, 1, 3
13: TOP_NOS, 1, 3
14: 
15: ** LOAD CASES
16: 
17: *Substructure Load Case, name=CASE
18: *load=0, multLoad=1, Type: Concentrated force Scale Factor: 2
19: *clLoad
20: BPT-1_Set-1, 3, 1
21: BPT-1_Set-1, 3, 1, Type: Concentrated force Scale Factor: 2
22: *clLoad
23: BPT-2_Set-1, 2, 1
24: BPT-2_Set-1, 2, 1, Type: Concentrated force Scale Factor: 2
25: *clLoad
26: BPT-3_Set-1, 2, 1
27: BPT-3_Set-1, 2, 1, Type: Concentrated force Scale Factor: 2
28: 
29: HPT-1_Set-1, 3, 1
30: HPT-1_Set-1, 3, 1, Type: Concentrated force Scale Factor: 2
31: *clLoad
32: HPT-2_Set-1, 3, 1
33: HPT-2_Set-1, 3, 1, Type: Concentrated force Scale Factor: 2
34: *clLoad
35: HPT-3_Set-1, 3, 1
36: HPT-3_Set-1, 3, 1, Type: Concentrated force Scale Factor: 2
37: 
38: *Substructure Matrix Output, FILE NAME=Results.mtx, MASS=YES,
39: STIFFNESS=YES, LOAD=YES, RECOVERY MATRIX=YES
40: *end Step

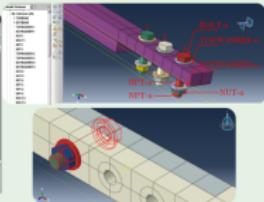
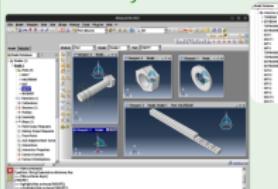
```

An Overview of the Modeling Pipeline

2. Outline of the Steps

Pre-Processing

Assembly



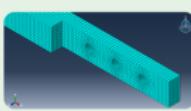
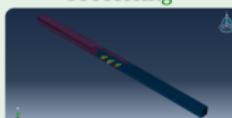
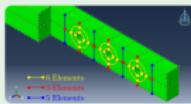
Constraints



Bolt Prestress as Static Load



Seeded Meshing, Processing



Component Mode Synthesis

$$\boxed{I} \quad \bullet\bullet\bullet \quad B$$

$\left\{ \begin{array}{c} U_B \\ U_I \end{array} \right\} \leftarrow \left[\begin{array}{c|c} I_0 & \\ \Phi_C & \Psi_I \end{array} \right] \left\{ \begin{array}{c} U_B \\ \eta \end{array} \right\}$

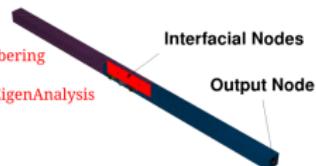
$U_{HCB}^{\text{FULL}} \quad \mathbf{T}_{HCB} \quad U_{HC}^{\text{CUT}}$

"Fixed interface" Modes: $[K_{IJ} - \Lambda_{IJ} M_{IJ}] \Psi_I = 0$
 "Constraint" Modes: $\Phi_C = -K_{IJ}^{-1} K_{IB}$

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{F}$$

$$\mathbf{M}_{HCB}\ddot{\mathbf{U}}_{HCB} + \mathbf{K}_{HCB}\mathbf{U}_{HCB} = \mathbf{T}_{HCB}^T F$$

- + Reorganize NodeSet numbering
- + Choose recovery nodes
- + Conduct Fixed Interface EigenAnalysis



Matrix Extraction

- + Use the provided python script to convert the mtx files into MATLAB mat files

2.1. Relative Coordinates Pipeline

[Outline of the Steps](#)

- Suppose that $\underline{u}_T, \underline{u}_B, \underline{u}_I$ are the vectors of top, bottom, and internal nodal DoFs.
