Astros Analysis Interface Module (AIM)

Ryan Durscher and Ed Alyanak
AFRL/RQVC

December 8, 2022

0.1 Introduction

0.1.1 Astros 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 ASTROS.

Current issues include:

- A thorough bug testing needs to be undertaken.

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

The Astros AIM can automatically execute Astros, with details provided in AIM Execution.

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

0.1.2 Clearance Statement

This software has been cleared for public release on 05 Nov 2020, case number 88ABW-2020-3462.

0.2 Astros AIM attributes

The following list of attributes are required for the Astros AIM inside the geometry input.

- **capsDiscipline** This attribute is a requirement if doing aeroelastic analysis within Astros. capsDiscipline allows the AIM to determine which bodies are meant for structural analysis and which are used for aerodynamics. Options are: Structure and Aerodynamic (case insensitive).
- **capsGroup** This is a name assigned to any geometric body. This body could be a solid, surface, face, wire, edge or node. Recall that a string in ESP starts with a $. For example, attribute `capsGroup $Wing`.
- **capsLoad** This is a name assigned to any geometric body where a load is applied. This attribute was separated from the `capsGroup` attribute to allow the user to define a local area to apply a load on without adding multiple `capsGroup` attributes. Recall that a string in ESP starts with a $. For example, attribute `capsLoad $force`.
- **capsConstraint** This is a name assigned to any geometric body where a constraint/boundary condition is applied. This attribute was separated from the `capsGroup` attribute to allow the user to define a local area to apply a boundary condition without adding multiple `capsGroup` attributes. Recall that a string in ESP starts with a $. For example, attribute `capsConstraint $fixed`.
- **capsIgnore** It is possible that there is a geometric body (or entity) that you do not want the Astros AIM to pay attention to when creating a finite element model. The capsIgnore attribute allows a body (or entity) to be in the geometry and ignored by the AIM. For example, because of limitations in OpenCASCADE a situation where two edges are overlapping may occur; capsIgnore allows the user to only pay attention to one of the overlapping edges.
- **capsConnect** This is a name assigned to any geometric body where the user wishes to create "fictitious" connections such as springs, dampers, and/or rigid body connections to. The user must manually specify the connection between two `capsConnect` entities using the "Connect" tuple (see AIM Inputs). Recall that a string in ESP starts with a $. For example, attribute `capsConnect $springStart`.

- **capsConnectLink** Similar to `capsConnect`, this is a name assigned to any geometric body where the user wishes to create "fictitious" connections to. A connection is automatically made if a `capsConnectLink` matches a `capsConnect` group. Again, further specifics of the connection are input using the "Connect" tuple (see AIM Inputs). Recall that a string in ESP starts with a $. For example, attribute `capsConnectLink $springEnd`.
- **capsBound** This is used to mark surfaces on the structural grid in which data transfer with an external solver will take place. See Astros Data Transfer for additional details.

Internal Aeroelastic Analysis

- **capsBound** This is used to mark surfaces on the structural grid in which a spline will be created between the structural and aero-loads.
- **capsReferenceArea** [Optional: Default 1.0] Reference area to use when doing aeroelastic analysis. This attribute may exist on any aerodynamic cross-section.
- **capsReferenceChord** [Optional: Default 1.0] Reference chord to use when doing aeroelastic analysis. This attribute may exist on any aerodynamic cross-section.
- **capsReferenceSpan** [Optional: Default 1.0] Reference span to use when doing aeroelastic analysis. This attribute may exist on any aerodynamic cross-section.

0.3 AIM Inputs

The following list outlines the Astros 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 = "astros_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 EDGE 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.
- **aimInputsAstros = 2**
  Minimum number of points on an edge including end points to use when creating a surface mesh (min 2).
- **Edge_Point_Max = 50**
  Maximum number of points on an edge including end points to use when creating a surface mesh (min 2).
- **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", "AeroelasticTrim", "AeroelasticTrim↩ Opt", "AeroelasticFlutter", "Optimization", "AeroelasticOptimization". For optimizations involving both steady and unsteady aerodynamic models, use "AeroelasticOptimization". Note: "Aeroelastic" and "StaticOpt" are still supported and refer to "AeroelasticTrim" and "Optimization".

- **File_Format = "Small"**
  Formatting type for the bulk file. Options: "Small", "Large", "Free".

- **Mesh_File_Format = "Free"**
  Formatting type for the mesh file. Options: "Small", "Large", "Free".

