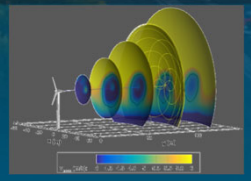
SAND2022-14504 TR

Sandia National Laboratories

NuMAD Software Training



PRESENTED BY

Ernesto Camarena, Evan Anderson, Ryan Clarke

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

What is NuMAD?

- Numerical Manufacturing And Design (NuMAD)
- Open-source blade design and analysis tool
- Developed by Sandia Labs over last two decades
- Specializes in 3D structural analyzes
  - enables panel buckling evaluations

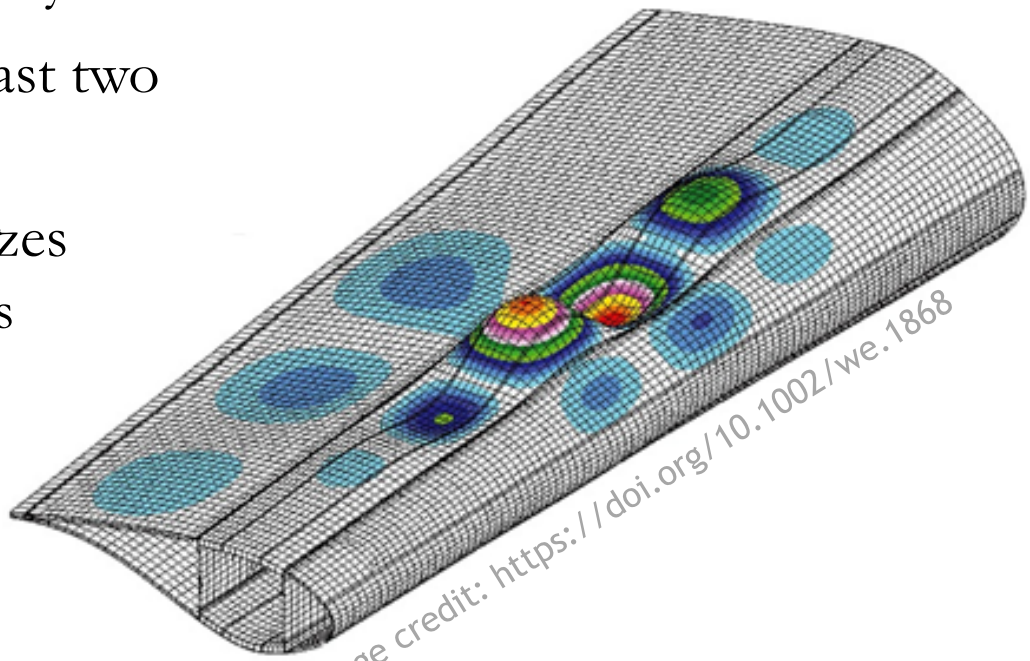
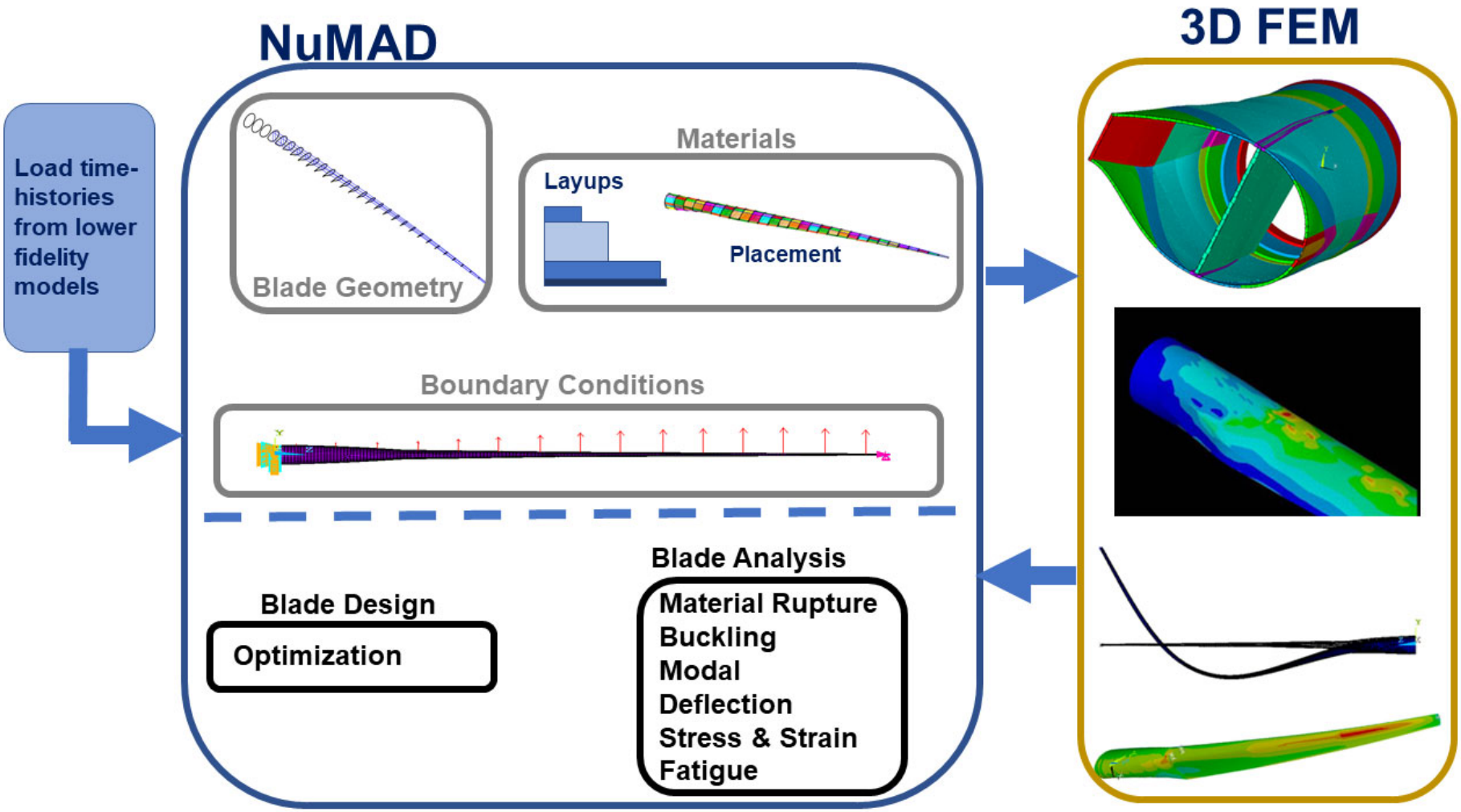


Image credit: https://doi.org/10.1002/we.1868

What is NuMAD? — Core Capabilities



What is NuMAD?

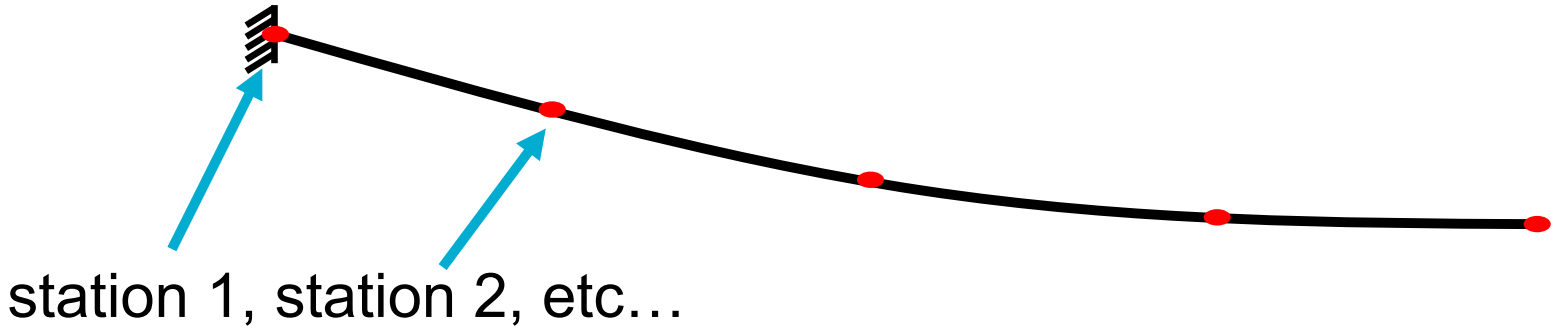
- Secondary Capabilities
  - Run and/or post-process NREL tools
    - PreComp, BModes, FASTv7, Crunch, Turbsim, IECWind, MBC
    - SNL Tools: BPE, BLAST
    - Plot3D file format for CFD meshing
- Code distribution
  - Formerly a static copy from energy.sandia.gov
  - Now on the Sandia Labs GitHub since January 4th, 2022
    - Code: https://github.com/sandialabs/NuMAD
    - Documentation: numad.readthedocs.io/

NuMAD Overview

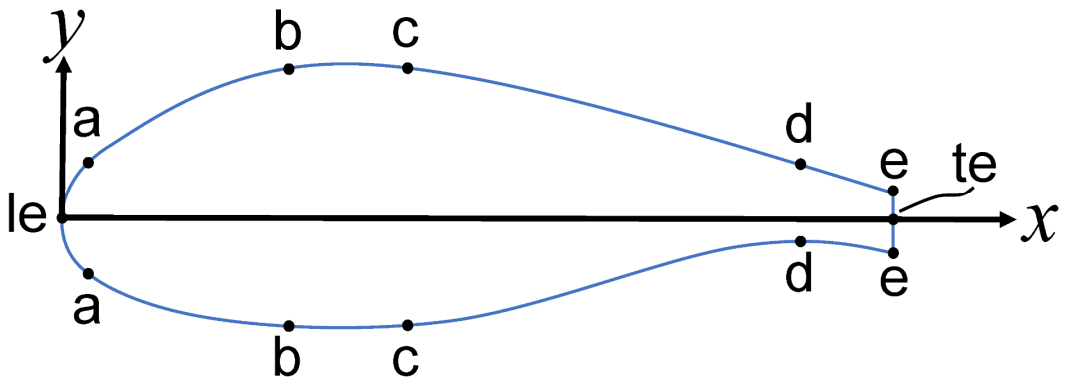
- Geometry
  - Stations & Keypoints
  - Blade Geomery
- Material Stacks
- The Blade Object
- Coordinate Systems
- Boundary Conditions

NuMAD Overview—Geometry: Stations & Keypoints

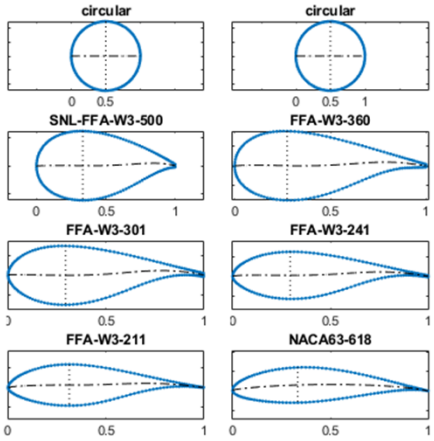
- "Stations" are positions along the span where cross-sectional data is known