- Then the governing equations look like:

$$\begin{bmatrix} \underline{\underline{M}}_{TT} & \underline{\underline{M}}_{TB} & \underline{\underline{M}}_{TI} \\ \text{sym} & \underline{\underline{M}}_{BB} & \underline{\underline{M}}_{BI} \\ & \underline{\underline{M}}_{IT} & \underline{\underline{M}}_{II} \end{bmatrix} \begin{bmatrix} \ddot{\underline{u}}_T \\ \ddot{\underline{u}}_B \\ \ddot{\underline{u}}_I \end{bmatrix} + \begin{bmatrix} \underline{\underline{K}}_{TT} & \underline{\underline{K}}_{TB} & \underline{\underline{K}}_{TI} \\ \text{sym} & \underline{\underline{K}}_{BB} & \underline{\underline{K}}_{BI} \\ & \underline{\underline{K}}_{IT} & \underline{\underline{K}}_{II} \end{bmatrix} \begin{bmatrix} \underline{u}_T \\ \underline{u}_B \\ \underline{u}_I \end{bmatrix} + \begin{bmatrix} \underline{\underline{F}}_T^{(c)} \\ \underline{\underline{F}}_B^{(c)} \\ 0 \end{bmatrix} = \underline{\underline{F}}_e(t)$$

2.1. Relative Coordinates Pipeline

[Outline of the Steps](#)

- Suppose that $\underline{u}_T, \underline{u}_B, \underline{u}_I$ are the vectors of top, bottom, and internal nodal DoFs.
- Then the governing equations look like:

$$\begin{bmatrix} \underline{\underline{M}}_{TT} & \underline{\underline{M}}_{TB} & \underline{\underline{M}}_{TI} \\ \text{sym} & \underline{\underline{M}}_{BB} & \underline{\underline{M}}_{BI} \\ & \underline{\underline{M}}_{IT} & \underline{\underline{M}}_{II} \end{bmatrix} \begin{bmatrix} \ddot{\underline{u}}_T \\ \ddot{\underline{u}}_B \\ \ddot{\underline{u}}_I \end{bmatrix} + \begin{bmatrix} \underline{\underline{K}}_{TT} & \underline{\underline{K}}_{TB} & \underline{\underline{K}}_{TI} \\ \text{sym} & \underline{\underline{K}}_{BB} & \underline{\underline{K}}_{BI} \\ & \underline{\underline{K}}_{IT} & \underline{\underline{K}}_{II} \end{bmatrix} \begin{bmatrix} \underline{u}_T \\ \underline{u}_B \\ \underline{u}_I \end{bmatrix} + \begin{bmatrix} \underline{F}_T^{(c)} \\ \underline{F}_B^{(c)} \\ 0 \end{bmatrix} = \underline{F}_e(t)$$

- Under the relative coordinate transformation,

$$\begin{bmatrix} \underline{u}_T \\ \underline{u}_B \\ \underline{u}_I \end{bmatrix} = \begin{bmatrix} \underline{\underline{I}}_T & \underline{\underline{I}}_T & \underline{\underline{0}} \\ \underline{\underline{0}} & \underline{\underline{I}}_T & \underline{\underline{0}} \\ \underline{\underline{0}} & \underline{\underline{0}} & \underline{\underline{I}}_I \end{bmatrix} \begin{bmatrix} \Delta \underline{u} \\ \underline{u}_B \\ \underline{u}_I \end{bmatrix}, \quad \Delta \underline{u} = \underline{u}_T - \underline{u}_B.$$

2.1. Relative Coordinates Pipeline

[Outline of the Steps](#)

- Suppose that $\underline{u}_T, \underline{u}_B, \underline{u}_I$ are the vectors of top, bottom, and internal nodal DoFs.