- **Design_Variable = NULL**
  The design variable tuple used to input design variable information for the model optimization, see FEA Design Variables for additional details.

- **Design_Variable_Relation = NULL**
  The design variable relation tuple is used to input design variable relation information for the model optimization, see FEA DesignVariableRelation for additional details.

- **Design_Constraint = NULL**
  The design constraint tuple used to input design constraint information for the model optimization, see FEA Design Constraints for additional details.

- **Objective_Min_Max = "Max"**
  Maximize or minimize the design objective during an optimization. Option: "Max" or "Min".

- **Objective_Response_Type = "Weight"**
  Object response type (see Astros manual).

- **Optimization_Control = NULL**
  Optimization case control (see Astros manual).

- **VLM_Surface = NULL**
  Vortex lattice method tuple input. See Vortex Lattice Surface for additional details.

- **VLM_Control = NULL**
  Vortex lattice method control surface tuple input. See Vortex Lattice Control Surface for additional details.

- **Support = NULL**
  Support tuple used to input support information for the model, see FEA Support for additional details.

- **Connect = NULL**
  Connect tuple used to define connection to be made in the, see FEA Connection for additional details.

- **Parameter = NULL**
  Parameter tuple used to define user entries. This can be used to input things to ASTROS such as CONVERT or MFORM etc. The input is in Tuple form ("DATACARD", "DATAVALUE"). All inputs are strings. Example: ("CONVERT","MASS, 0.00254"). Note: Inputs assume a "," delimited entry. Notice the "," after MASS in the Example.

- **Aero_Reference = NULL**
  A JSON dictionary used to define aerodynamic reference parameters. see FEA Aerodynamic References for additional details

- **Mesh = NULL**
  A Mesh link.

0.4 AIM Execution

If auto execution is enabled when creating an Astros AIM, the AIM will execute Astros just-in-time with the command line:

```
$ASTROS_ROOT/astros < $Proj_Name.dat > $Proj_Name.out
```

where preAnalysis generated the file Proj_Name + ".dat" which contains the input information. The environemtn variable ASTROS_ROOT is assumed to point to the location where the "astros.exe" executable and run files "↩ ASTRO.D01" and "ASTRO.IDX" are located.

The analysis can be also be explicitly executed with caps_execute in the C-API or via Analysis.runAnalysis in the pyCAPS API.

Calling preAnalysis and postAnalysis is NOT allowed when auto execution is enabled.

Auto execution can also be disabled when creating an Astros AIM object. In this mode, caps_execute and Analysis.runAnalysis can be used to run the analysis, or Astros can be executed by calling preAnalysis, system call, and posAnalysis as demonstrated below with a pyCAPS example:

```
print ("\n\preAnalysis......")
astros.preAnalysis()
print ("\n\nRunning......")
astros.system(ASTROS_ROOT + os.sep + "astros.exe < " + astros.input.Proj_Name + ".dat > " +
       astros.input.Proj_Name + ".out"); # Run via system call
print ("\n\postAnalysis......")
astros.postAnalysis()
```

0.5 AIM Outputs

The following list outlines the Astros 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.
- **EigenGeneralStiffness** = List of generalized stiffness for the Eigen-Values.

0.6 Astros Data Transfer

The Astros 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.

0.6.1 Data transfer from Astros (FieldOut)

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

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

0.6.2 Data transfer to Astros (FieldIn)

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

0.7 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).

0.7.1 JSON String Dictionary

If "Value" is JSON string dictionary (e.g. "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**
  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.

0.7.2 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!!!!

0.8 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).

0.8.1 JSON String Dictionary

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

- **propertyType = No Default value**
  Type of property to apply to a give capsGroup `Name`. Options: ConcentratedMass, Rod, Bar, Shear, Shell, Membrane, 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**
  Non-structural 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.

- **crossSecType = NULL**
  Cross-section type. Must be one of following character variables: I, T, BOX, BAR, TUBE, ROD, HAT, or GBOX.

- **crossSecDimension = [0,0,0,....]**
  Cross-sectional dimensions (length of array is dependent on the "crossSecType"). Max supported length array is 10!

- **membraneThickness = 0.0**
  Membrane thickness.

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

- **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.

- **materialShear = "Material Name" (FEA Material)**
  "Material Name" from FEA Material to use for property shear.

- **massPerArea = 0.0**
  Non-structural mass per unit area.

