Mystran Analysis Interface Module (AIM)

Ryan Durscher
AFRL/RQVC

June 12, 2017

Contents

1 Introduction

1.1 MYSTRAN AIM Overview

A module in the Computational Aircraft Prototype Syntheses (CAPS) has been developed to interact (primarily through input files) with the finite element structural solver MYSTRAN [1]. MYSTRAN is an open source, general purpose, linear finite element analysis computer program written by Dr. Bill Case. Available at, http://www.mystran.com/ , MYSTRAN currently supports Linux and Windows operating systems.

Current issues include:

- A thorough bug testing needs to be undertaken.

An outline of the AIM's inputs and outputs are provided in AIM Inputs and AIM Outputs, respectively.

The accepted and expected geometric representation and analysis intentions are detailed in Geometry Representation and Analysis Intent.

Details of the AIM's shareable data structures are outlined in AIM Shareable Data if connecting this AIM to other AIMs in a parent-child like manner.

Details of the AIM's automated data transfer capabilities are outlined in MYSTRAN Data Transfer

1.2 Examples

An example problem using the MYSTRAN AIM may be found at Mystran AIM Basic Example .

2 Geometry Representation and Analysis Intent

The attribute capsIntent may be set to either ALL or STRUCTURE for the MYSTRAN AIM. The geometric representation for the AIM requires that bodies be:

- WIREBODY for purely 1D simulations
- FACEBODY or SHEETBODY (non-manifold) for 2D simulations
- SOLIDBODY or SHEETBODY (manifold) for 3D simulations

3 AIM Inputs

The following list outlines the MYSTRAN inputs along with their default value available through the AIM interface. Unless noted these values will be not be linked to any parent AIMs with variables of the same name.

- **Proj_Name = "mystran_CAPS"**
  This corresponds to the project name used for file naming.
- **Tess_Params = [0.025, 0.001, 15.0]**
  Body tessellation parameters used when creating a boundary element model. Tess_Params[0] and Tess↩_Params[1] get scaled by the bounding box of the body. (From the EGADS manual) A set of 3 parameters that drive the EDGE discretization and the FACE triangulation. The first is the maximum length of an ED↩GE segment or triangle side (in physical space). A zero is flag that allows for any length. The second is a curvature-based value that looks locally at the deviation between the centroid of the discrete object and the underlying geometry. Any deviation larger than the input value will cause the tessellation to be enhanced in those regions. The third is the maximum interior dihedral angle (in degrees) between triangle facets (or Edge segment tangents for a WIREBODY tessellation), note that a zero ignores this phase
- **Edge_Point_Min = 4**
  Minimum number of points along an edge to use when creating a boundary element model.
- **Edge_Point_Max = 10**
  Maximum number of points along an edge to use when creating a boundary element model.
- **Quad_Mesh = False**
  Create a quadratic mesh on four edge faces when creating the boundary element model.
- **Property = NULL**
  Property tuple used to input property information for the model, see FEA Property for additional details.
- **Material = NULL**
  Material tuple used to input material information for the model, see FEA Material for additional details.
- **Constraint = NULL**
  Constraint tuple used to input constraint information for the model, see FEA Constraint for additional details.
- **Load = NULL**
  Load tuple used to input load information for the model, see FEA Load for additional details.
- **Analysis = NULL**
  Analysis tuple used to input analysis/case information for the model, see FEA Analysis for additional details.
- **Analysis_Type = "Modal"**
  Type of analysis to generate files for, options include "Modal", "Static", and "Craig-Bampton".
- **Support = NULL**
  Support tuple used to input support information for the model, see FEA Support for additional details.

4 AIM Shareable Data

Currently the MYSTRAN AIM does not have any shareable data types or values. It will try, however, to inherit a "FEA_MESH" or "Volume_Mesh" from any parent AIMs. Note that the inheritance of the mesh is not required.

5 AIM Outputs

The following list outlines the MYSTRAN outputs available through the AIM interface.

- **EigenValue** = List of Eigen-Values (λ) after a modal solve.