- "Keypoints" partition the circumference of a cross-section where layup changes



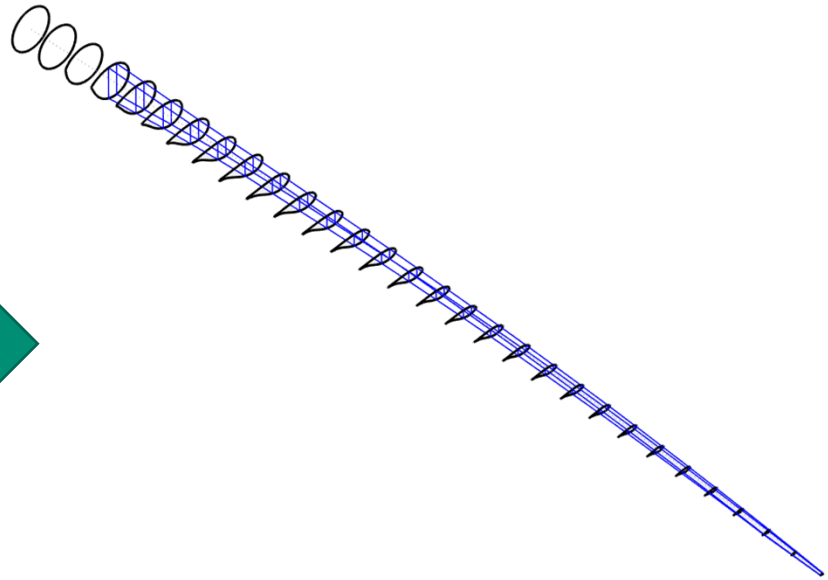
NuMAD Overview—Geometry: Blade Geometry



2D Non-dimensional airfoil coordinates

NuMAD

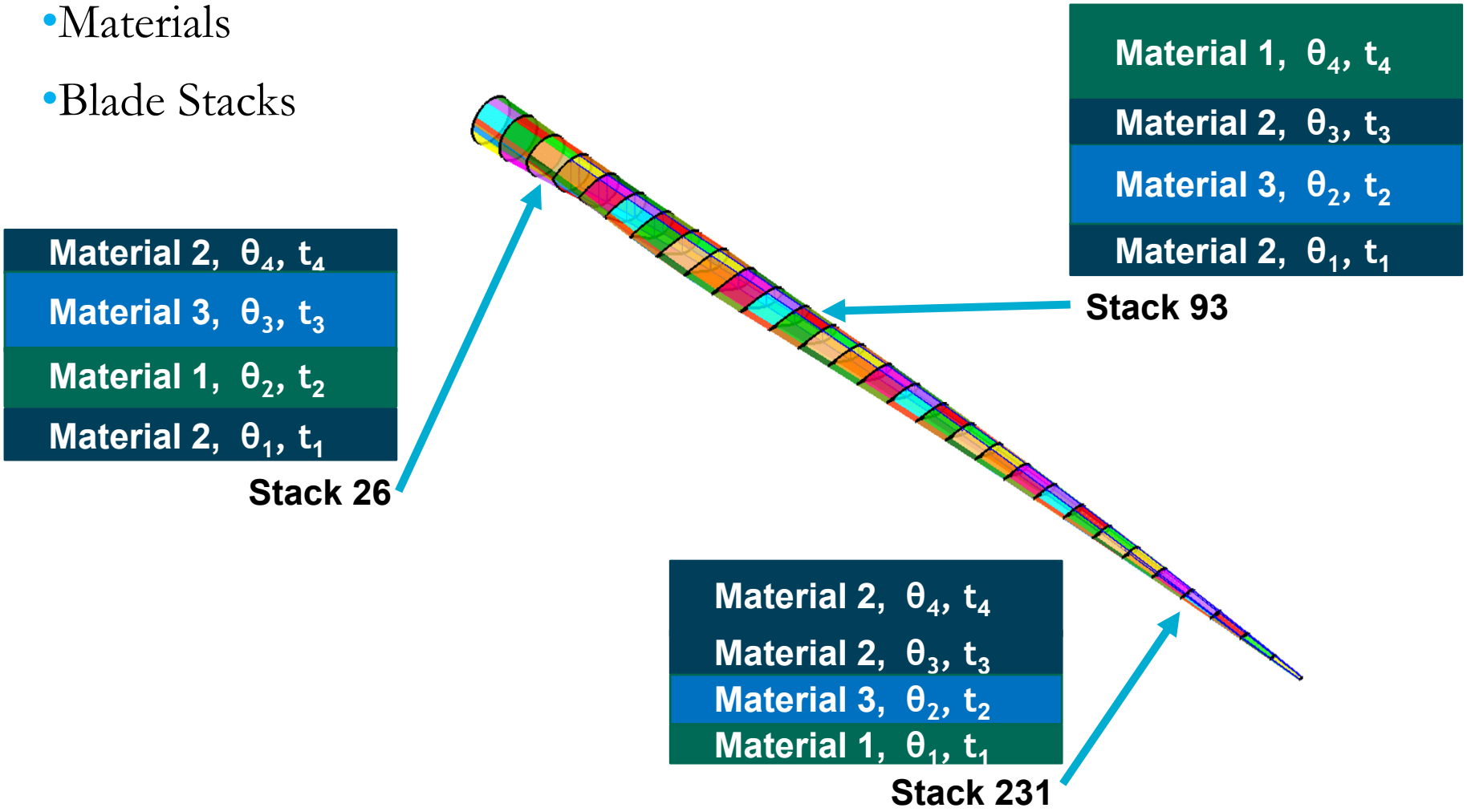
- Scale
- Translate
- Rotate
- Interpolate
- Webs



3D Blade Geometry

NuMAD Overview—Materials and Stacks

- Materials
- Blade Stacks



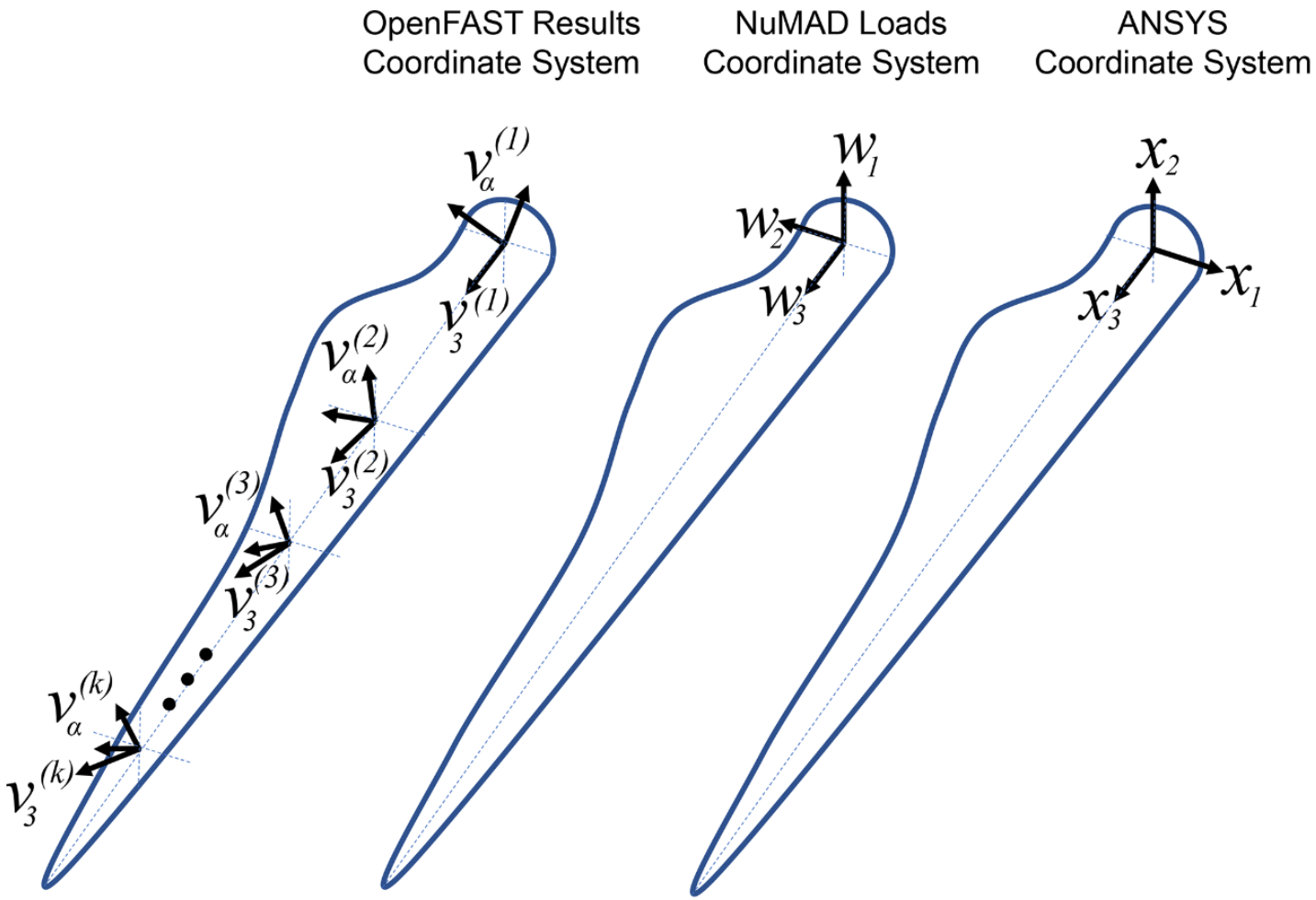
NuMAD Overview—The Blade Object

- In object-oriented programming, a class is like a template for a group of variables that an object always has
- Blade Class
- Stores mostly geometric and material data
- A blade object is a specific instance of the blade class with real values assigned to the variables

NuMAD's Blade Class

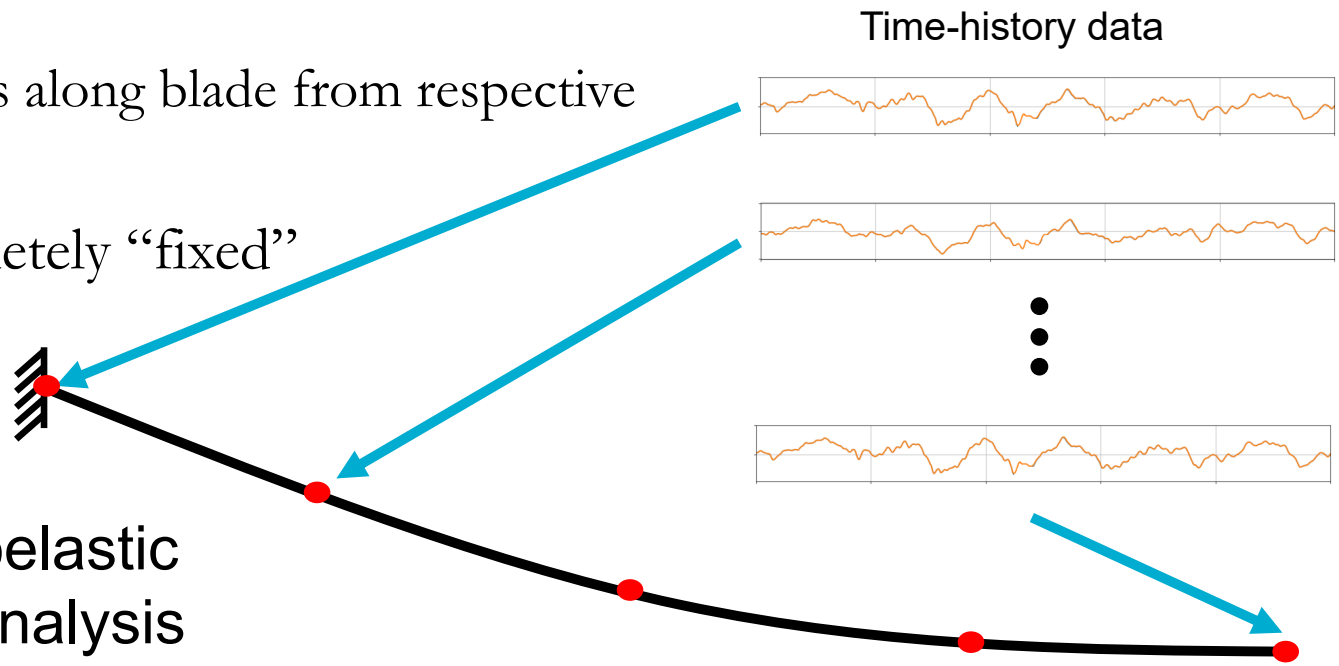
- Station locations
- Key points
- Materials
- Airfoils
- Spar cap widths
- Chord lengths
- etc…

NuMAD Overview—Coordinate Systems



NuMAD Overview—Boundary Conditions

- NuMAD reads time-history data from OpenFast
- Synthesizes static analysis from dynamic data
- Two Load Cases Categories
  - Blade deflection –moments along blade from a specific time
  - Blade failure – moments along blade from respective max/min
- DOFs at root are completely "fixed"



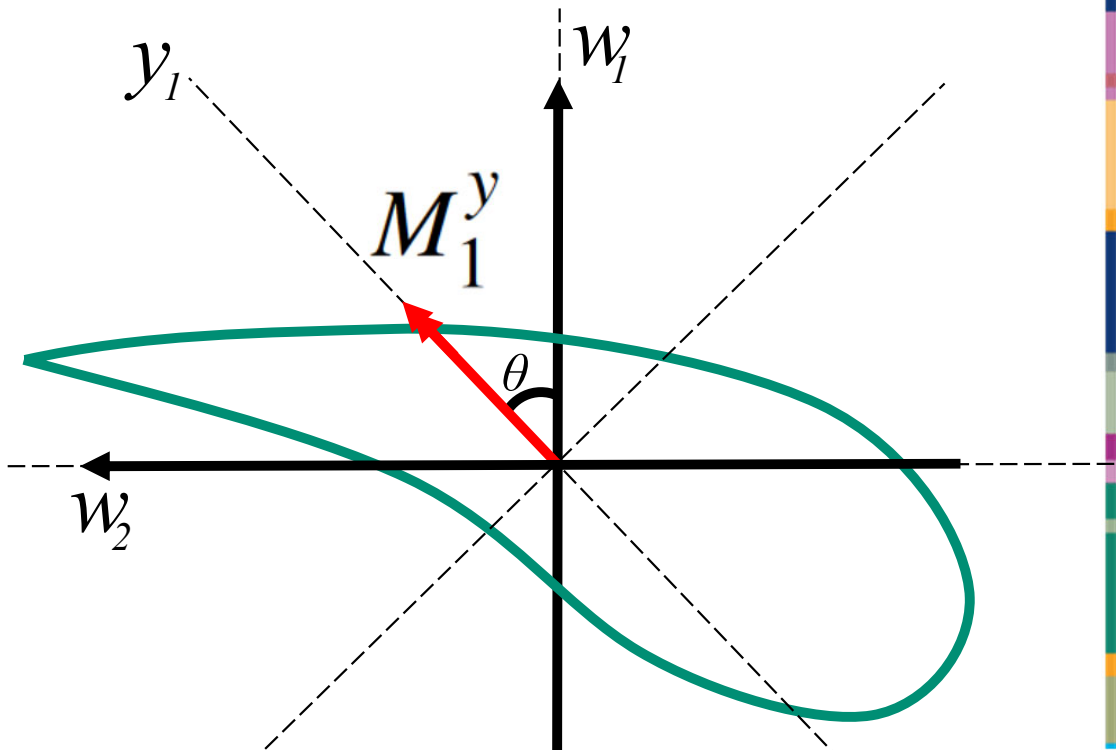
NuMAD Overview—Boundary Conditions

- Since approximating aeroelastic response with static analysis, IEC requires applying loads for the failure case along numerous directions.
- Beam loads at a given cross-section are:

$$\left\{ \begin{array}{l} \max\left(F_3^w(t)\right) \\ \max\left(M_1^y(t)\right)\cos(\theta) \\ \max\left(M_1^y(t)\right)\sin(\theta) \\ \max\left(M_3^w(t)\right) \end{array} \right\} \quad 0° \le \theta < 180°$$