- **zOffsetRel = 0.0**
  Relative offset from the surface of grid points to the element reference plane as a percentage of the thickness. zOffSet = thickness∗zOffsetRel/100

- **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. If "True" only half the plies are specified (the plies will be repeated in reverse order internally in the PCOMP card). For an odd number of plies, the 1/2 thickness of the 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.

- **compositeThickness = (no default)**
  List of composite thickness for each layer (e.g. [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 gravity.

0.8.2 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!!!!

0.9 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).

0.9.1 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".

- **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".

0.9.2 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!!!!

0.10 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).

0.10.1 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).

0.10.2 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!!!!

0.11 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).

0.11.1 JSON String Dictionary

If "Value" is JSON string dictionary (e.g. "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), RigidBody, RigidBodyInterpolate.

- **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, damper elements (scalar), and rigid body interpolative connection (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).

- **weighting = 1**
  Weighting factor for a rigid body interpolative connections.

- **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",...].

0.11.2 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!!!!

0.12 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).

0.12.1 JSON String Dictionary

If "Value" is JSON string dictionary (e.g. "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.

- **pressureDistributeForce = [0.0, 0.0, 0.0, 0.0]**
  Distributed pressure force for a "PressureDistribute" load. 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.

- **temperature = 0.0**
  Temperature at a given node for a "Temperature" load.

- **temperatureDefault = 0.0**
  Default temperature at a node not explicitly being used for a "Temperature" load.

0.12.2 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!!!!

0.13 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).

0.13.1 JSON String Dictionary

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

- **analysisType = "Modal"**
  Type of load. Options: "Modal", "Static", "AeroelasticTrim", "AeroelasticFlutter" Note: "AeroelasticStatic" is still supported but refers to "AeroelasticTrim" Note: "Optimization" and "StaticOpt" are not valid - Optimization is initialized by the Analysis_Type AIM Input

- **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",...].

- **analysisDesignConstraint = "(no default)"**
  Single or list of "Design Constraint Name"s defined in FEA Design Constraints 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 renormalization. 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 eigen↩ Normaliztion = "POINT"

- **machNumber = 0.0 or [0.0, ..., 0.0]**
  Mach number used in trim analysis OR Mach up to 6 values used in flutter analysis..

- **dynamicPressure = 0.0**
  Dynamic pressure used in trim analysis.

- **density = 0.0**
  Density used in trim analysis to determine true velocity, or flutter analysis.

- **aeroSymmetryXY = "(no default)"**
  Aerodynamic symmetry about the XY Plane. Options: SYM, ANTISYM, ASYM. Aerodynamic symmetry about the XY Plane. Options: SYM, ANTISYM, ASYM. SYMMETRIC Indicates that a half span aerodynamic model is moving in a symmetric manner with respect to the XY plane. ANTISYMMETRIC Indicates that a half span aerodynamic model is moving in an antisymmetric manner with respect to the XY plane. ASYMMETRIC Indicates that a full aerodynamic model is provided.

- **aeroSymmetryXZ = "(no default)"**
  Aerodynamic symmetry about the XZ Plane. Options: SYM, ANTISYM, ASYM. SYMMETRIC Indicates that a half span aerodynamic model is moving in a symmetric manner with respect to the XZ plane. ANTISYMMETRIC Indicates that a half span aerodynamic model is moving in an antisymmetric manner with respect to the XZ plane. ASYMMETRIC Indicates that a full aerodynamic model is provided.

- **rigidVariable = ["no default"]**
  List of rigid body motions to be used as trim variables during a trim analysis. Nastran format labels are used and will be converted by the AIM automatically. Expected inputs: ANGLEA, SIDES, ROLL, PITCH, YAW, URDD1, URDD2, URDD3, URDD4, URDD5, URDD6

- **rigidConstraint = ["no default"]**
  List of rigid body motions to be used as trim constraint variables during a trim analysis. Nastran format labels are used and will be converted by the AIM automatically. Expected inputs: ANGLEA, SIDES, ROLL, PITCH, YAW, URDD1, URDD2, URDD3, URDD4, URDD5, URDD6

- **magRigidConstraint = [0.0 , 0.0, ...]**
  List of magnitudes of trim constraint variables. If none and 'rigidConstraint'(s) are specified then 0.0 is assumed for each rigid constraint.

- **controlConstraint = ["no default"]**
  List of controls surfaces to be used as trim constraint variables during a trim analysis.

- **magControlConstraint = [0.0 , 0.0, ...]**
  List of magnitudes of trim control surface constraint variables. If none and 'controlConstraint'(s) are specified then 0.0 is assumed for each control surface constraint.