- **EigenRadian** = List of Eigen-Values in terms of radians ($\omega = \sqrt{\lambda}$) after a modal solve.
- **EigenFrequency** = List of Eigen-Values in terms of frequencies ($f = \frac{\omega}{2\pi}$) after a modal solve.
- **EigenGeneralMass** = List of generalized masses for the Eigen-Values.

6 MYSTRAN Data Transfer

The MYSTRAN AIM has the ability to transfer displacements and eigenvectors from the AIM and pressure distributions to the AIM using the conservative and interpolative data transfer schemes in CAPS. Currently these transfers may only take place on triangular meshes.

6.1 Data transfer from MYSTRAN

- **"Displacement"**
  Retrieves nodal displacements from the $*$.F06 file.

- **"EigenVector_#"**
  Retrieves modal eigen-vectors from the $*$.F06 file, where "#" should be replaced by the corresponding mode number for the eigen-vector (eg. EigenVector_3 would correspond to the third mode, while EigenVector_6 would be the sixth mode).

6.2 Data transfer to MYSTRAN

- **"Pressure"**
  Writes appropriate load cards using the provided pressure distribution.

7 FEA Material

Structure for the material tuple = ("Material Name", "Value"). "Material Name" defines the reference name for the material being specified. The "Value" can either be a JSON String dictionary (see Section JSON String Dictionary) or a single string keyword (see Section Single Value String).

7.1 JSON String Dictionary

If "Value" is JSON string dictionary (eg. "Value" = {"density": 7850, "youngModulus": 120000.0, "poissonRatio": 0.5, "materialType": "isotropic"}) the following keywords (= default values) may be used:

- **materialType = "Isotropic"**
  Material property type. Options: Isotropic, Anisothotropic, Orthotropic, or Anisotropic.

- **youngModulus = 0.0**
  Also known as the elastic modulus, defines the relationship between stress and strain. Default if 'shear↩ Modulus' and 'poissonRatio' != 0, youngModulus = 2$*$(1+poissonRatio)$*$shearModulus

- **shearModulus = 0.0**
  Also known as the modulus of rigidity is defined as the ratio of shear stress to the shear strain. Default if 'youngModulus' and 'poissonRatio' != 0, shearModulus = youngModulus/(2$*$(1+poissonRatio))

- **poissonRatio = 0.0**
  Is the fraction of expansion divided by the fraction of compression. Default if 'youngModulus' and 'shear↩ Modulus' != 0, poissonRatio = (2$*$youngModulus/shearModulus) - 1

- **density = 0.0**
  Density of the material.

- **thermalExpCoeff = 0.0**
  Thermal expansion coefficient of the material.

- **thermalExpCoeffLateral = 0.0**
  Thermal expansion coefficient of the material.

- **temperatureRef = 0.0**
  Reference temperature for material properties.

- **dampingCoeff = 0.0**
  Damping coefficient for the material.

- **allowType = 0**
  This flag defines if the above allowables `compressAllow` etc. are defined in terms of stress (0) or strain (1). The default is stress (0).

- **youngModulusLateral = 0.0**
  Elastic modulus in lateral direction for an orthotropic material

- **shearModulusTrans1Z = 0.0**
  Transverse shear modulus in the 1-Z plane for an orthotropic material

- **shearModulusTrans2Z = 0.0**
  Transverse shear modulus in the 2-Z plane for an orthotropic material

7.2 Single Value String

If "Value" is a string, the string value may correspond to an entry in a predefined material lookup table. NOT YET IMPLEMENTED!!!!

8 FEA Property

Structure for the property tuple = ("Property Name", "Value"). "Property Name" defines the reference `capsGroup` for the property being specified. The "Value" can either be a JSON String dictionary (see Section JSON String Dictionary) or a single string keyword (see Section Single Value String).

8.1 JSON String Dictionary

If "Value" is JSON string dictionary (eg. "Value" = {"shearMembraneRatio": 0.83, "bendingInertiaRatio": 1.0, "membraneThickness": 0.2, "propertyType": "Shell"}) the following keywords (= default values) may be used:

- **propertyType = No Default value**
  Type of property to apply to a given capsGroup `Name`. Options: ConcentratedMass, Rod, Bar, Shear, Shell, Composite, and Solid

- **material = "Material Name" (FEA Material)**
  "Material Name" from FEA Material to use for property. If no material is set the first material created will be used

- **crossSecArea = 0.0**
  Cross sectional area.