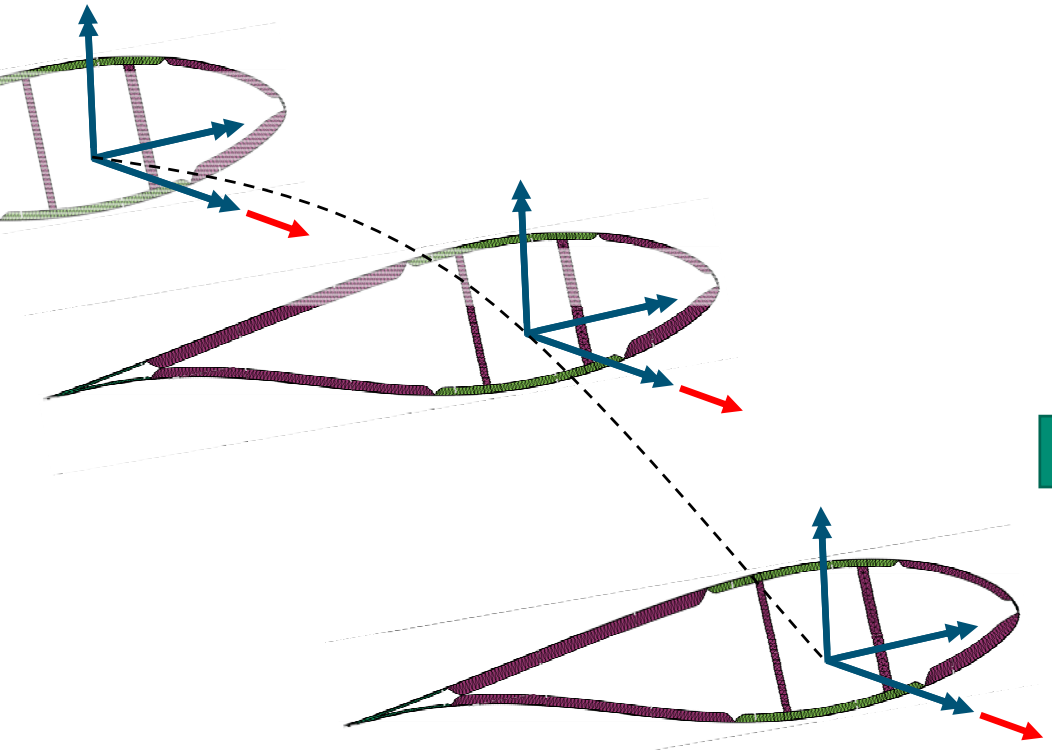
- `loadsTable` variable is a 1x8 array
- Each entry contains the loads for each analysis direction

$$M_1^y = M_1^w(t, w_3)\cos(\theta) + M_2^w(t, w_3)\sin(\theta)$$

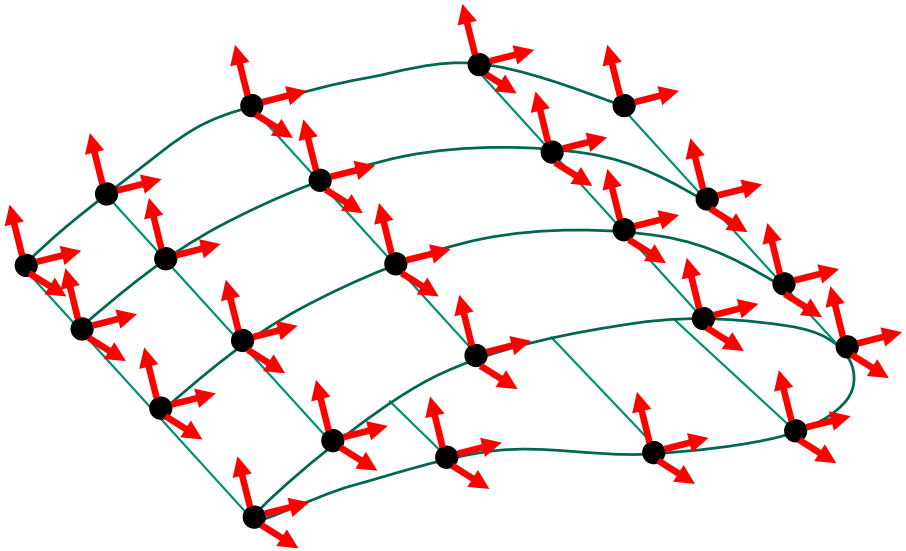


NuMAD Overview—Boundary Conditions

- Beam forces and moments along blade are converted to nodal forces



1D model, axial force and moments



3D model, forces only

Input formats

- Excel
- Yaml

Excel Input — General Layout

- 3 tabs needed
  - Materials
    - Elastic Properties, layer thickness, density, and strength properties.
  - Geometry
    - Airfoil shape, twist, chord dimensions, chord offset, aero center, etc.
  - Components
    - Brings materials and geometry together.
    - Specify spanwise and chordwise distributions of materials.
- I would start the build of any new blade using Excel2Object example as a base.
- xlsblade.m in the source code is where the excel data is read into the blade object.

Excel Input — Material Tab

- Order matters in specification of materials, outer -> inner materials. i.e. Gelcoat always goes first.

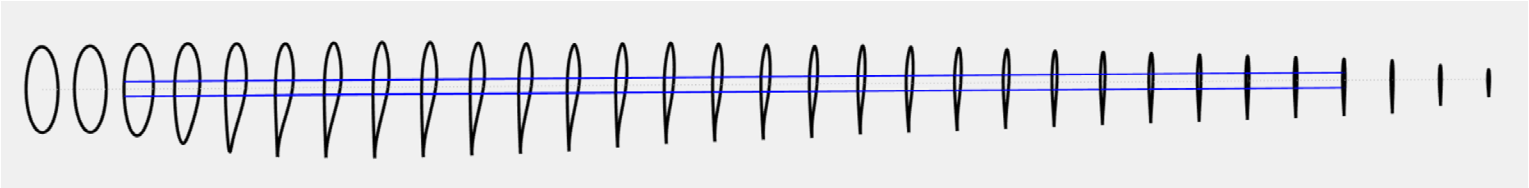
- If Ez is left blank for orthotropic material type then it will automatically be set to equal Ey, same with Gyz, Gxz and pryz, prxz.

Excel Input — Geometry Tab: Natural Offset Flag

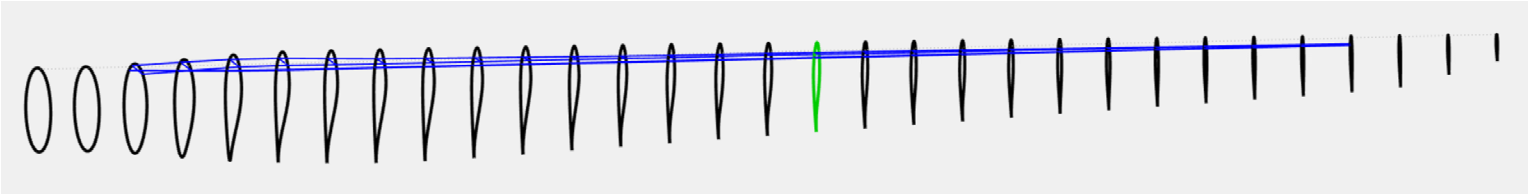
Natural Offset: True - blade reference axis is offset relative to max airfoil thickness

Natural Offset: False - blade reference axis is offset relative to airfoil leading edge

Natural Offset: True

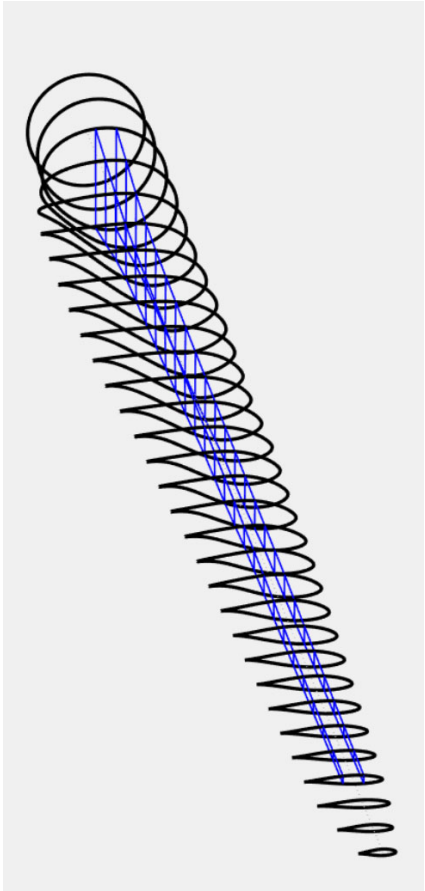


Natural Offset: False

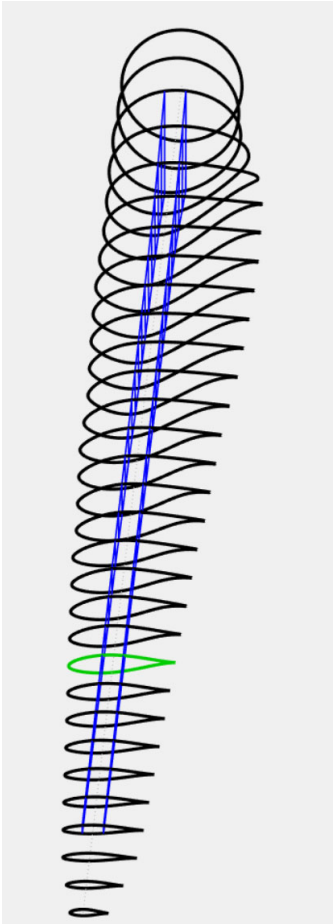


Excel Input — Geometry Tab: Rotor Spin Flag

Clockwise (CW) rotor spin



Counter-clockwise (CCW) rotor spin

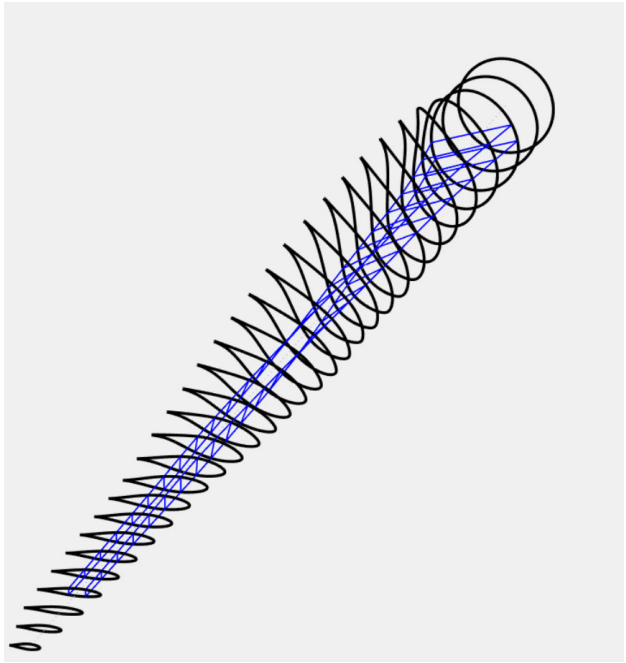


Excel Input — Geometry Tab: Shear Web Twist

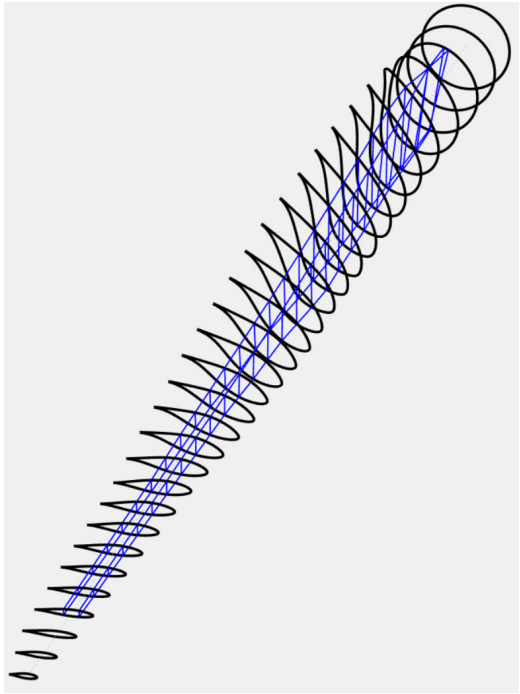
Shear Web Twist: True – Shear webs will twist with the blade.

Shear Web Twist: False – Planar shear webs.