- **reducedFreq = [0.1, ..., 20.0], No Default Values are defined.**
  Reduced Frequencies to be used in Flutter Analysis. Up to 8 values can be defined.

- **flutterVel = [0.1, ..., 20.0]**
  Velocities to be used in Flutter Analysis. If no values are provided the following relation is used

  v = sqrt(2∗dynamicPressure/density) dv = (v∗2 - v/2) / 20;

  flutterVel[0] = v/10 flutterVel[i] = v/2 + i∗dv; where i = 1....21 flutterVel[22] = v∗10;

- **flutterConvergenceParam = 1e-5**
  Convergence parameter for flutter eigenvalue.

- **visualFlutter = False**
  Turn on flutter visualization f06 output.

0.13.2 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!!!!

0.14 FEA Design Variables

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).

0.14.1 JSON String Dictionary

If "Value" is JSON string dictionary (eg. "Value" = {"initialValue": 5.0, "upperBound": 10.0}) the following keywords (= default values) may be used:

- **initialValue = 0.0**
  Initial value for the design variable.

- **lowerBound = 0.0**
  Lower bound for the design variable.

- **upperBound = 0.0**
  Upper bound for the design variable.

- **maxDelta = 0.0**
  Move limit for the design variable.

- **discreteValue = 0.0**
  List of discrete values to use for the design variable (e.g. [0.0,1.0,1.5,3.0].

0.15 FEA DesignVariableRelation

Structure for the design variable tuple = ("DesignVariableRelation Name", "Value"). "DesignVariableRelation 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).

0.15.1 JSON String Dictionary

If "Value" is JSON string dictionary (eg. "Value" = {"componentType": "Property", "componentName": "plate", "variableName": "MyDesVar"}) the following keywords (= default values) may be used:

- **componentType = "Property"**
  The type of component for this design variable relation. Options: "Property".

- **componentName = "(no default)"**
  Single FEA Property linked to the design variable (e.g. "Name1").
  - For `componentType` Property a FEA Property name (or names) is given.

- **variableName = "(no default)"**
  Name of design variable linked to this relation

- **constantCoeff = 0.0**
  Constant term of relation.

- **linearCoeff = 1.0**
  Single or list of coefficients of linear relation. Must be same length as `variableName`.

0.16 FEA Design Constraints

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).

0.16.1 JSON String Dictionary

If "Value" is JSON string dictionary (eg. "Value" = {"groupName": "plate", "upperBound": 10.0}) the following keywords (= default values) may be used:

- **groupName = "(no default)"**
  Single or list of `capsGroup` name(s) to the design variable (e.g. "Name1" or ["Name1","Name2",...].The property (see FEA Property) also assigned to the same `capsGroup` will be automatically related to this constraint entry.

- **constraintType = "Property"**
  The type of design constraint. Options: "Property", "Flutter"

- **lowerBound = 0.0**
  Lower bound for the design constraint.

- **upperBound = 0.0**
  Upper bound for the design constraint.

- **responseType = "(no default)"**
  Response type options for DRESP1 Entry (see Nastran manual).

  – Implemented Options

    1. `STRESS`, for `propertyType = "Rod" or "Shell"` (see FEA Property)
    2. `CFAILURE`, for `propertyType = "Composite"` (see FEA Property)

- **fieldName = "(no default)"**
  For constraints, this field is only used currently when applying constraints to composites. This field is used to identify the specific lamina in a stacking sequence that a constraint is being applied too. Note if the user has design variables for both THEATA1 and T1 it is likely that only a single constraint on the first lamina is required. For this reason, the user can simply enter LAMINA1 in addition to the possible entries defined in the FEA Design Variables section. Additionally, the `fieldPosition` integer entry below can be used. In this case `"LAMINA1" = 1`.

  – -# **Property Types** (see FEA Property)

    * **PCOMP** `propertyType = "Composite"`
      · "T1", "THETA1", "T2", "THETA2", ... "Ti", "THETAi"
      · "LAMINA1", "LAMINA2", ... "LAMINAi"

- **fieldPosition = 0**
  This input is ignored if not defined. The user may use this field instead of the `fieldName` input defined above to identify a specific lamina in a composite stacking sequence where a constraint is applied. Please read the `fieldName` information above for more information.

- **velocityType = "TRUE"**
  The nature of the velocity values defined with "velocity" keyword. Can be either "TRUE" for true velocity or "EQUIV" for equivalent air speed.

- **scalingFactor = 0.10**
  The constraint scaling factor.