- **torsionalConst = 0.0**
  Torsional constant.

- **torsionalStressReCoeff = 0.0**
  Torsional stress recovery coefficient.

- **massPerLength = 0.0**
  Mass per unit length.

- **zAxisInertia = 0.0**
  Section moment of inertia about the element z-axis.

- **yAxisInertia = 0.0**
  Section moment of inertia about the element y-axis.

- **yCoords[4] = [0.0, 0.0, 0.0, 0.0]**
  Element y-coordinates, in the bar cross-section, of four points at which to recover stresses

- **zCoords[4] = [0.0, 0.0, 0.0, 0.0]**
  Element z-coordinates, in the bar cross-section, of four points at which to recover stresses

- **areaShearFactors[2] = [0.0, 0.0]**
  Area factors for shear.

- **crossProductInertia = 0.0**
  Section cross-product of inertia.

- **shearPanelThickness = 0.0**
  Shear panel thickness.

- **nonStructMassPerArea = 0.0**
  Nonstructural mass per unit area.

- **membraneThickness = 0.0**
  Membrane thickness.

- **bendingInertiaRatio = 1.0**
  Ratio of actual bending moment inertia to the bending inertia of a solid plate of thickness "membranThickness"

- **shearMembraneRatio = 5.0/6.0**
  Ratio shear thickness to membrane thickness.

- **materialBending = "Material Name" (FEA Material)**
  "Material Name" from FEA Material to use for property bending. If no material is given and "bendingIneria↩
  Ratio" is greater than 0, the material name provided in "material" is used.

- **materialShear = "Material Name" (FEA Material)**
  "Material Name" from FEA Material to use for property shear. If no material is given and "shearMembrane↩
  Ratio" is greater than 0, the material name provided in "material" is used.

- **massPerArea = 0.0**
  Mass per unit area.

- **compositeMaterial = "no default"**
  List of "Material Name"s, ["Material Name -1", "Material Name -2", ...], from FEA Material to use for composites.

- **shearBondAllowable = 0.0**
  Allowable interlaminar shear stress.

- **symmetricLaminate = False**
  Symmetric lamination option. True- SYM only half the plies are specified, for odd number plies 1/2 thickness of center ply is specified with the first ply being the bottom ply in the stack, default (False) all plies specified.

- **compositeFailureTheory = "(no default)"**
  Composite failure theory. Options: "HILL", "HOFF", "TSAI", and "STRN"

- **compositeThickness = (no default)**
  List of composite thickness for each layer (eg. [1.2, 4.0, 3.0]). If the length of this list doesn't match the length of the "compositeMaterial" list, the list is either truncated [>length("compositeMaterial")] or expanded [<length("compositeMaterial")] in which case the last thickness provided is repeated.

- **compositeOrientation = (no default)**
  List of composite orientations (angle relative element material axis) for each layer (eg. [5.0, 10.0, 30.0]). If the length of this list doesn't match the length of the "compositeMaterial" list, the list is either truncated [>length("compositeMaterial")] or expanded [<length("compositeMaterial")] in which case the last orientation provided is repeated.

- **mass = 0.0**
  Mass value.

- **massOffset = [0.0, 0.0, 0.0]**
  Offset distance from the grid point to the center of gravity for a concentrated mass.

- **massInertia = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]**
  Mass moment of inertia measured at the mass center of gratvity.

8.2 Single Value String

If "Value" is a string, the string value may correspond to an entry in a predefined property lookup table. NOT YET IMPLEMENTED!!!!

9 FEA Constraint

Structure for the constraint tuple = ("Constraint Name", "Value"). "Constraint Name" defines the reference name for the constraint being specified. The "Value" can either be a JSON String dictionary (see Section JSON String Dictionary) or a single string keyword (see Section Single Value String).

9.1 JSON String Dictionary

If "Value" is JSON string dictionary (eg. "Value" = {"groupName": "plateEdge", "dofConstraint": 123456}) the following keywords (= default values) may be used:

- **constraintType = "ZeroDisplacement"**
  Type of constraint. Options: "Displacement", "ZeroDisplacement".