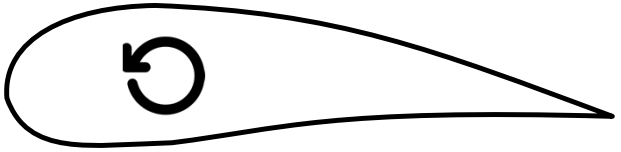
Shear Web Twist: **True**



Shear Web Twist: **False**



Excel Input — Geometry Tab

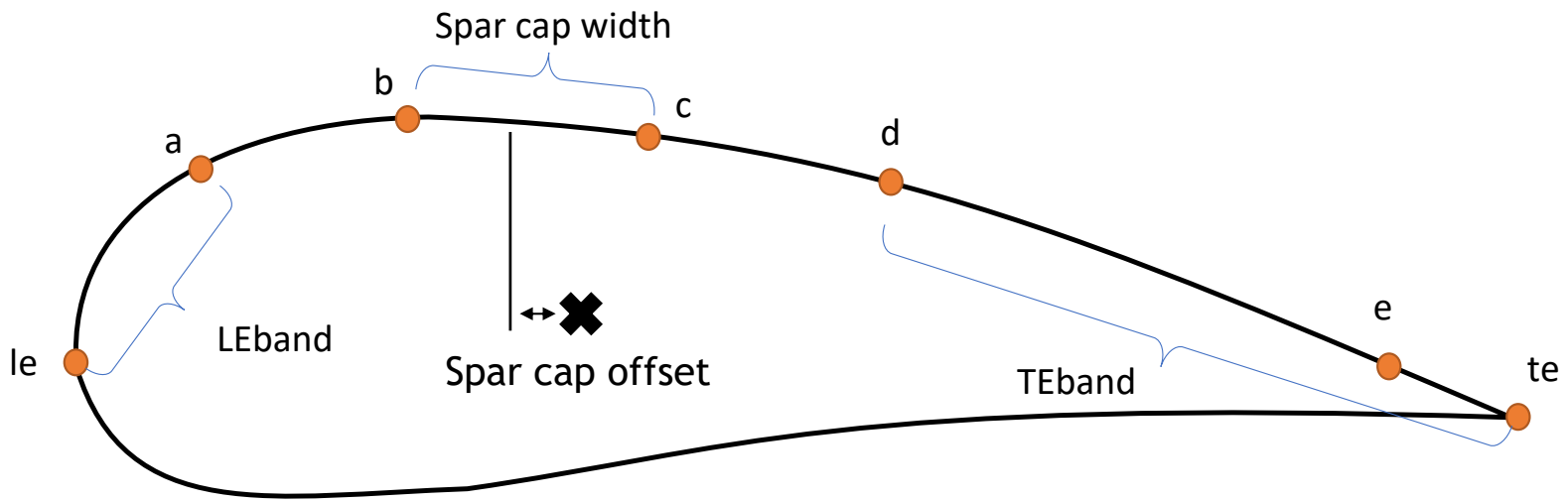


- Direction of twist: counter-clockwise while looking at airfoil with LE to the left and TE to the right.
- X-offset: blade reference axis is either offset from max thickness of airfoil or from LE depending on flag.
- If twist, chord, % thick., offset, and aero center have gaps in specified data, it will be interpolated.
- There are 3 span columns in geometry, the first one corresponds to values such as twist, chord, aero center, etc. the second corresponds to the spanwise position of each airfoil, and the third specifies how many final interpolated stations is desired.

Excel Input — Component Tab: Component Dimensions

- Spar cap width can be one number so both spar caps are equal or two i.e. [400, 300] mm. The High pressure side spar cap is the first entry always.
- LE Band width denotes the length of leading edge reinforcement.
- TE Band width denotes the length of trailing edge reinforcement.
- Spar caps are centered along the blade reference line.
- Spar cap offset, offsets the spar cap midpoint relative to the blades reference axis. Positive spar cap offset moves center of spar caps towards trailing edge.

Excel Input — Component Tab: Airfoil Station Definition



Keypoints:

✖ - Blade Reference axis

a - arclength of leading edge reinforcement; no greater than 10% arclength, no less than 1% arclength.

b - spar cap center minus half of the spar cap width; no less than 15% arclength.

c - spar cap center plus half of the spar cap width; no greater than 80% arclength.

d - arclength of trailing edge reinforcement; no less than 85% arclength, no greater than 98% arclength.

e - total arclength if flatback airfoil; 99% arclength otherwise

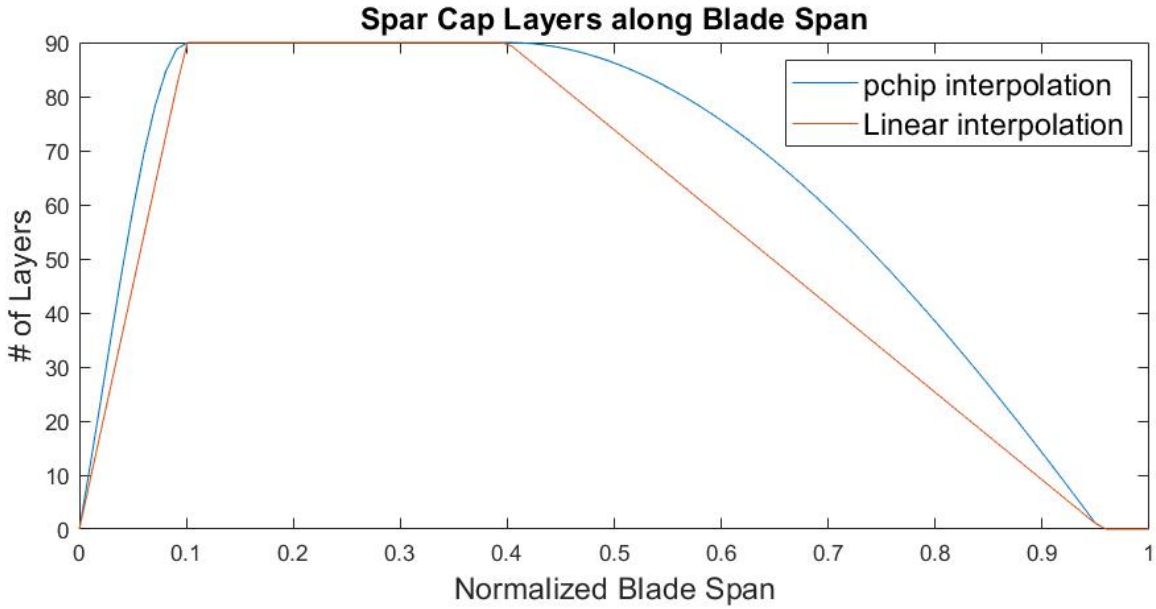
Excel Input — Component Tab: Groups, HP and LP Extends

- Component group 0 is always the blade shell, 1,2,3, etc. denote shear webs.

- HPExtends and LPExtends are used to define cross-section width of each component. i.e. HPExtends = [le, te] for component Gelcoat; denotes the gelcoat extends the entire length of the airfoils high pressure side

Excel Input — Component Tab: CP span, CP nLayers, and Layer Interp

- CP span denotes at what point along the normalized blade span layers of the component are called out.
- CP nlayers denotes number of layers at that blade span
- Layer Interp denotes what interpolation scheme is used between CP points.



Example: Spar CP span = [0, 0.1, 0.4, 0.95]; CP nLayers = [0, 90, 90, 1]; Layer Interp = pchip and linear

Excel Input — Limitations and Future Work

- Blade Object allows for additional material strength and failure data beyond UTS and UCS, however this needs to be added in separate the excel file for now.
- Prebend and presweep currently can't be specified from an excel file.
- Spar cap width cannot change along span in excel format.

YAML Input — formatting

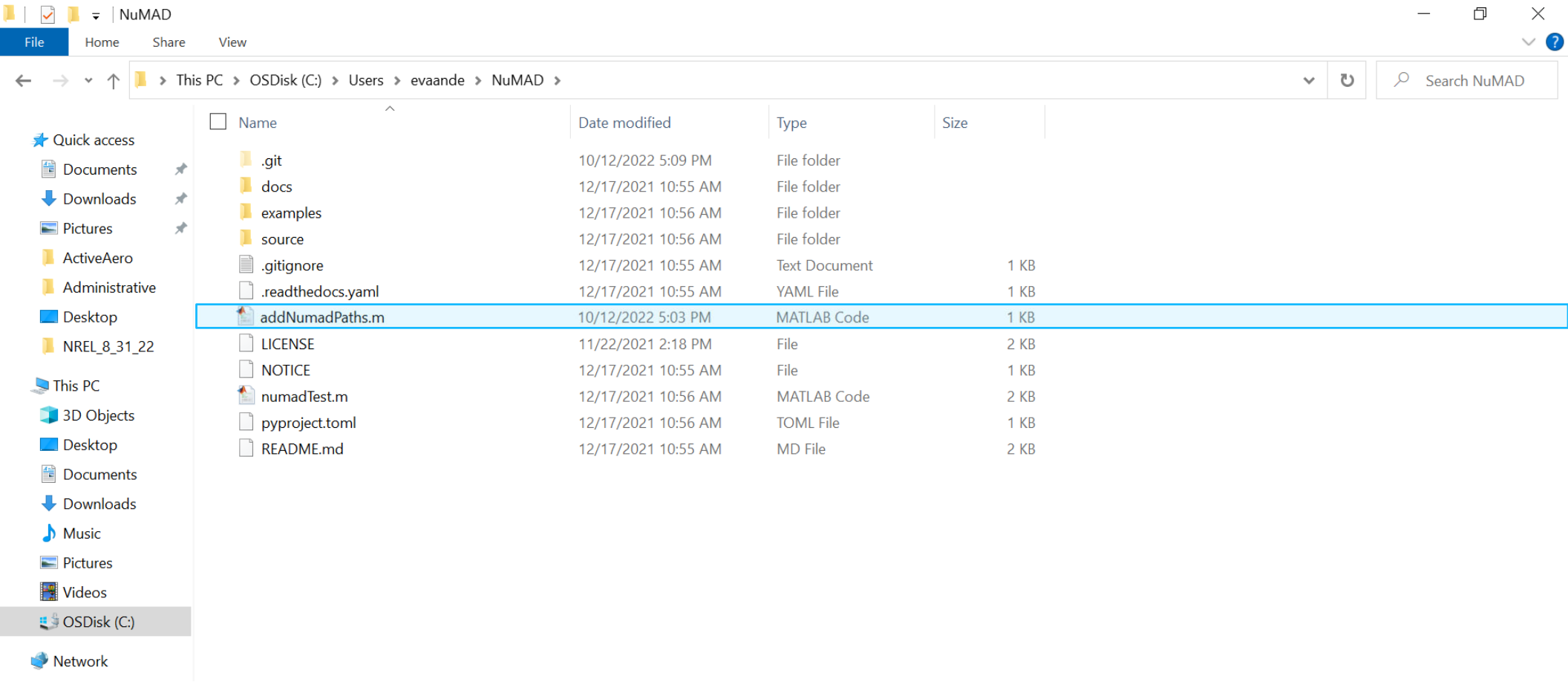
- A standardized type of input file that eases collaboration across various codes
- YAML-schema of a wind turbine and plant ontology developed within IEA Task 37 for systems engineering MDAO called WindIO.
- NuMAD reads and modifies only the turbine ontology
- For more detailed information:
  - https://windio.readthedocs.io/en/latest/index.html
  - https://iea-wind.org/task37/
  - https://www.nrel.gov/docs/fy22osti/82621.pdf

Wind turbine YAML for NuMAD

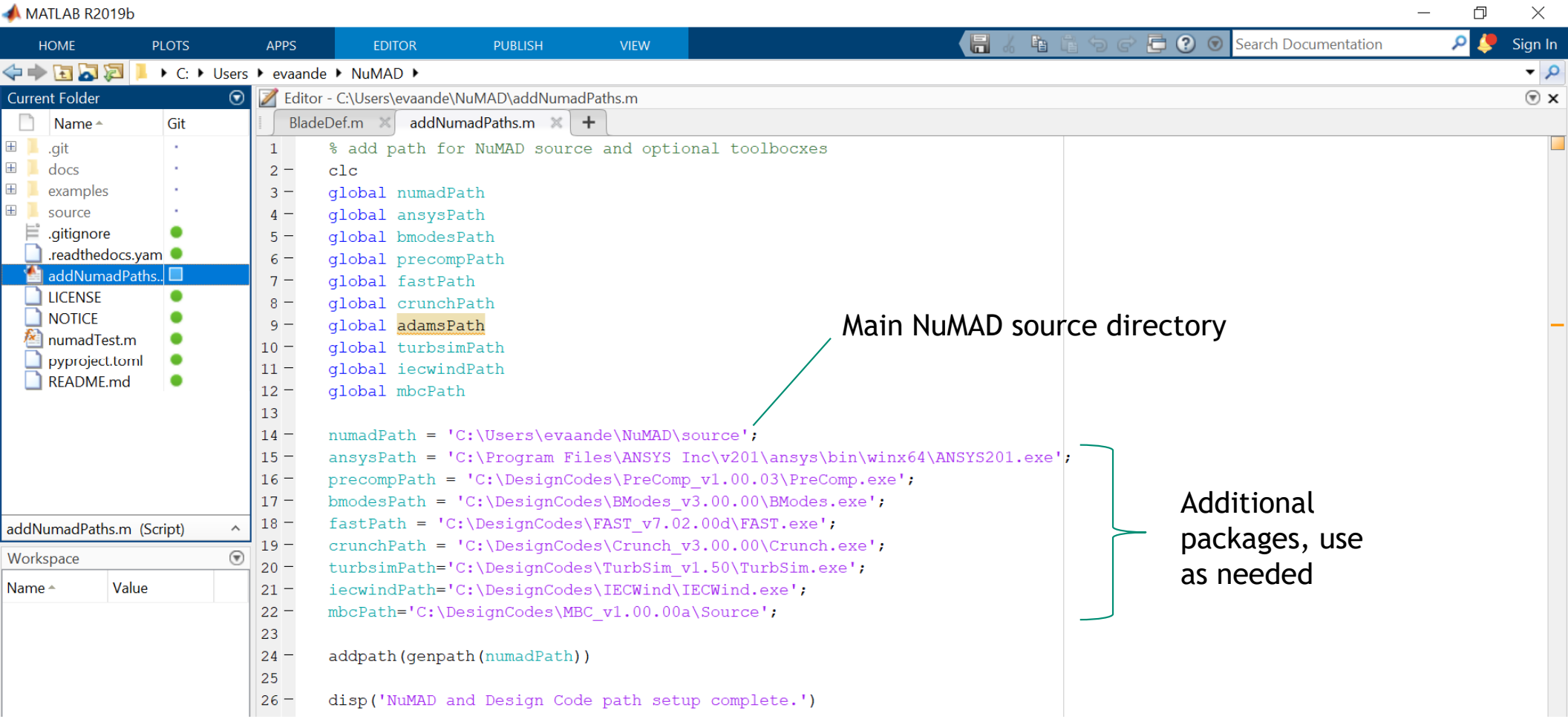
- NuMAD is only concerned with the Blade component within the turbine YAML.
- WindIO format is consistent with OpenFAST and Beamdyn's coordinate system for how the blade is constructed.
- NuMAD automatically converts necessary blade definition information for the blade object into NuMADs coordinate system.