- Then the governing equations look like:

$$\begin{bmatrix} \underline{\underline{M}}_{TT} & \underline{\underline{M}}_{TB} & \underline{\underline{M}}_{TI} \\ \text{sym} & \underline{\underline{M}}_{BB} & \underline{\underline{M}}_{BI} \\ & \underline{\underline{M}}_{IT} & \underline{\underline{M}}_{II} \end{bmatrix} \begin{bmatrix} \ddot{\underline{u}}_T \\ \ddot{\underline{u}}_B \\ \ddot{\underline{u}}_I \end{bmatrix} + \begin{bmatrix} \underline{\underline{K}}_{TT} & \underline{\underline{K}}_{TB} & \underline{\underline{K}}_{TI} \\ \text{sym} & \underline{\underline{K}}_{BB} & \underline{\underline{K}}_{BI} \\ & \underline{\underline{K}}_{IT} & \underline{\underline{K}}_{II} \end{bmatrix} \begin{bmatrix} \underline{u}_T \\ \underline{u}_B \\ \underline{u}_I \end{bmatrix} + \begin{bmatrix} \underline{F}_T^{(c)} \\ \underline{F}_B^{(c)} \\ 0 \end{bmatrix} = \underline{F}_e(t)$$

- Under the relative coordinate transformation,

Transformed Equations of Motion

$$\begin{bmatrix} \underline{\underline{M}}_{TT} & \underline{\underline{M}}_{TT} + \underline{\underline{M}}_{TB} & \underline{\underline{M}}_{TI} \\ \text{sym} & \underline{\underline{M}}_{TT} + \underline{\underline{M}}_{BB} + \underline{\underline{M}}_{TB} + \underline{\underline{M}}_{TB}^T & \underline{\underline{M}}_{TI} + \underline{\underline{M}}_{BI} \\ & \underline{\underline{M}}_{TB} + \underline{\underline{M}}_{TB}^T & \underline{\underline{M}}_{II} \end{bmatrix} \begin{bmatrix} \ddot{\underline{u}}_T \\ \ddot{\underline{u}}_B \\ \ddot{\underline{u}}_I \end{bmatrix} + \begin{bmatrix} \underline{\underline{K}} \\ \underline{\underline{K}} \\ \underline{\underline{K}} \end{bmatrix} \begin{bmatrix} \underline{u}_T \\ \underline{u}_B \\ \underline{u}_I \end{bmatrix} + \begin{bmatrix} \underline{F}_T^{(c)} \\ 0 \\ 0 \end{bmatrix} = \underline{F}_e(t).$$

2.1. Relative Coordinates Pipeline

Outline of the Steps

- Suppose that $\underline{u}_T, \underline{u}_B, \underline{u}_I$ are the vectors of top, bottom, and internal nodal DoFs.
- Then the governing equations look like:

$$\begin{bmatrix} \underline{\underline{M}}_{TT} & \underline{\underline{M}}_{TB} & \underline{\underline{M}}_{TI} \\ \text{sym} & \underline{\underline{M}}_{BB} & \underline{\underline{M}}_{BI} \\ & \underline{\underline{M}}_{IT} & \underline{\underline{M}}_{II} \end{bmatrix} \begin{bmatrix} \ddot{\underline{u}}_T \\ \ddot{\underline{u}}_B \\ \ddot{\underline{u}}_I \end{bmatrix} + \begin{bmatrix} \underline{\underline{K}}_{TT} & \underline{\underline{K}}_{TB} & \underline{\underline{K}}_{TI} \\ \text{sym} & \underline{\underline{K}}_{BB} & \underline{\underline{K}}_{BI} \\ & \underline{\underline{K}}_{IT} & \underline{\underline{K}}_{II} \end{bmatrix} \begin{bmatrix} \underline{u}_T \\ \underline{u}_B \\ \underline{u}_I \end{bmatrix} + \begin{bmatrix} \underline{F}_T^{(c)} \\ \underline{F}_B^{(c)} \\ 0 \end{bmatrix} = \underline{F}_e(t)$$

- Under the relative coordinate transformation,

Transformed Equations of Motion

$$\begin{bmatrix} \underline{\underline{M}}_{TT} & \underline{\underline{M}}_{TT} + \underline{\underline{M}}_{TB} & \underline{\underline{M}}_{TI} \\ \text{sym} & \underline{\underline{M}}_{TT} + \underline{\underline{M}}_{BB} + \underline{\underline{M}}_{TB}^T & \underline{\underline{M}}_{TI} + \underline{\underline{M}}_{BI} \\ & \underline{\underline{M}}_{TB} + \underline{\underline{M}}_{TB}^T & \underline{\underline{M}}_{II} \end{bmatrix} \begin{bmatrix} \ddot{\underline{u}}_T \\ \ddot{\underline{u}}_B \\ \ddot{\underline{u}}_I \end{bmatrix} + \begin{bmatrix} \underline{\underline{K}} \\ \underline{\underline{K}} \\ \underline{\underline{K}} \end{bmatrix} \begin{bmatrix} \underline{u}_T \\ \underline{u}_B \\ \underline{u}_I \end{bmatrix} + \begin{bmatrix} \underline{F}_T^{(c)} \\ 0 \\ 0 \end{bmatrix} = \underline{F}_e(t).$$

2.1. Relative Coordinates Pipeline

[Outline of the Steps](#)

- Suppose that $\underline{u}_T, \underline{u}_B, \underline{u}_I$ are the vectors of top, bottom, and internal nodal DoFs.