- **groupName = "(no default)"**
  Single or list of `capsConstraint` names on which to apply the constraint (e.g. "Name1" or ["Name1","↩ Name2",...]. If not provided, the constraint tuple name will be used.

- **dofConstraint = 0**
  Component numbers / degrees of freedom that will be constrained (123 - zero translation in all three directions).

- **gridDisplacement = 0.0**
  Value of displacement for components defined in "dofConstraint".

9.2 Single Value String

If "Value" is a string, the string value may correspond to an entry in a predefined constraint lookup table. NOT YET IMPLEMENTED!!!!

10 FEA Support

Structure for the support tuple = ("Support Name", "Value"). "Support Name" defines the reference name for the support being specified. The "Value" can either be a JSON String dictionary (see Section JSON String Dictionary) or a single string keyword (see Section Single Value String).

10.1 JSON String Dictionary

If "Value" is JSON string dictionary (eg. "Value" = {"groupName": "plateEdge", "dofSupport": 123456}) the following keywords (= default values) may be used:

- **groupName = "(no default)"**
  Single or list of `capsConstraint` names on which to apply the support (e.g. "Name1" or ["Name1","↩ Name2",...]. If not provided, the constraint tuple name will be used.

- **dofSupport = 0**
  Component numbers / degrees of freedom that will be supported (123 - zero translation in all three directions).

10.2 Single Value String

If "Value" is a string, the string value may correspond to an entry in a predefined support lookup table. NOT YET IMPLEMENTED!!!!

11 FEA Connection

Structure for the connection tuple = ("Connection Name", "Value"). "Connection Name" defines the reference name to the capsConnect being specified and denotes the "source" node for the connection. The "Value" can either be a JSON String dictionary (see Section JSON String Dictionary) or a single string keyword (see Section Single Value String).

11.1 JSON String Dictionary

If "Value" is JSON string dictionary (eg. "Value" = {"dofDependent": 1, "propertyType": "RigidBody"}) the following keywords (= default values) may be used:

- **connectionType = RigidBody**
  Type of connection to apply to a given capsConnect pair defined by "Connection Name" and the "groupName". Options: Mass (scalar), Spring (scalar), Damper (scalar), RigidBody.

- **dofDependent = 0**
  Component numbers / degrees of freedom of the dependent end of rigid body connections (ex. 123 - translation in all three directions).

- **componentNumberStart = 0**
  Component numbers / degrees of freedom of the starting point of the connection for mass, spring, and damper elements (scalar) (0 <= Integer <= 6).

- **componentNumberEnd= 0**
  Component numbers / degrees of freedom of the ending point of the connection for mass, spring, and damper elements (scalar) (0 <= Integer <= 6).

- **stiffnessConst = 0.0**
  Stiffness constant of a spring element (scalar).

- **dampingConst = 0.0**
  Damping coefficient/constant of a spring or damping element (scalar).

- **stressCoeff = 0.0**
  Stress coefficient of a spring element (scalar).

- **mass = 0.0**
  Mass of a mass element (scalar).

- **groupName = "(no default)"**
  Single or list of `capsConnect` names on which to connect the nodes found with the tuple name ("↩ Connection Name") to. (e.g. "Name1" or ["Name1","Name2",...].

11.2 Single Value String

If "Value" is a string, the string value may correspond to an entry in a predefined connection lookup table. NOT YET IMPLEMENTED!!!!

12 FEA Load

Structure for the load tuple = ("Load Name", "Value"). "Load Name" defines the reference name for the load being specified. The "Value" can either be a JSON String dictionary (see Section JSON String Dictionary) or a single string keyword (see Section Single Value String).

12.1 JSON String Dictionary

If "Value" is JSON string dictionary (eg. "Value" = {"groupName": "plate", "loadType": "Pressure", "pressureForce": 2000000.0}) the following keywords (= default values) may be used:

- **loadType = "(no default)"**
  Type of load. Options: "GridForce", "GridMoment", "Rotational", "Thermal", "Pressure", "PressureDistribute", "PressureExternal", "Gravity".