Setting up your local directories/paths

In the main NuMAD directory, open addNumadPaths.m

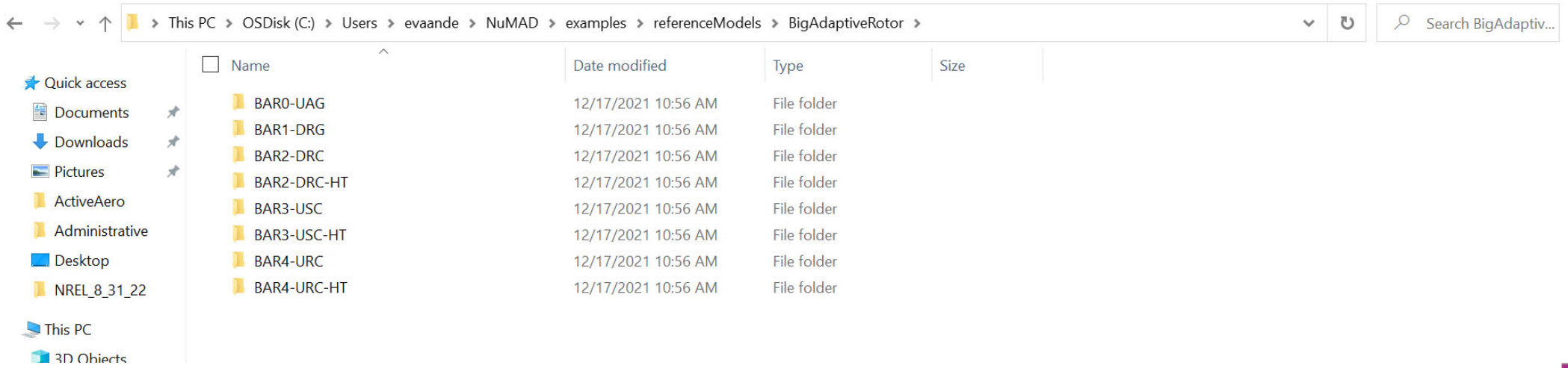


Setting up your local directories/paths



The .yaml file

Example .yaml files can be found in NuMAD\examples\referenceModels\BigAdaptiveRotor



The .yaml file

Definition of outer mold line:

```yaml
name: BAR0
description: Stiff upwind design - not transportable - Spar caps with industry baseline GFRP - last updated on August 20th 2020 by Pietro F
assembly: {turbine_class: III, turbulence_class: A, drivetrain: Constant_eff, rotor_orientation: Upwind, number_of_blades: 3, rotor_diamete
components:
    blade:
        outer_shape_bem:
            airfoil_position:
                grid: [0.0, 0.02, 0.18, 0.25, 0.35, 0.524, 0.799, 1.0]
                labels: [circular, circular, SNL-FFA-W3-500, FFA-W3-360, FFA-W3-301, FFA-W3-241, FFA-W3-211, NACA63-618]
            chord:
                grid: [0.0, 0.034482758620689655, 0.06896551724137931, 0.10344827586206896, 0.13793103448275862, 0.1724137931034483, 0.2068
                values: [4.5, 4.516534551546473, 4.592554552493484, 4.720268647024349, 4.878450369199802, 5.0423597204468615, 5.1888039720
            twist: &id002
                grid: [0.0, 0.034482758620689655, 0.06896551724137931, 0.10344827586206896, 0.13793103448275862, 0.1724137931034483, 0.2068
                values: [0.3491, 0.3241639295566382, 0.2978904216627936, 0.27036560235093493, 0.24167559765353058, 0.20775991808748892, 0.1
            pitch_axis:
                grid: [0.0, 0.034482758620689655, 0.06896551724137931, 0.10344827586206896, 0.13793103448275862, 0.1724137931034483, 0.2068
                values: [0.5, 0.4804815083118872, 0.45368069201011907, 0.42215292213980427, 0.3901686299260218, 0.3610978079524211, 0.3365
            reference_axis: &id001
                x:
                    grid: [0.0, 0.034482758620689655, 0.06896551724137931, 0.10344827586206896, 0.13793103448275862, 0.1724137931034483, 0.
                    values: [0.0, 0.01458960863306122, 0.05222307680765803, 0.10814189079154829, 0.17425050028042421, 0.24113677336298817,
                y:
                    grid: [0.0, 0.034482758620689655, 0.06896551724137931, 0.10344827586206896, 0.13793103448275862, 0.1724137931034483, 0.
                    values: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                z:
                    grid: [0.0, 0.034482758620689655, 0.06896551724137931, 0.10344827586206896, 0.13793103448275862, 0.1724137931034483, 0.
                    values: [0.0, 3.4481471658071863, 6.8962943316143726, 10.34444149742156, 13.792588663228745, 17.240735829035934, 20.688
        internal_structure_2d_fem:
            reference_axis: *id001
            webs:
              - name: fore_web
                rotation:
```

The .yaml file

Airfoil definitions:

```yaml
airfoils:
  - name: circular
    coordinates:
      x: [1.0, 0.99901, 0.99606, 0.99114, 0.98429, 0.97553, 0.96489, 0.95241, 0.93815, 0.92216, 0.90451, 0.88526, 0.86448, 0.84227, 0.8
      y: [0.0, 0.0314, 0.06267, 0.09369, 0.12434, 0.15451, 0.18406, 0.21289, 0.24088, 0.26791, 0.29389, 0.31871, 0.34227, 0.36448, 0.38
    relative_thickness: 1.0
    aerodynamic_center: 0.5
    polars:
      - configuration: Default
        re: 3000000.0
        c_m:
          grid: [-3.141592653589793, 3.141592653589793]
          values: [0.0, 0.0]
        c_l:
          grid: [-3.141592653589793, 3.141592653589793]
          values: [0.0, 0.0]
        c_d:
          grid: [-3.141592653589793, 3.141592653589793]
          values: [0.5, 0.5]
  - name: SNL-FFA-W3-500
    coordinates:
      x: [1.0, 0.99944304, 0.99812049, 0.99569352, 0.99230484, 0.98802844, 0.98281508, 0.97666422, 0.96964069, 0.96174313, 0.95297315,
      y: [0.0105, 0.01086983, 0.0117452, 0.01334146, 0.01554892, 0.01830045, 0.02160574, 0.02544031, 0.02973944, 0.03448209, 0.0396488
    relative_thickness: 0.5
    aerodynamic_center: 0.316
    polars:
      - configuration: Default
        re: 5280000.0
        c_l:
          grid: &id004 [-3.141592653589793, -2.967, -2.793, -2.618, -2.443, -2.269, -2.094, -1.92, -1.745, -1.571, -1.396, -1.222, -1
          values: [0.0, 0.40714078, 0.81289953, 0.93008162, 0.80570788, 0.68718404, 0.54494169, 0.37656041, 0.18883741, 0.00021815,
        c_d:
```

The .yaml file

Definition of components:

```yaml
layers:
  - name: UV_protection
    material: Gelcoat
    thickness:
        grid: [0.0, 0.034482758620689655, 0.06896551724137931, 0.10344827586206896, 0.13793103448275862, 0.1724137931034483, 0.2068965
        values: [0.0005, 0.0005, 0.0005, 0.0005, 0.0005, 0.0005, 0.0005, 0.0005, 0.0005, 0.0005, 0.0005, 0.0005, 0.0005, 0.0005, 0.000
    start_nd_arc:
        grid: [0.0, 0.034482758620689655, 0.06896551724137931, 0.10344827586206896, 0.13793103448275862, 0.1724137931034483, 0.2068965
        values: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
    end_nd_arc:
        grid: [0.0, 0.034482758620689655, 0.06896551724137931, 0.10344827586206896, 0.13793103448275862, 0.1724137931034483, 0.2068965
        values: [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.
    fiber_orientation:
        grid: [0.0, 0.034482758620689655, 0.06896551724137931, 0.10344827586206896, 0.13793103448275862, 0.1724137931034483, 0.2068965
        values: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
  - name: Shell_skin_outer
    material: glass_triax
    thickness:
        grid: [0.0, 0.034482758620689655, 0.06896551724137931, 0.10344827586206896, 0.13793103448275862, 0.1724137931034483, 0.2068965
        values: [0.053187711, 0.044341438, 0.035877651999999996, 0.028118241000000002, 0.021283717, 0.015571871, 0.011093277, 0.007860
    start_nd_arc:
        grid: [0.0, 0.034482758620689655, 0.06896551724137931, 0.10344827586206896, 0.13793103448275862, 0.1724137931034483, 0.2068965
        values: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
    end_nd_arc:
        grid: [0.0, 0.034482758620689655, 0.06896551724137931, 0.10344827586206896, 0.13793103448275862, 0.1724137931034483, 0.2068965
        values: [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.
    fiber_orientation:
        grid: [0.0, 0.034482758620689655, 0.06896551724137931, 0.10344827586206896, 0.13793103448275862, 0.1724137931034483, 0.2068965
        values: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
  - name: Spar_cap_ss
    material: glass_uni
    thickness:
```

The .yaml file

Definition of materials

```yaml
materials:
  - {name: Gelcoat, orth: 0.0, rho: 1235.0, E: 3440000000.0, G: 1323000000.0, nu: 0.3, alpha: 0.0, Xt: 1e10, Xc: 1e10, S: 1e10, GIc: 41.(
  - {name: Adhesive, orth: 0.0, rho: 1100.0, E: 4560000.0, G: 1450000.0, nu: 0.3, alpha: 0.0, Xt: 61510000.0, Xc: 65360000.0, S: 3661000(
  - name: glass_uni
    description: Vectorply E-LT-5500, Epikote MGS RIMR 135/Epicure MGS RIMH 1366 epoxy
    source: MSU composites database 3D property tests
    orth: 1.0
    rho: 1940.0
    E: [43700000000.0, 16500000000.0, 15450000000.0]
    G: [3265000000.0, 3495000000.0, 3480000000.0]
    nu: [0.262, 0.264, 0.35]
    Xt: [640230000.0, 38100000.0, 0.0]
    Xc: [370690000.0, 82180000.0, 0]
    S: [30170000.0, 18970000.0, 6210000.0]
    m: 10
    GIc: 61.0
    GIIc: 101.0
    alp0: 50.0
    fvf: 0.57
    fwf: 0.7450682696347697
    ply_t: 0.005
    unit_cost: 1.87
    waste: 0.05
    fiber_density: 2535.5
    area_density_dry: 7.227162215457267
    component_id: 4
  - name: glass_biax
    description: Vectorply E-LT-5500, Epikote MGS RIMR 135/Epicure MGS RIMH 1366 epoxy
    source: Vectorply E-LT-5500, Hybrid Rule of Mixtures
    orth: 1.0
    rho: 1940.0
    E: [11023100000.0, 11023100000.0, 16047700000.0]
```

The .yaml file

The command to read the contents of a yaml file into the workspace is a method in the BladeDef class. Usage:

Create an instance of BladeDef:

```
blade = BladeDef;
```

Set the name of the .yaml file:

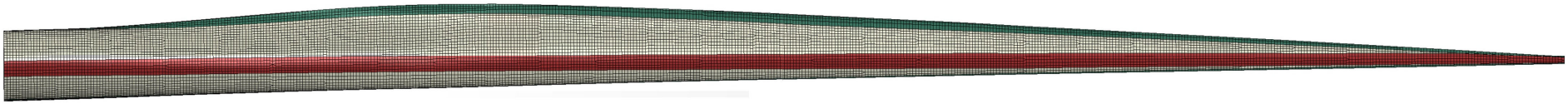
```
fileName = 'myBlade.yaml'
```

Read the contents of the .yaml file into blade object:

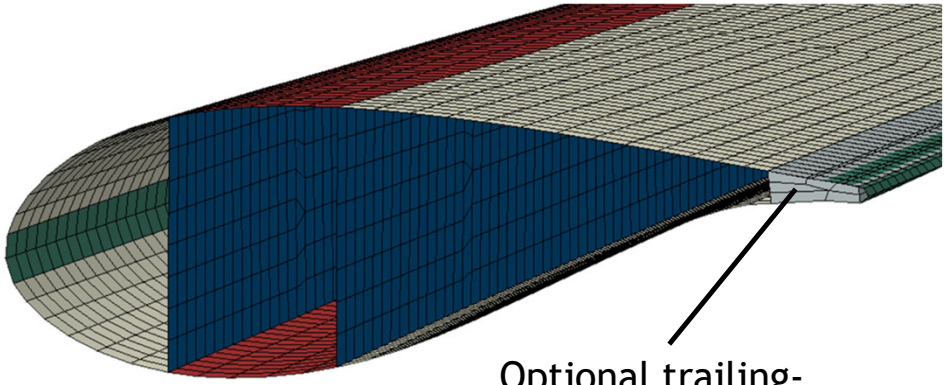
```
blade.readYAML(fileName);
```

Generating meshes

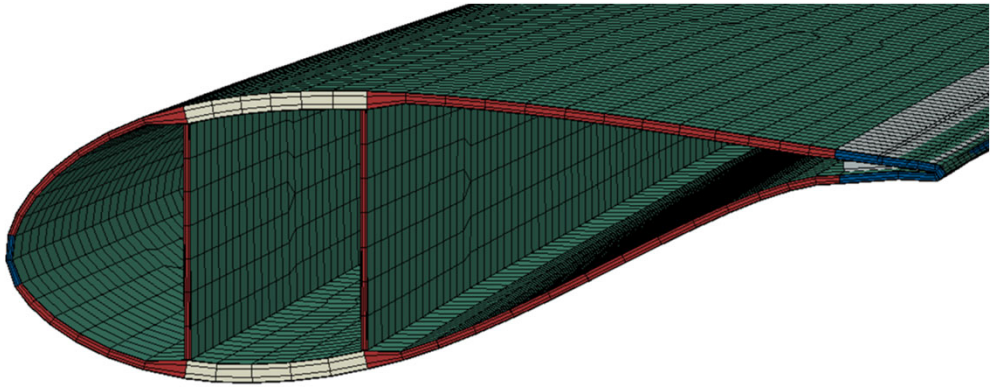
NuMAD can now generate full-blade shell element and solid element meshes in-house.



Shell meshes:



Solid meshes:



Generating meshes

The commands to shell blade meshes in-house are methods in the BladeDef class. Usage:

Set the option whether to include trailing-edge adhesive in the shell model (0 for no, 1 for yes):

```
adhes = 0;
```

Generate the shell model mesh:

```
[nodes,elements,OSSets,SWSets,adNds,adEls] = blade.getShellMesh(adhes);
```

Outputs:

- nodes: coordinates of all shell nodes
- elements: element connectivity of all shell elements
- OSSets: array of objects of the elementSet class storing element sets in the outer shell
- SWSets: array of objects of the elementSet class storing element sets in the shear webs
- adNds: coordinates of solid adhesive nodes if applicable
- adEls: element connectivity of solid adhesive nodes if applicable

Generating meshes

The commands to solid blade meshes in-house are methods in the BladeDef class. Usage:

Set the number of elements through the outer, middle and inner layers of the blade for the solid model:

```
layerNumEls = [1,3,1];
```

Generate the solid model mesh:

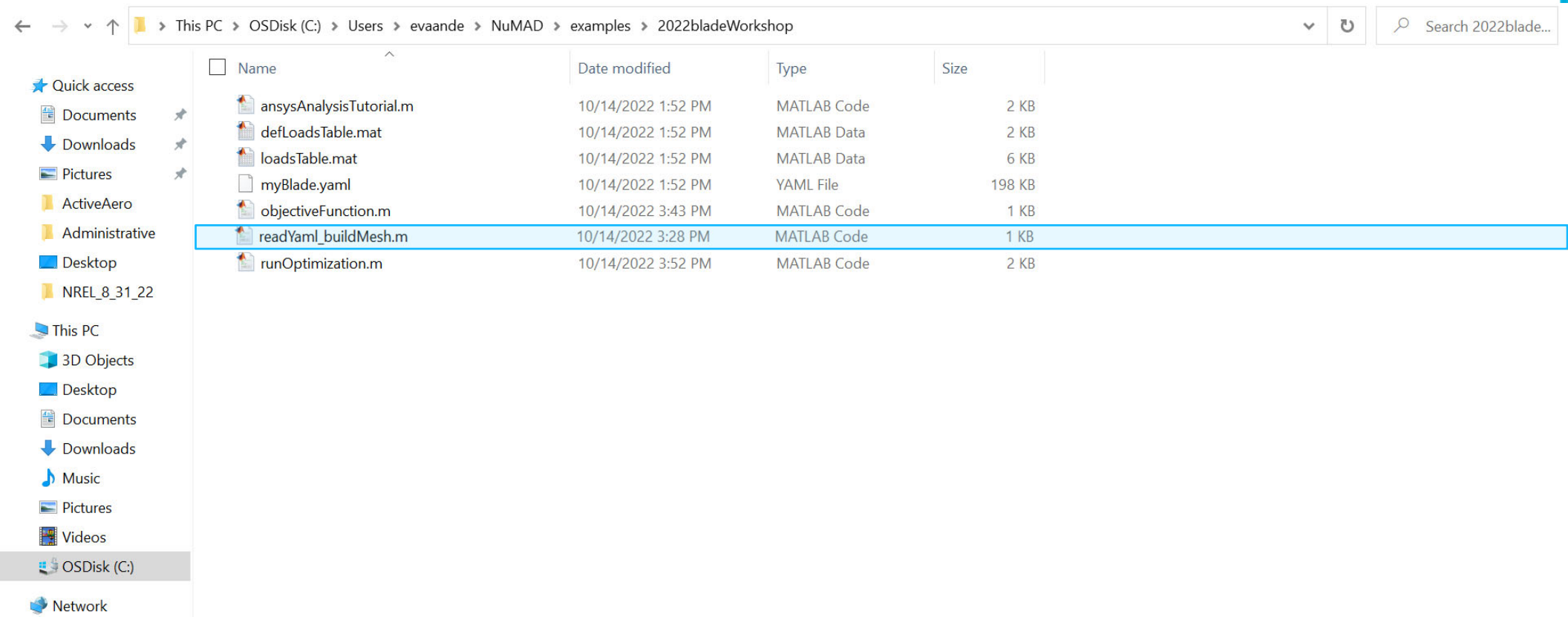
```
[nodes,elements,OSSets,SWSets,adSet] = blade.getSolidMesh(layerNumEls);
```

Outputs:

- nodes: coordinates of all nodes
- elements: element connectivity of all elements
- OSSets: array of objects of the elementSet class storing element sets in the outer shell
- SWSets: array of objects of the elementSet class storing element sets in the shear webs
- adSet: a single object of the elementSet class storing the elements in the adhesive

Example Script: readYaml_buildMesh.m

NuMAD\examples\2022bladeWorkshop\readYaml_buildMesh.m



Example Script: readYaml_buildMesh.m

Editor - C:\Users\evaande\NuMAD\examples\2022bladeWorkshop\readYaml_buildMesh.m

readYaml_buildMesh.m

```
1   %% Add paths to NuMAD source
2   cd('..\..');
3   addNumadPaths;
4
5   %% Return to the example directory where the needed files are located
6   cd('examples\2022bladeWorkshop');
7
8   %% Create a blade object as an instance of BladeDef and read in data from the .yaml file
9
10  blade = BladeDef;
11  fileName = 'myBlade.yaml';
12  blade.readYAML(fileName);
13
14  %% Set the global element size for the shell mesh
15
16  blade.mesh = 0.2;
17
18  %% Generate the shell mesh
19
20  adhes = 1;
21  [nodes,elements,OSSets,SWSets,adNds,adEls] = blade.getShellMesh(adhes);
22
```

Example Script: readYaml_buildMesh.m

Editor - C:\Users\evaande\NuMAD\examples\2022bladeWorkshop\readYaml_buildMesh.m

readYaml_buildMesh.m

```
23    %% Print first 10 nodes coordinates
24
25 -  nodes(1:10,:)
26
27    %% Print first 10 element connectivities
28
29 -  elements(1:10,:)
30
31    %% Print first 10 elements in section 4, spanwise segment 10 (pressure side spar cap)
32
33 -  OSSets(4,10).elementList(1:10)
34
35    %% Print first 10 elements in spanwise segment 10 of first shear web
36
37 -  SWSets{1}(10).elementList(1:10)
38
39
40
```

Generating loads

The fundamental structure for defining the applied loads on a blade for structural analysis is the loads table, which has the following internal structure:

```
>> defLoadsTable{1}

ans =

  struct with fields:

      input: [1×1 struct]
     rBlade: [1×20 double]
        Fxb: [1×20 double]
        Fyb: [1×20 double]
        Fzb: [1×20 double]
        Mxb: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
        Myb: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
        Mzb: [1×20 double]
      Alpha: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
   prebend: [1×20 double]
  presweep: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

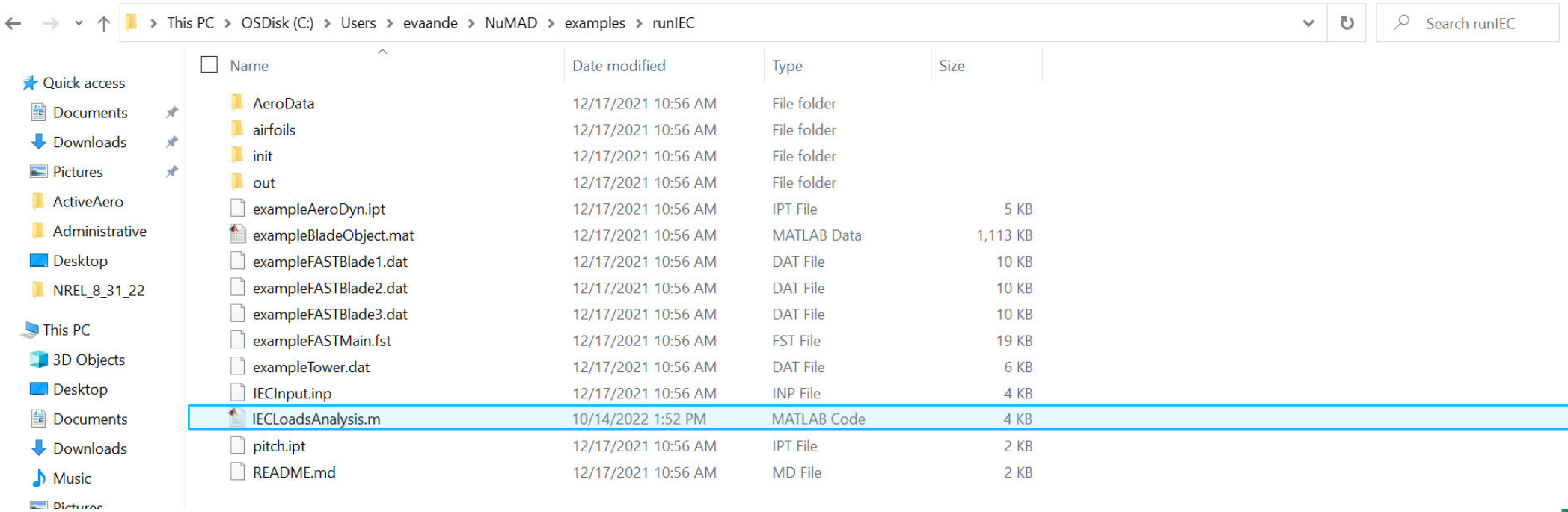
Spanwise position of the load arrays that follow

Spanwise distribution of forces along the blade

Spanwise distribution of moments along the blade

Generating loads

NuMAD\examples\runIEC\IECLoadsAnalysis.m ***Note: to be phased out soon**



Main FEA Analysis Script

`mainAnsysAnalys.m`

- blade object
- loads data
- mesh data
- analyses selections

Applies nodal forces

Writes selected analyses based on configuration variable

Also instructs ANSYS to write QOIs to text files

Issues system call to have ANSYS run the built text file

Reads results from ANSYS generated text files

Assigns read results to MATLAB variable

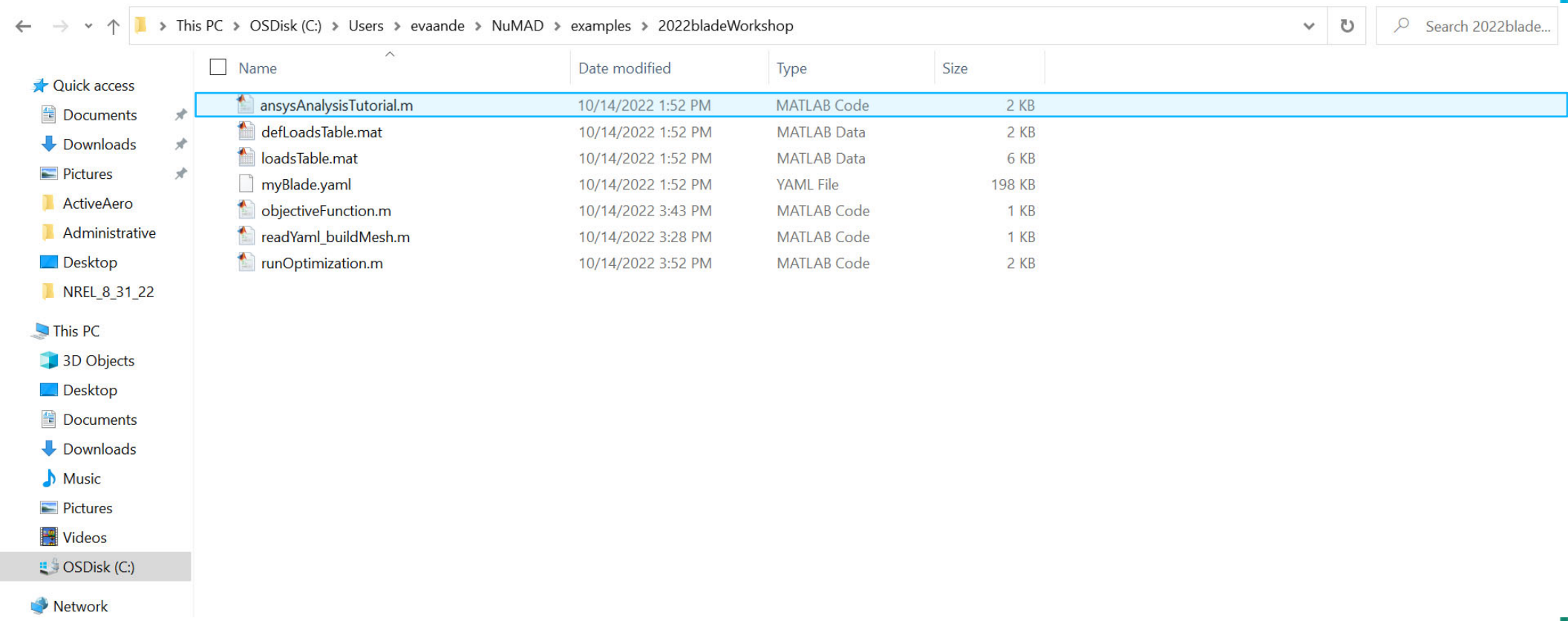
QOIs in MATLAB variable

Blade Analysis

- Material Rupture
- Buckling
- Modal
- Deflection
- Stress & Strain
- Fatigue

Example Script: ansysAnalysisTutorial.m

NuMAD\examples\2022bladeWorkshop\ansysAnalysisTutorial.m



Demonstration