- Then the governing equations look like:

$$\begin{bmatrix} \underline{\underline{M}} \\ \vdots \\ \text{sym} \end{bmatrix}$$

It is clearly advantageous (when possible) to use the relative coordinate transformation as early as possible in the CMS process.

Thankfully, this is possible fully in Abaqus itself!

- Under the relative coordinate transformation,

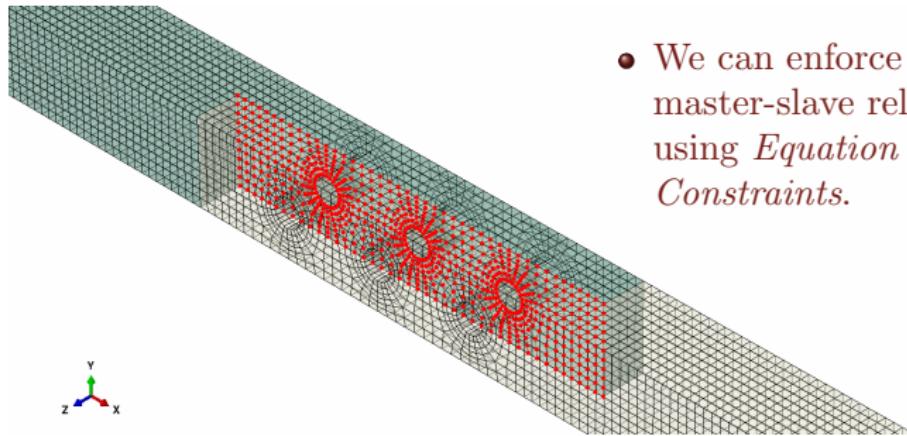
Transformed Equations of Motion

$$\begin{bmatrix} \underline{\underline{M}}_{TT} & \underline{\underline{M}}_{TT} + \underline{\underline{M}}_{TB} & \underline{\underline{M}}_{TI} \\ \vdots & \vdots & \vdots \\ \text{sym} & \underline{\underline{M}}_{TT} + \underline{\underline{M}}_{BB} + \underline{\underline{M}}_{TB}^T & \underline{\underline{M}}_{TI} + \underline{\underline{M}}_{BI} \\ & \underline{\underline{M}}_{TB} + \underline{\underline{M}}_{TB}^T & \underline{\underline{M}}_{II} \end{bmatrix} \begin{bmatrix} \ddot{\underline{u}}_T \\ \ddot{\underline{u}}_B \\ \ddot{\underline{u}}_I \end{bmatrix} + \begin{bmatrix} \bar{K} \\ \bar{K} \\ \bar{K} \end{bmatrix} \begin{bmatrix} \underline{u}_T \\ \underline{u}_B \\ \underline{u}_I \end{bmatrix} + \begin{bmatrix} \underline{F}_T^{(c)} \\ \underline{0} \\ \underline{0} \end{bmatrix} = \underline{F}_e(t).$$

2.1. Relative Coordinates Pipeline

Outline of the Steps

- By adding Reference Points to the model at the location of the interfacial nodes, we can create **slave nodes whos DoF will be the relative DoF of the corresponding nodes on the interface.**
- This can be done quite easily with Python scripting (the website has a script for the BRB which can be adapted to arbitrary contexts).



- We can enforce the master-slave relationship using *Equation Constraints*.