- **groupName = "(no default)"**
  Single or list of `capsLoad` names on which to apply the load (e.g. "Name1" or ["Name1","Name2",...]. If not provided, the load tuple name will be used.

- **loadScaleFactor = 1.0**
  Scale factor to use when combining loads.

- **forceScaleFactor = 0.0**
  Overall scale factor for the force for a "GridForce" load.

- **directionVector = [0.0, 0.0, 0.0]**
  X-, y-, and z- components of the force vector for a "GridForce", "GridMoment", or "Gravity" load.

- **momentScaleFactor = 0.0**
  Overall scale factor for the moment for a "GridMoment" load.

- **gravityAcceleration = 0.0**
  Acceleration value for a "Gravity" load.

- **pressureForce = 0.0**
  Uniform pressure force for a "Pressure" load (only applicable to 2D elements).

- **pressureDistributeForce = [0.0, 0.0, 0.0, 0.0]**
  Distributed pressure force for a "PressureDistribute" load (only applicable to 2D elements). The four values correspond to the 4 (quadrilateral elements) or 3 (triangle elements) node locations.

- **angularVelScaleFactor = 0.0**
  An overall scale factor for the angular velocity in revolutions per unit time for a "Rotational" load.

- **angularAccScaleFactor = 0.0**
  An overall scale factor for the angular acceleration in revolutions per unit time squared for a "Rotational" load.

- **coordinateSystem = "(no default)"**
  Name of coordinate system in which defined force components are in reference to. If no value is provided the global system is assumed.

12.2 Single Value String

If "Value" is a string, the string value may correspond to an entry in a predefined load lookup table. NOT YET IMPLEMENTED!!!!

13 FEA Analysis

Structure for the analysis tuple = (‘Analysis Name', ‘Value'). 'Analysis Name' defines the reference name for the analysis being specified. The "Value" can either be a JSON String dictionary (see Section JSON String Dictionary) or a single string keyword (see Section Single Value String).

13.1 JSON String Dictionary

If "Value" is JSON string dictionary (eg. "Value" = {"numDesiredEigenvalue": 10, "eigenNormaliztion": "MAS↩
S", "numEstEigenvalue": 1, "extractionMethod": "GIV", "frequencyRange": [0, 10000]}) the following keywords (= default values) may be used:

- **analysisType = "Modal"**
  Type of load. Options: "Modal", "Static".

- **analysisLoad = "(no default)"**
  Single or list of "Load Name"s defined in FEA Load in which to use for the analysis (e.g. "Name1" or ["Name1","Name2",...].

- **analysisConstraint = "(no default)"**
  Single or list of "Constraint Name"s defined in FEA Constraint in which to use for the analysis (e.g. "Name1" or ["Name1","Name2",...].

- **analysisSupport = "(no default)"**
  Single or list of "Support Name"s defined in FEA Support in which to use for the analysis (e.g. "Name1" or ["Name1","Name2",...].

- **extractionMethod = "(no default)"**
  Extraction method for modal analysis.

- **frequencyRange = [0.0, 0.0]**
  Frequency range of interest for modal analysis.

- **numEstEigenvalue = 0**
  Number of estimated eigenvalues for modal analysis.

- **numDesiredEigenvalue = 0**
  Number of desired eigenvalues for modal analysis.

- **eigenNormaliztion = "(no default)"**
  Method of eigenvector renormilization. Options: "POINT", "MAX", "MASS"

- **gridNormaliztion = 0**
  Grid point to be used in normalizing eigenvector to 1.0 when using eigenNormaliztion = "POINT"

- **componentNormaliztion = 0**
  Degree of freedom about "gridNormalization" to be used in normalizing eigenvector to 1.0 when using eigenNormaliztion = "POINT"

- **lanczosMode = 2**
  Mode refers to the Lanczos mode type to be used in the solution. In mode 3 the mass matrix, Maa,must be nonsingular whereas in mode 2 the matrix K aa - sigma∗Maa must be nonsingular

- **lanczosType = "(no default)"**
  Lanczos matrix type. Options: DPB, DGB.