```
%% create a shell model in ansys w/o adhesive
includeAdhesive=0;
meshData=blade.generateShellModel('ansys',includeAdhesive);

meshData =

  struct with fields:

                nodes: [7073×3 double]
             elements: [7480×4 double]
      outerShellElSets: [12×29 elementSet]
        shearWebElSets: {[1×29 elementSet]  [1×29 elementSet]}
             adhesNds: []
             adhesEls: []
```

Demonstration

```
%% Load a previously build loadsTable
load('defLoadsTable.mat')

defLoadsTable =

  1×1 cell array

    {1×1 struct}
```

Demonstration

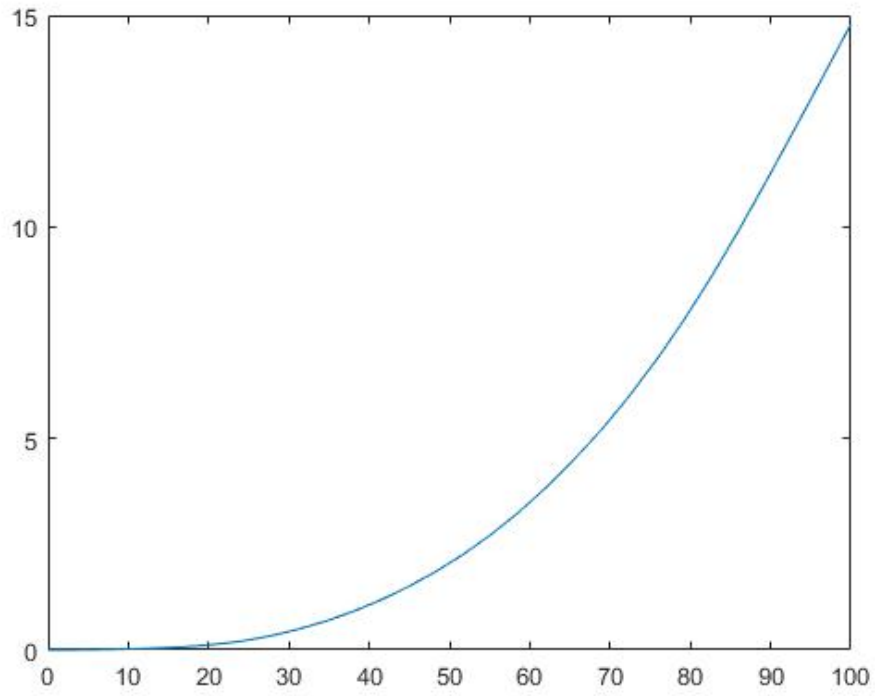
```
%% Set up configureation for deflection run
analysisConfig.meshFile = 'master.db';
analysisConfig.analysisFileName = 'bladeAnalysis';
analysisConfig.np = 1;
analysisConfig.analysisFlags.mass = 0;
analysisConfig.analysisFlags.deflection = 1;
ansysResult =
mainAnsysAnalysis(blade,meshData,defLoadsTable,analysisConfi
g)
```

Demonstration

```
ansysResult =
  struct with fields:
    deflection: {[30×6 double]}

ansysResult.deflection{1}
ans =
         0         0         0         0         0         0
   -0.0012    0.0026    0.0007         0         0         0
   -0.0040    0.0091    0.0013         0         0         0
   -0.0085    0.0217    0.0016         0         0         0
   ...
```

Demonstration

```
%% Plot the flapwise deflection
y=ansysResult.deflection{1}(:,2); %flapwise deflection

figure(1)
plot(blade.ispan,y)
hold on;
```



Demonstration

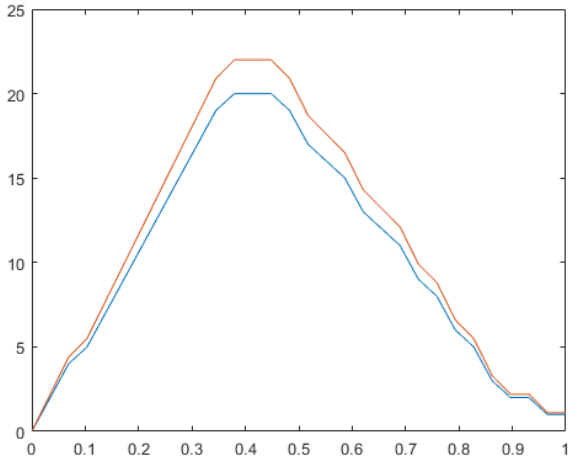
```
%Modify Spar cap suction side
blade.components(3).name
ans =
    'Spar_cap_ss'
figure(2)
plot(blade.components(3).cp(:,1),blade.components(3).cp(:,2))
hold on
blade.components(3).cp(:,2)=blade.components(3).cp(:,2)*1.1;
%Increase component thickness by 10% throughout
plot(blade.components(3).cp(:,1),blade.components(3).cp(:,2))
blade.updateBlade %Run updateBlade with blade data was
modified by user
```

Demonstration

```
%Modify Spar cap suction side
blade.components(3).name
ans =
    'Spar_cap_ss'

blade.components(3).cp
ans =
         0         0
    0.0345    2.2000
    0.0690    4.4000
       …         …
    0.9310    2.2000
    0.9655    1.1000
    1.0000    1.1000
```

Demonstration

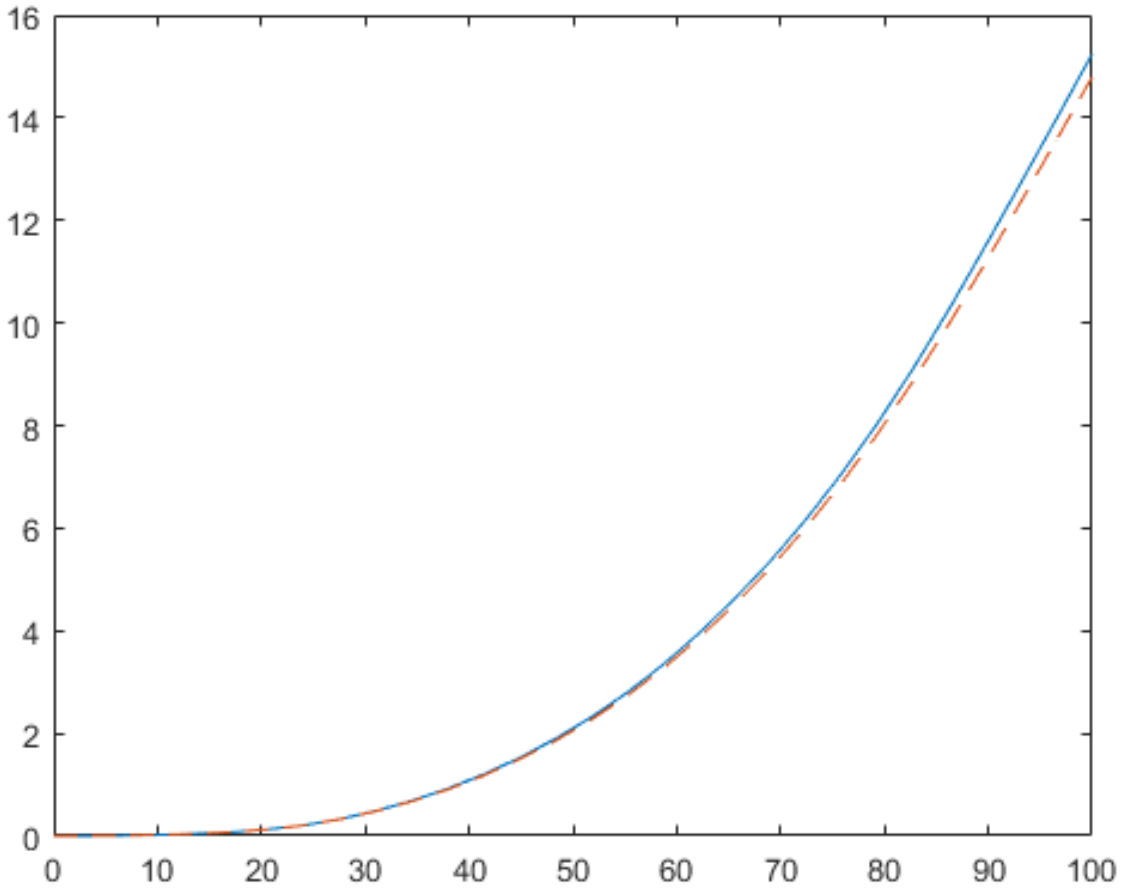
```
figure(2)
plot(blade.components(3).cp(:,1),blade.components(3).cp(:,2))
hold on
blade.components(3).cp(:,2)=blade.components(3).cp(:,2)*1.1;
%Increase component thickness by 10% throughout
plot(blade.components(3).cp(:,1),blade.components(3).cp(:,2))
blade.updateBlade %Run updateBlade with blade data was
modified by user
```



Demonstration

```
%% Build a new ANSYS model; This time with previous mesh data.
Then run deflection analysis again;
blade.generateShellModel('ansys',includeAdhesive,meshData);

ansysResult =
mainAnsysAnalysis(blade,meshData,defLoadsTable,analysisConfig);

X=blade.ispan;
x=ansysResult.deflection{1}(:,1);
y=ansysResult.deflection{1}(:,2); %flapwise deflection
z=ansysResult.deflection{1}(:,3);

figure(1)
plot(blade.ispan,y,'--')
hold on;
```

Demonstration



Demonstration

```
failConfig.meshFile = 'master.db';
failConfig.analysisFileName = 'bladeAnalysis';
failConfig.np = 1;
failConfig.analysisFlags.mass = 1;
failConfig.analysisFlags.globalBuckling = 10;
failConfig.analysisFlags.failure='TWSI';
failConfig.analysisFlags.localFields=1;


ansysResult =
mainAnsysAnalysis(blade,meshData,loadsTable,failConfig)
```

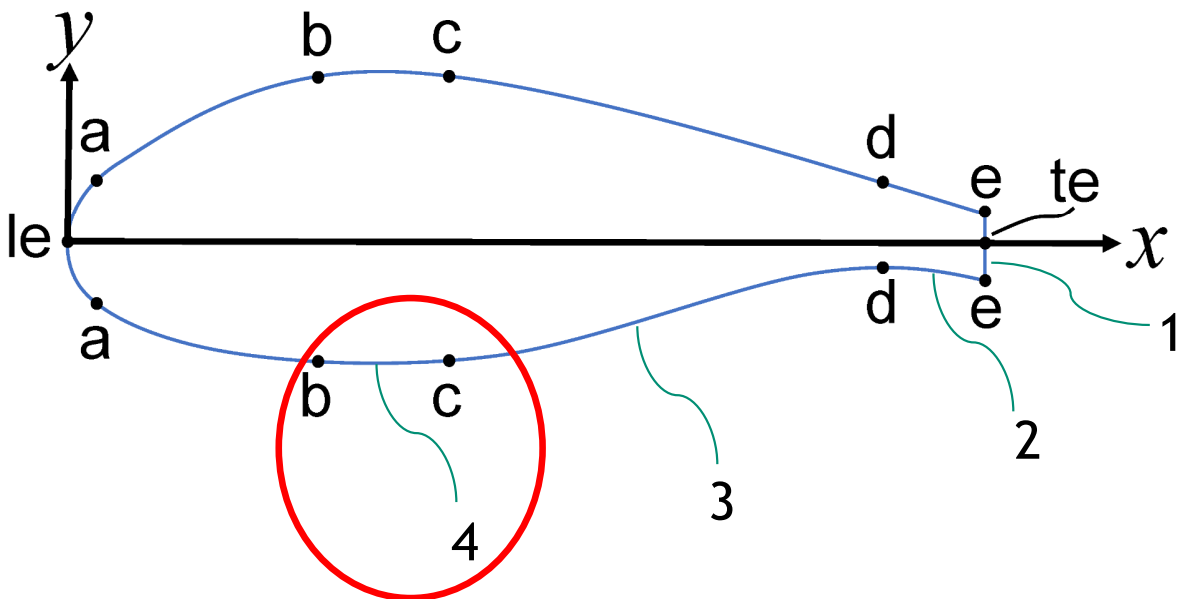
Demonstration