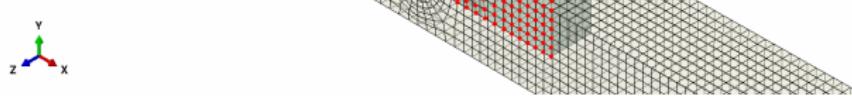
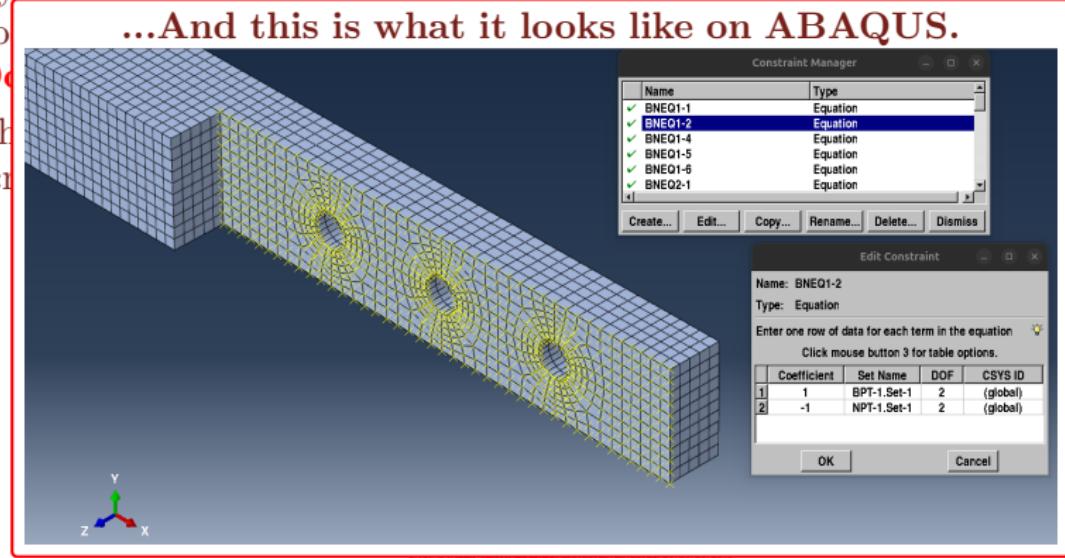
2.1. Relative Coordinates Pipeline

[Outline of the Steps](#)

- By adding Reference Points to the model at the location of the interfacial nodes

...And this is what it looks like on ABAQUS.

- Then we can define the contact pairs as a combination of these reference points



2.1. Relative Coordinates Pipeline

Outline of the Steps

- By adding Reference Points to the model at the location of the interfacial no ...And this is what it looks like on ABAQUS.

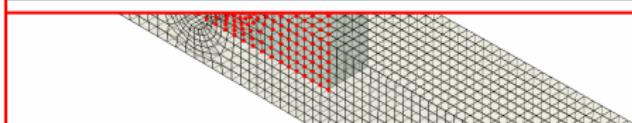
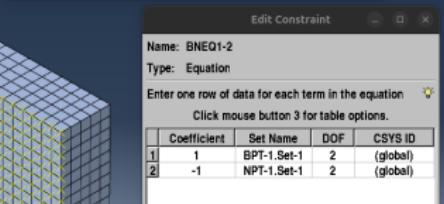
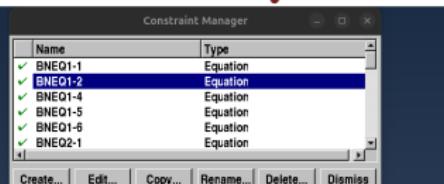
De

- The scri

.inp File Entries:

```
** Constraint: RELCS-1
*Equation
3
TOPS_NDS, 1, 1.
RELCSET, 1, -1.
BOTS_NDS, 1, -1.
** Constraint: RELCS-2
*Equation
3
TOPS_NDS, 2, 1.
RELCSET, 2, -1.
BOTS_NDS, 2, -1.
** Constraint: RELCS-3
*Equation
3
TOPS_NDS, 3, 1.
RELCSET, 3, -1.
BOTS_NDS, 3, -1.
```

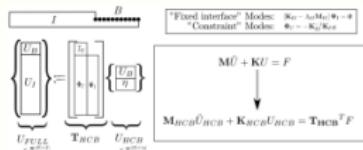
$$\underline{u}_{TOP} - \underline{u}_{rel} - \underline{u}_{BOT} = 0$$



Nonlinear Analysis in Sub-100 Lines of MATLAB Code!

3. Nonlinear Analysis in MATLAB

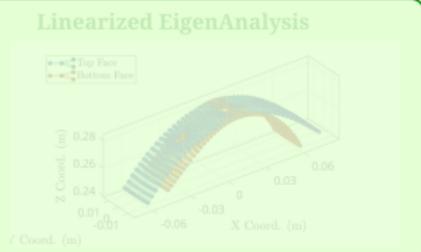
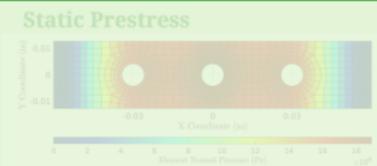
Abaqus Workflow



M , K , F_b , R
 Mass, Stiffness
 Bolt load vector
 Selected rows for Recovery

DoFs: $3N_{\text{int}} + N_{\text{gen}}$

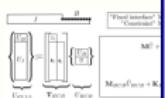
W
 and Elements
 s, vector (s)
 Rule
 Node (s)
 e Constitution
 d Solve!



Nonlinear Analysis in Sub-100 Lines of MATLAB Code!