- **velocity = (No Default)**
  The velocity values for flutter constraint. Must be an array with equal length to defined "damping" values.

- **damping = (No Default)**
  The damping values for flutter constraint. Must be an array with equal length to defined "velocity" values.

0.17 FEA Optimization Control

Structure for the optimization control dictionary = ‘Value’. The "Value" must be a JSON String dictionary (see Section JSON String Dictionary).

0.18 FEA Design Equations

Structure for the design equation tuple = ("DesignEquation Name", ["Value1", ... , "ValueN"]). "DesignEquation Name" defines the reference name for the design equation being specified. This string will be used in the FEA input directly. The values "Value1", ... , "ValueN" are a list of strings containing the equation defintions. (see Section List of equation strings).

0.18.1 List of equation strings

Each design equation tuple value is a list of strings containing the equation definitions

0.19 FEA Table Constants

Structure for the table constant tuple = ("TableConstant Name", "Value"). "TableConstant Name" defines the reference name for the table constant being specified. This string will be used in the FEA input directly. The "Value" is the value of the table constant.

0.20 FEA Design Responses

Structure for the design response tuple = ("DesignResponse Name", "Value"). "DesignResponse Name" defines the reference name for the design response 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).

0.20.1 JSON String Dictionary

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

0.21 FEA Design Equation Responses

Structure for the design equation response tuple = ("DesignEquationResponse Name", "Value"). "DesignEquation↩ Response Name" defines the reference name for the design equation response 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).

0.21.1 JSON String Dictionary

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

0.22 FEA Design Optimization Parameters

Structure for the design optimization parameter tuple = ("DesignOptParam Name", "Value"). "DesignOptParam Name" defines the reference name for the design optimization parameter being specified. This string will be used in the FEA input directly. The "Value" is the value of the design optimization parameter.

0.23 FEA Aerodynamic References

The aerodynamic reference input must be a JSON String dictionary (see Section JSON String Dictionary).

0.23.1 JSON String Dictionary

The following keywords (= default values) may be used:

- **referenceNode**
  Defines the reference `capsReference` for the node to be used for stability derivative calculations.

0.24 Vortex Lattice Surface

Structure for the Vortex Lattice Surface tuple = ("Name of Surface", "Value"). "Name of surface defines the name of the surface in which the data should be applied. The "Value" can either be a JSON String dictionary (see Section JSON String Dictionary) or a single string keyword string (see Section Single Value String).

0.24.1 JSON String Dictionary

If "Value" is a JSON string dictionary (eg. "Value" = {"numChord": 5, "spaceChord": 1.0, "numSpan": 10, "space↩ Span": 0.5}) the following keywords (= default values) may be used:

- **groupName = "(no default)"**
  Single or list of *capsGroup* names used to define the surface (e.g. "Name1" or ["Name1","Name2",...]. If no groupName variable is provided an attempted will be made to use the tuple name instead;

- **numChord = 10**
  The number of chordwise horseshoe vortices placed on the surface.

- **spaceChord = 0.0**
  The chordwise vortex spacing parameter.

- **numSpanTotal = 0**
  Total number of spanwise horseshoe vortices placed on the surface. The vorticies are 'evenly' distributed across sections to minimize jumps in spacings. numpSpanPerSection must be zero if this is set.

- **numSpanPerSection = 0**
  The number of spanwise horseshoe vortices placed on each section the surface. The total number of spanwise vorticies are (numSection-1)$*$numSpanPerSection. The vorticies are 'evenly' distributed across sections to minimize jumps in spacings. numSpanTotal must be zero if this is set.

- **spaceSpan = 0.0**
  The spanwise vortex spacing parameter.

- **surfaceType = "Wing"**
  Type of aerodynamic surface being described: "Wing", "Canard", "Tail".

0.24.2 Single Value String

If "Value" is a single string the following options maybe used:

- (NONE Currently)

0.25 Vortex Lattice Control Surface

Structure for the Vortex Lattice Control Surface tuple = ("Name of Control Surface", "Value"). "Name of control surface defines the name of the control surface in which the data should be applied. The "Value" must be a JSON String dictionary (see Section JSON String Dictionary).

0.25.1 JSON String Dictionary

If "Value" is a JSON string dictionary (e.g. "Value" = {"deflectionAngle": 10.0}) the following keywords (= default values) may be used:

- **surfaceSymmetry = "SYM"**
  The surface type of symmetry.

0.25.2 Single Value String

If "Value" is a single string, the following options maybe used:

- (NONE Currently)