13.2 Single Value String

If "Value" is a string, the string value may correspond to an entry in a predefined analysis lookup table. NOT YET IMPLEMENTED!!!!

14 FEA DesignVariable

Structure for the design variable tuple = ("DesignVariable Name", "Value"). "DesignVariable Name" defines the reference name for the design variable being specified. This string will be used in the FEA input directly. The "Value" must be a JSON String dictionary (see Section JSON String Dictionary).

14.1 JSON String Dictionary

If "Value" is JSON string dictionary the following keywords (= default values) may be used:

15 FEA DesignConstraint

Structure for the design constraint tuple = (‘DesignConstraint Name', ‘Value'). 'DesignConstraint Name' defines the reference name for the design constraint being specified. The "Value" must be a JSON String dictionary (see Section JSON String Dictionary).

15.1 JSON String Dictionary

If "Value" is JSON string dictionary the following keywords (= default values) may be used:

16 Mystran AIM Basic Example

This is a walkthrough for using MYSTRAN AIM to analyze a three-dimensional wing with internal ribs and spars.

16.1 Prerequisites

It is presumed that ESP and CAPS have been already installed, as well as MYSTRAN.

16.1.1 Script files

Two scripts are used for this illustration:

1. feaWingBEM.csm: Creates geometry, as described in the next section (Creating Geometry using ESP).
2. mystran_PyTest.py: pyCAPS script for performing analysis, as described in Performing analysis using pyC↩ APS .

16.2 Creating Geometry using ESP

In our example ∗.csm file setting up the CAPS fidelity is the first step. If multiple bodies exist in the ∗.csm file the tag, capsIntent, can be used to distinguish what type of analysis the body may be used for. In this example, the geometry model generated can be used for structural analysis, as shown:

```
attribute capsIntent STRUCTURE
```

A typical geometry model can be created and interactively modified using design parameters. These design parameters are either design- or geometry- based. In this example a wing configuration is created using following design parameters.

```
# Design Parameters for OML
despmtr   thick     0.12      frac of local chord
despmtr   camber    0.04      frac of loacl chord

despmtr   area      10.0
despmtr   aspect    6.00
despmtr   taper     0.60
despmtr   sweep     20.0      deg (of c/4)

despmtr   washout   5.00      deg (down at tip)
despmtr   dihedral  4.00      deg
```

```
# Design Parameters for BEM
despmtr   nrib      11        number of ribs
despmtr   spar1     0.20      frac of local chord
despmtr   spar2     0.75      frac of local chord
```

After our design parameters are defined they are used to setup other local variables (analytically) for the outer model line (OML).

```
# OML
set       span      sqrt(aspect*area)
set       croot     2*area/span/(1+taper)
set       ctip      croot*taper
set       dxtip     (croot-ctip)/4+span/2*tand(sweep)
set       dytip     span/2*tand(dihedral)
```

In a similar manner, local variables are defined for the ribs and spars.

```
# wing ribs
dimension waffle  nrib+4 4 0
set       Nrib    nint(nrib)
patbeg    i Nrib
   set    waffle[i,1]     (span/2)*(2*i-Nrib-1)/Nrib
   set    waffle[i,2]     -0.01*croot
   set    waffle[i,3]     (span/2)*(2*i-Nrib-1)/Nrib
   set    waffle[i,4]     max(croot,dxtip+ctip)
patend

# wing spars
set       eps             0.01*span

set       waffle[Nrib+1,1]  -span/2-eps
set       waffle[Nrib+1,2]   spar1*ctip+dxtip
set       waffle[Nrib+1,3]   0
set       waffle[Nrib+1,4]   spar1*croot
set       waffle[Nrib+2,1]   span/2+eps
set       waffle[Nrib+2,2]   spar1*ctip+dxtip
set       waffle[Nrib+2,3]   0
set       waffle[Nrib+2,4]   spar1*croot
set       waffle[Nrib+3,1]  -span/2-eps
set       waffle[Nrib+3,2]   spar2*ctip+dxtip
set       waffle[Nrib+3,3]   0
set       waffle[Nrib+3,4]   spar2*croot
set       waffle[Nrib+4,1]   span/2+eps
set       waffle[Nrib+4,2]   spar2*ctip+dxtip
set       waffle[Nrib+4,3]   0
set       waffle[Nrib+4,4]   spar2*croot
```