3. Nonlinear Analysis in MATLAB

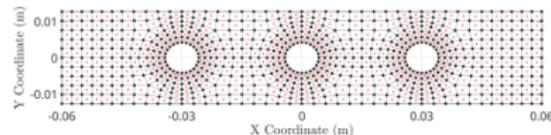
Abaqus Workflow



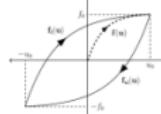
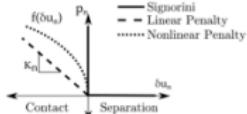
M , K , F_b
 Mass, Stiffness
 Bolt load vector
 Selected rows for Recovery
 DoFs: $3N_{int} + N_{gen}$

MATLAB Workflow

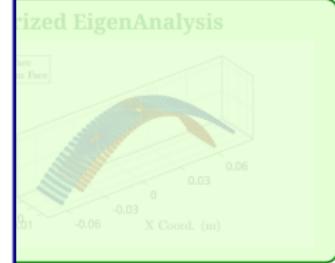
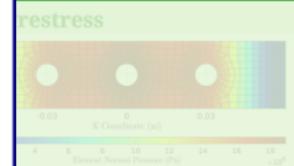
1. Read in Nodes and Elements
(Only interface)
2. Read in Matrices, vector (s)
3. Choose Quadrature Rule



4. Choose Interface Constitution



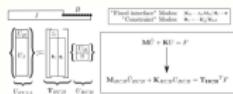
5. Apply Loads and Solve!



Nonlinear Analysis in Sub-100 Lines of MATLAB Code!

3. Nonlinear Analysis in MATLAB

Abaqus Workflow

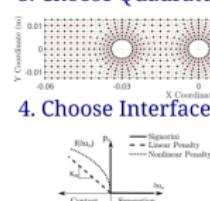


M , K , F_b , R
 Mass, Stiffness
 Bolt load vector
 Selected rows for Recovery

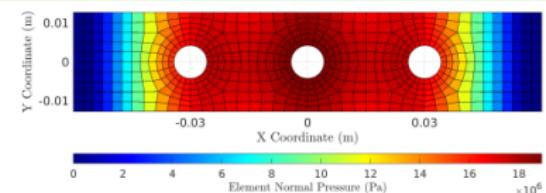
DoFs: $3N_{int} + N_{gen}$

MATLAB Workflow

1. Read in Nodes and Elements
2. Read in Matrices
3. Choose Quadrature
4. Choose Interface
5. Apply Loads and Constraints

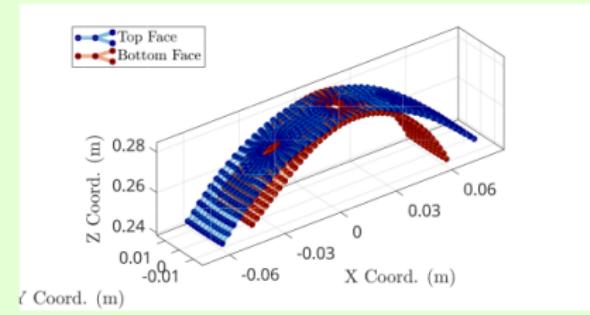


Static Prestress



Linearized EigenAnalysis

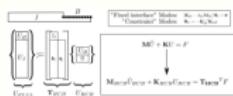
Linearized EigenAnalysis



Nonlinear Analysis in Sub-100 Lines of MATLAB Code!

3. Nonlinear Analysis in MATLAB

Abaqus Workflow

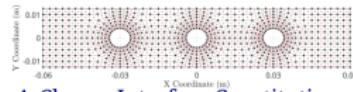


M , K , F_b , R
 Mass, Stiffness
 Bolt load vector
 Selected rows for Recovery

DoFs: $3N_{int} + N_{gen}$

MATLAB Workflow

1. Read in Nodes and Elements
(Only interface)
2. Read in Matrices, vector (s)
3. Choose Quadrature Rule

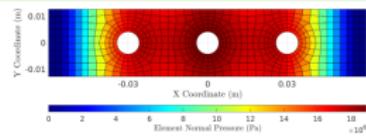


4. Choose Interface Constitution

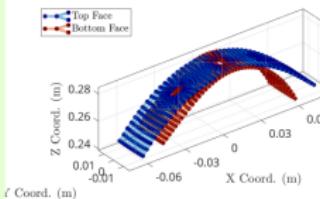


5. Apply Loads and Solve!

Static Prestress



Linearized EigenAnalysis



Comparison Between Absolute and Relative Coordinate Pipelines

Nonlinear Analysis in MATLAB

Component-Mode Synthesis

Absolute Coordinates (AC) Fixed interface HCB/CMS.