```
ansysResult =

  struct with fields:

              mass: 6.7506e+04
    globalBuckling: {[1.0236]  [1.3190]}
           failure: {[0.4882]  [1.9171]}
       localFields: {[]  []}
```

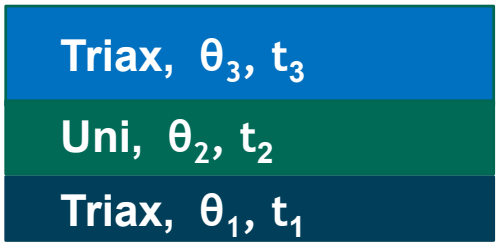
Demonstration

```
meshData.outerShellElSets
ans =
  12×29 elementSet array with properties: name plygroups  elementList
```

Suppose you needed to know the stresses and strains in every layer in the HP spar



Spar cap stack



Demonstration

```
 meshData.outerShellElSets(4,12)
ans =
  elementSet with properties:
            name: '037931_HP_SPAR'
       plygroups: [1×4 struct]
     elementList: [3177 3178 3179 3180 3181 3182 3183 3184 3185
3186 3187 3188 3189 3190 3191 3192 3193 3194 3195 3196 3197
3198 3199 3200]
```

Further suppose you are interested in these two elements

Demonstration

```
%% Local Fields Example
elNo=[3183,3194]; %HP spar cap (station 12).
coordSys='local'
myresult=extractFieldsThruThickness('plateStrains-all-2.txt',meshData,blade.materials,blade.stacks,blade.swstacks,elNo,coordSys)


myresult =
  struct with fields:
    element3183: [1×1 struct]
    element3194: [1×1 struct]
```

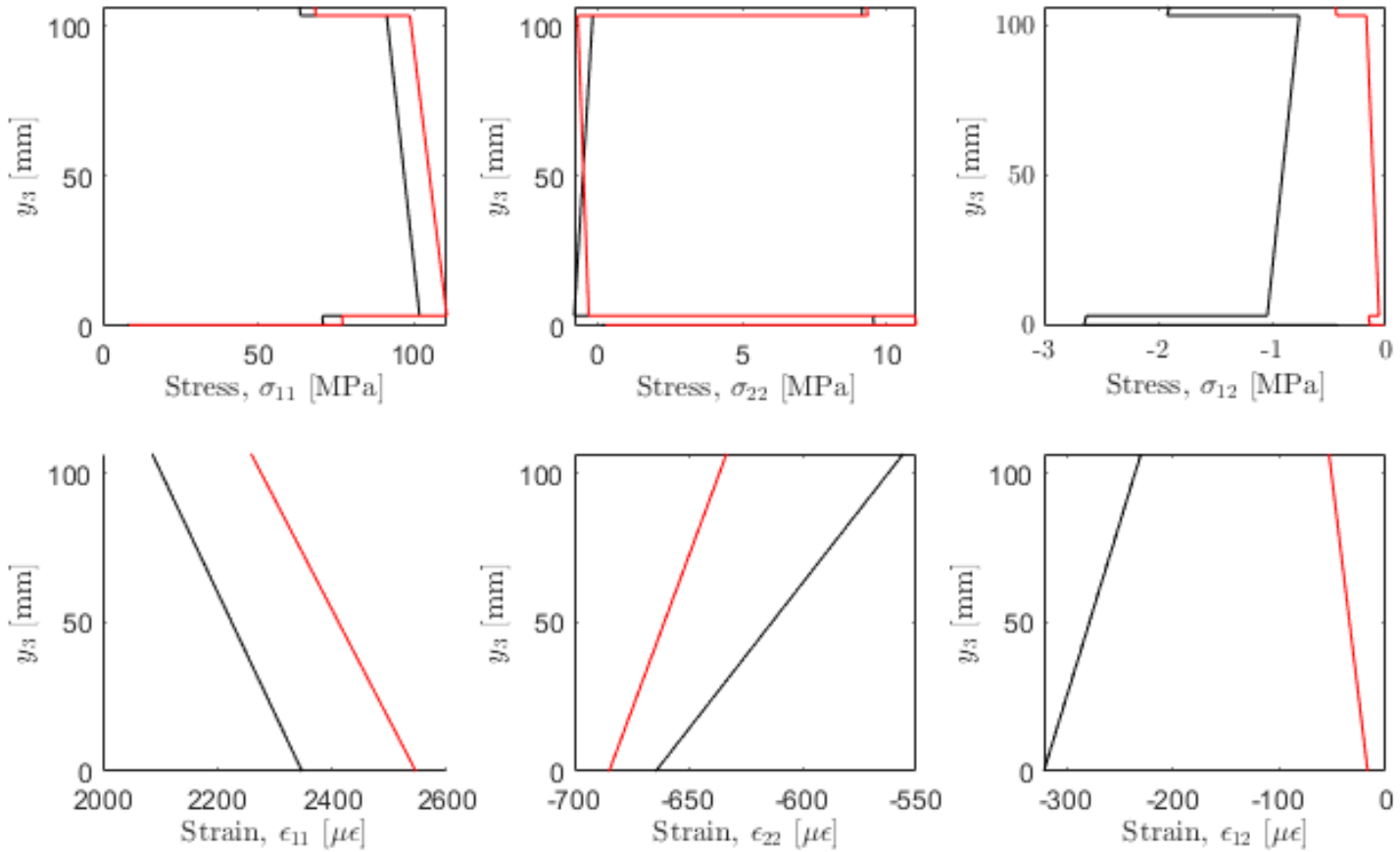
Demonstration

```
myresult.element3183
ans =
  struct with fields:

           x3: [8×1 double]
         eps11: [8×1 double]
         eps22: [8×1 double]
         eps33: [8×1 double]
         eps23: [8×1 double]
         eps13: [8×1 double]
         eps12: [8×1 double]
         sig11: [8×1 double]
         sig22: [8×1 double]
         sig12: [8×1 double]
    matNumber: [8×1 double]
```

Demonstration

```
%Plot the stress and strains though-the-thickness

scaleFactor=1e6; %Used to plot stress in MPa and strain in
microstrain

close all;

figureNumbers=[4,5];

myYlabel='$y_3$ [mm]';

plotLocalFields(figureNumbers,myYlabel,myresult.(['element'
num2str(elNo(1))]),scaleFactor,'k')

hold on;

plotLocalFields(figureNumbers,myYlabel,myresult.(['element'
num2str(elNo(2))]),scaleFactor,'r')
```

Demonstration



Demonstration (template)

```
%% your text here
includeAdhesive=0;
meshData=blade.generateShellModel('ansys',includeAdhesive);
```

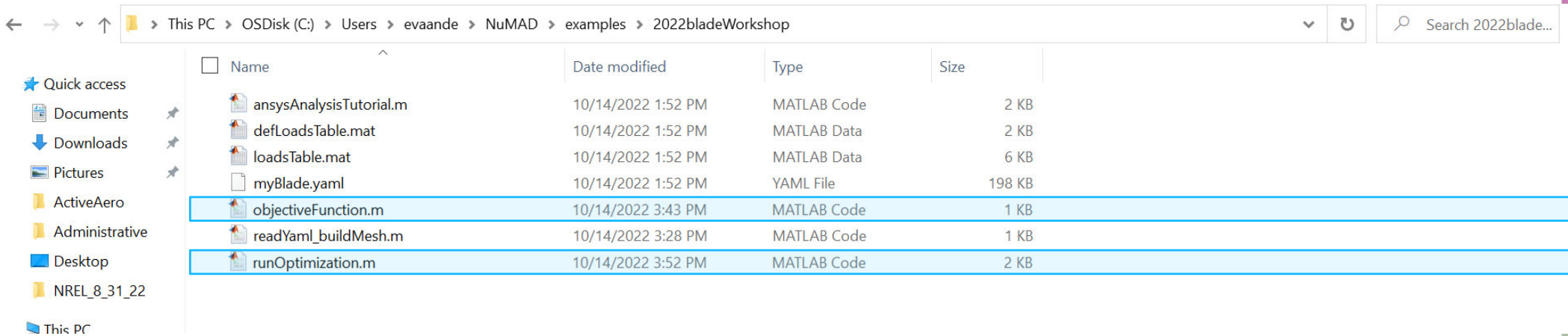
Optimization

First steps in setting up an optimization:

- Identify an objective
  - Minimize mass
  - Minimize cost
- Identify design space/variables and their value ranges
  - Component thickness/width
  - Outer shape/geometry
  - Material properties/layup
- Identify constraints
  - Maximum stress
  - Maximum deflection
  - Resistance to buckling
- Choose an optimizer
  - Gradient based/gradient free

Optimization

NuMAD\examples\2022bladeWorkshop\objectiveFunction.m,
NuMAD\examples\2022bladeWorkshop\runOptimization.m



Optimization

NuMAD\examples\2022bladeWorkshop\objectiveFunction.m,

Editor - C:\Users\evaande\NuMAD\examples\2022bladeWorkshop\objectiveFunction.m

objectiveFunction.m

```
1  function objVal = objectiveFunction(XVar,blade,meshData,defLoadsTable,config)
2      %% Set the thickness of the spar caps based on the current values of XVar
3      blade.components(3).cp(:,2) = XVar(1)*blade.components(3).cp(:,2); % Suction side spar cap
4      blade.components(4).cp(:,2) = XVar(2)*blade.components(4).cp(:,2); % Pressure side spar cap
5
6      blade.updateBlade();
7
8      %% Run ANSYS to determine tip deflection
9      ansysResult = mainAnsysAnalysis(blade,meshData,defLoadsTable,config);
10
11     flapDef = ansysResult.deflection{1}(end,2);
12
13     %% Calculate the value of the objective function
14     objVal = (flapDef - 20.0)^2;
15 end
```

Optimization

NuMAD\examples\2022bladeWorkshop\runOptimization.m

Editor - C:\Users\evaande\NuMAD\examples\2022bladeWorkshop\runOptimization.m

objectiveFunction.m | runOptimization.m

```
%% Add paths to NuMAD source
cd('..\..');
addNumadPaths;

%% Return to the example directory where the needed files are located
cd('examples\2022bladeWorkshop');

%% Create a blade object as an instance of BladeDef and read in data from the .yaml file

blade = BladeDef;
fileName = 'myBlade.yaml';
blade.readYAML(fileName);

%% Generate shell mesh
blade.mesh = 0.2;
includeAdhesive=0;
meshData=blade.generateShellModel('ansys',includeAdhesive);

%% Import the loads table into the workspace
load('defLoadsTable.mat')

%% Set configuration parameters for the ANSYS analsysis
config.meshFile = 'master.db';
config.analysisFileName = 'bladeAnalysis';
config.np = 1;
config.analysisFlags.resultantVSspan = 1;
config.analysisFlags.mass = 0;
config.analysisFlags.deflection = 1;

```

Optimization

NuMAD\examples\2022bladeWorkshop\runOptimization.m

Editor - C:\Users\evaande\NuMAD\examples\2022bladeWorkshop\runOptimization.m

objectiveFunction.m | runOptimization.m

```
22    %% Set configuration parameters for the ANSYS analysis
23 -  config.meshFile = 'master.db';
24 -  config.analysisFileName = 'bladeAnalysis';
25 -  config.np = 1;
26 -  config.analysisFlags.resultantVSspan = 1;
27 -  config.analysisFlags.mass = 0;
28 -  config.analysisFlags.deflection = 1;
29
30    %% Set basic optimization options settings
31 -  Aeq = [];
32 -  beq = [];
33 -  A = [];
34 -  b = [];
35 -  nonlcon = [];
36 -  Xinit = [1.0,1.0];
37 -  lb = [0.25,0.25];
38 -  ub = [2.0,2.0];
39 -  maxIt = 15;
40 -  maxFunc = 75;
41 -  fdStep = [0.2,0.2];
42 -  options = optimoptions('fmincon','SpecifyObjectiveGradient',false,...
43    'FiniteDifferenceType','central','FiniteDifferenceStepSize',fdStep,...
44    'MaxIterations',maxIt,'MaxFunctionEvaluations',maxFunc);
45
46    %% Define the objective function
47 -  fun = @(XVar)objectiveFunction(XVar,blade,meshData,defLoadsTable,config);
48
49    %% Run optimization
50 -  [Xfinal,fval,exitflag,output] = fmincon(fun,Xinit,A,b,Aeq,beq,lb,ub,nonlcon,options);
```