Once all design and local variables are defined, a full span, solid model is created by "ruling" together NACA series airfoils (following a series of scales, rotations, and translations).

```
mark
   # Right tip
   udprim    naca      Thickness thick     Camber    camber
   scale     ctip
   rotatez   washout   ctip/4    0
   translate dxtip     dytip     -span/2

   # root
   udprim    naca      Thickness thick     Camber    camber
   scale     croot

   # left tip
   udprim    naca      Thickness thick     Camber    camber
   scale     ctip
   rotatez   washout   ctip/4    0
   translate dxtip     dytip     +span/2
rule
   attribute OML 1
```

Once complete, the wing is stored for later use under the name OML.

```
store     OML
```

Next, the inner layout of the ribs and spars are created using the waffle udprim.

```
udprim    waffle    Depth     +6*thick*croot      Segments  waffle
```

An attribute is then placed on ribs and spars so that the geometry components may be reference by the MYSTRAN AIM.

```
attribute capsGroup $Ribs_and_Spars
```

Following a series of rotations and translations the ribs and spars are stored for latter use.

```
translate 0         0         -3*thick*croot
rotatey   90        0         0
rotatez  -90        0         0

store     layoutRibSpar
```

Next, the layout of the ribs and spars are intersected the outer mold line of wing, which results in only keeping the part of layout that is inside the OML.

```
restore   layoutRibSpar
restore   OML
intersect
```

Finally, select faces (airfoil sections at the root) are tagged, so that a constraint may be applied later.

```
select face 31
    attribute capsConstraint $Rib_Constraint
select face 27
    attribute capsConstraint $Rib_Constraint
select face 26
    attribute capsConstraint $Rib_Constraint
```

The above ∗.csm file results in the follow geometry model:

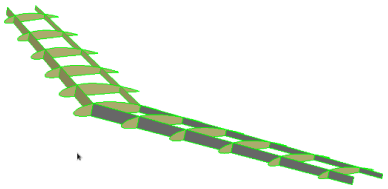

Figure 1: Wing built up element model

16.3 Performing analysis using pyCAPS

The first step in the pyCAPS script is to import the required modules. For this example the following modules are used,

```
from __future__ import print_function

try:
    import os
except:
    print ("Unable to import os module")
    raise SystemError
```

In order to create a new capsProblem the pyCAPS module also needs to be imported; on Linux and OSx this is the pyCAPS.so file, while on Windows it is the pyCAPS.pyd file. For convenience, it is recommended that the path to this file is added to the environmental variable PYTHONPATH.

```
from pyCAPS import capsProblem
```

Similarly, local variables used throughout the script may be defined.

```
workDir = "MystranModalWingBEM"
projectName = workDir
```

Once the required modules have been loaded, a capsProblem can be instantiated.

```
myProblem = capsProblem()
```

Next, using the loadCAPS() function, the desired geometry file is then loaded into the problem.

```
myProblem.loadCAPS("./csmData/feaWingBEM.csm")
```

After the geometry is loaded, the MYSTRAN AIM needs to be instantiated. Note that below, the capsIntent is set to "ALL" as opposed to "STRUCTURE" as specified above in the ∗.csm file. This is only valid since there is only one body in the ∗.csm file. If more than one body existed in the ∗.csm file the capsIntent during the loadAIM() function call should be set "STRUCTURE".

```
mystranAIM = myProblem.loadAIM(aim = "mystranAIM",
                               altName = "mystran",
                               analysisDir= workDir,
                               capsIntent = "ALL")
```

Once loaded analysis parameters specific to MYSTRAN need to be set (see AIM Inputs). These parameters are automatically converted into MYSTRAN specific format and transferred into the MYSTRAN configuration file. One will note in the following snippet the instance of the AIM is referenced in two different manners: 1. Using the returned object from load call and 2. Using the "altName" name reference in the analysis dictionary. While syntactically different, these two forms are essentially identical.

```
# Set project name so a mesh file is generated
mystranAIM.setAnalysisVal("Proj_Name", projectName)

# Set meshing inputs
myProblem.analysis["mystran"].setAnalysisVal("Edge_Point_Max", 4)

myProblem.analysis["mystran"].setAnalysisVal("Quad_Mesh", True)
```