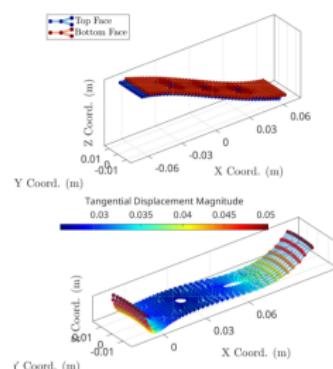
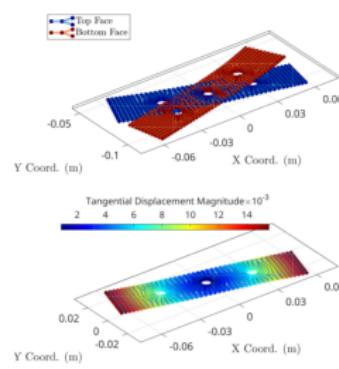
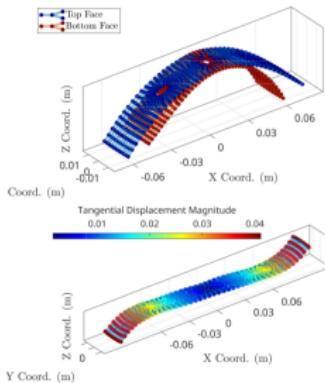
Relative Coordinates (RC) Relative DoFs are fixed. Not exactly HCB.

Numerical Comparison

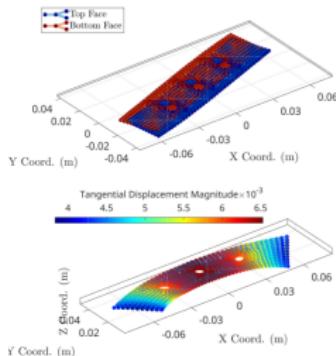
Type	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Time (s)
AC	143.74	151.30	575.08	642.81	870.59	39.86 s
RC	143.74	151.30	575.04	642.77	870.5	8.33 s

First Five Mode-Shapes At the Interface

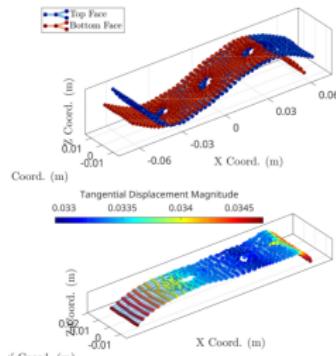
Mode 1



Mode 4



Mode 5



4. Outro and Recommendations

- When we're interested in working on friction modeling at the interfacial level, it makes sense to off-load everything else to a software.
- Try to think of ways to utilize the ABAQUS solvers as much as possible (for CMS modal analysis, etc.).

Some Drawbacks of this Pipeline

- Applicability is **strictly restricted to small displacement contact**.
- Geometrical Nonlinearities can't be present. If so, try an ICE fitting coupled with the relative coordinates approach.

We've GOT to Speak About Free and Open Source Software!

Outro and Recommendations

- ABAQUS is an impressive piece of software.
- MATLAB is great, too!

We've GOT to Speak About Free and Open Source Software!

Outro and Recommendations

- ABAQUS is an impressive piece of software.
- MATLAB is great, too!

Less Annoying Alternatives?

- Do check out [Julia!](#)
- Python has a great ecosystem too.



python™

We've GOT to Speak About Free and Open Source Software!

Outro and Recommendations

- ABAQUS is an impressive piece of software.

FOSS Alternatives

- Calculix: input very similar to ABAQUS, very practical.
- Code_Aster: Feature Rich!, super functional GUI front-end through Salome.



Less Annoying Alternatives?

- Do check out [Julia!](#)
- Python has a great ecosystem too.



Thank You!



*All the detailed instructions are hosted through Github in a repository named
[Nidish96/Abaqus4Joints](#)*

Comments, suggestions and contributions welcome!
Please email me at nidish@iitm.ac.in.