Along the same lines of setting the input values above the "Analysis" (see FEA Analysis), "Material" (see F↩ EA Material), "Property" (see FEA Property), and "Constraint" (see FEA Constraint) tuples are used to set more complex information. The user is encouraged to read the additional documentation on these inputs for further explanations. Once provided this information is converted into MYSTRAN specific syntax and set in the MYSTRAN configuration file.

```
# Set analysis
eigen = { "extractionMethod"     : "Lanczos",
          "frequencyRange"       : [0, 50],
          "numEstEigenvalue"     : 1,
          "eigenNormaliztion"    : "MASS"}
mystranAIM.setAnalysisVal("Analysis", ("EigenAnalysis", eigen))

# Set materials
unobtainium  = {"youngModulus" : 2.2E11 ,
                "poissonRatio" : .33,
                "density"      : 7850}

madeupium    = {"materialType" : "isotropic",
                "youngModulus" : 1.2E9 ,
                "poissonRatio" : .5,
                "density"      : 7850}
mystranAIM.setAnalysisVal("Material", [("Unobtainium", unobtainium),
                                       ("Madeupium", madeupium)])

# Set property
shell  = {"propertyType"      : "Shell",
          "membraneThickness" : 0.2,
          "bendingInertiaRatio" : 1.0, # Default
          "shearMembraneRatio"  : 5.0/6.0} # Default }

mystranAIM.setAnalysisVal("Property", ("Ribs_and_Spars", shell))

# Set constraints
constraint = {"groupName" : ["Rib_Constraint"],
              "dofConstraint" : 123456}

mystranAIM.setAnalysisVal("Constraint", ("ribConstraint", constraint))
```

After all desired options are set aimPreAnalysis needs to be executed. Based on the input provided, MYSTRAN specific files are generated during this call.

```
mystranAIM.aimPreAnalysis()
```

At this point the required files necessary run MYSTRAN should have be created and placed in the specified analysis working directory. Next MYSTRAN needs to executed. In this example an OS system is made such as,

```
print ("\n\nRunning MYSTRAN......")
currentDirectory = os.getcwd() # Get our current working directory

os.chdir(mystranAIM.analysisDir) # Move into test directory

os.system("mystran.exe " + projectName +  ".dat"); # Run MYSTRAN via system call

os.chdir(currentDirectory) # Move back to working directory
print ("Done running MYSTRAN!")
```

After MYSTRAN is finished running aimPostAnalysis needs to be executed.

```
mystranAIM.aimPostAnalysis()
```

Finally available AIM outputs (see AIM Outputs) may be retrieved, for example:

```
print ("\nGetting results for natural frequencies.....")
natrualFreq = mystranAIM.getAnalysisOutVal("EigenFrequency")

mode = 1
for i in natrualFreq:
    print ("Natural freq ( Mode", mode, ") = ", i, "(Hz)")
    mode += 1
```

results in,

```
Natural freq (Mode 1) = 1.89166 (Hz)
Natural freq (Mode 2) = 6.33335 (Hz)
Natural freq (Mode 3) = 6.51397 (Hz)
Natural freq (Mode 4) = 23.88463 (Hz)
Natural freq (Mode 5) = 24.98205 (Hz)
Natural freq (Mode 6) = 28.23676 (Hz)
Natural freq (Mode 7) = 32.53667 (Hz)
Natural freq (Mode 8) = 33.92054 (Hz)
Natural freq (Mode 9) = 43.49964 (Hz)
```

When finally finished with the script, the open CAPS problem should be closed.

```
myProblem.closeCAPS()
```

16.4 Executing pyCAPS script

Issuing the following command executes the script:

```
python mystran_PyTest.py
```

References

[1] William Case. *MYSTRAN General Purpose Finite Element Structural Analysis Computer Program (Linux Version 6.35) [Software Manual]*, Nov. 2011. Available from http://www.MYSTRAN.com. 1