CART3D Analysis Interface Module (AIM) Manual

December 8, 2022

0.1 Introduction

0.1.1 Cart3D AIM Overview

Cart3D

0.1.2 Assumptions

This documentation contains four sections to document the use of the CART3D AIM. Cart3D Examples contains example ∗.csm input files and pyCAPS scripts designed to make use of the CART3D AIM. These example scripts make extensive use of the Cart3D attributes and Cart3D AIM Inputs and Cart3D AIM Outputs.

The Cart3D AIM can automatically execute Cart3D, with details provided in AIM Execution.

Details of the AIM's automated data transfer capabilities are outlined in Cart3D Data Transfer

0.1.3 Dependencies

ESP client of libxddm. For XDDM documentation, see $CART3D/doc/xddm/xddm.html. The library uses XML Path Language (XPath) to navigate the elements of XDDM documents. For XPath tutorials, see the web, e.g. www.↩ developer.com/net/net/article.php/3383961/NET-and-XML-XPath-Queries.htm

Dependency: libxml2, www.xmlsoft.org. This library is usually present on most systems, check existence of 'xml2-config' script

0.2 Cart3D Examples

This example contains a set of ∗.csm and pyCAPS (∗.py) inputs that uses the Cart3D AIM. A user should have knowledge on the generation of parametric geometry in Engineering Sketch Pad (ESP) before attempting to integrate with any AIM. Specifically, this example makes use of Design Parameters, Set Parameters, User Defined Primitive (UDP) and attributes in ESP.

0.3 Cart3D attributes

The following list of attributes drives the Cart3D geometric definition.

- **capsAIM** This attribute is a CAPS requirement to indicate the analysis the geometry representation supports.
  - **capsReferenceArea** This attribute may exist on any *Body*. Its value will be used as the Reference_Area entry in the Cart3D input.
  - **capsReferenceChord** This attribute may exist on any *Body*. Its value will be used as the Reference↩ _Length entry in the Cart3D input.
  - **capsReferenceX** This attribute may exist on any *Body*. Its value will be used in the Moment_Point entry in the Cart3D input.
  - **capsReferenceY** [Optional: Default 0.0] This attribute may exist on any *Body*. Its value will be used in the Moment_Point entry in the Cart3D input.
  - **capsReferenceZ** [Optional: Default 0.0] This attribute may exist on any *Body*. Its value will be used in the Moment_Point entry in the Cart3D input.
- **cart3d_BBox** [Optional] This attribute may exist on any *Body*. If present, the bounding box of the body is used to define a BBox region in the preSpec.c3d.cntl file (the body it self is ignored). The BBox refinement level is given by the cart3d_BBox value. Note: This will replace the preSpec.c3d.cntl file generated by auto↩ Inputs.

0.4 Cart3D Data Transfer

0.4.1 Data transfer to Cart3D (FieldIn)

- **"Displacement"**
  Retrieves nodal displacements (as from a structural solver) and updates Cart3D's surface mesh.

The Cart3D AIM has the ability to transfer surface data (e.g. pressure distributions) to and from the AIM using the conservative and interpolative data transfer schemes in CAPS. Currently these transfers may only take place on triangular meshes.

0.4.2 Data transfer from Cart3D (FieldOut)

- **"Pressure"**
  Loads the pressure distribution from Components.i.trix file. This distribution may be scaled based on Pressure = Pressure_Scale_Factor∗Pressure, where "Pressure_Scale_Factor" is an AIM input (AIM Inputs)

0.5 AIM Inputs

The following list outlines the Cart3D inputs along with their default value available through the AIM interface.

- **Tess_Params = [double, double, double]**. <Default [0.025, 0.001, 15.00]>
  These parameters are used to create the surface mesh for Cart3D. Their order definition is as follows.
  1. Max Edge Length (0 is any length)
  2. Max Sag or distance from mesh segment and actual curved geometry
  3. Max angle in degrees between triangle facets
- **mesh2d = bool**. <Default false>
  Specify 2D analysis
- **outer_box = double**. <Default 30>
  Factor of outer boundary box based on geometry length scale defined by the diagonal of the 3D tightly fitting bounding box around body being modeled.
- **nDiv = int**. <Default 5>
  nominal # of divisions in backgrnd mesh
- **maxR = int**. <Default 7>
  Max Num of cell refinements to perform
- **preSpec = Bool**. <Default false>
  Use preSpec.c3d.cntl to guide mesh generation. Enabled automatically if cart3d_preSpec bodies are provided.
- **symmX = bool**. <Default false>
  Symmetry on x-min boundary
- **symmY = int**. <Default false>
  Symmetry on y-min boundary
- **symmZ = int**. <Default false>
  Symmetry on z-min boundary

- **halfBody = bool**. <Default false>
  Input geometry is a half-body
- **Mach = NULL**. <Default 0.76>
- **alpha = double**. <Default 0.0>
  Angle of Attach in Degrees
- **beta = double**. <Default 0.0>
  Side Slip Angle in Degrees
- **gamma = double**. <Default 1.4>
  Ratio of specific heats (default is air)
- **Pressure_Scale_Factor = 1.0**
  Value to scale Pressure data when transferring data. Data is scaled based on Pressure = Pressure_Scale↩
  _Factor∗Pressure.
- **TargetCL = NULL**.
  Target lift coefficient. If set, Cart3D will adjust alpha to such that CL matches TargetCL.
  See TargetCL_ inputs for additional controls.
- **TargetCL_Tol = double**. <Default 0.01>
  Target lift coefficient tolerance
- **TargetCL_Start_Iter = int**. <Default 60>
  Iteration for first Target lift coefficient alpha adjustment
- **TargetCL_Freq = int**. <Default 5>
  Iteration frequency for Target lift coefficient alpha adjustment
- **TargetCL_NominalAlphaStep = double**. <Default 0.2>
  Initial alpha step and max step size in Degrees
- **maxCycles = int**. <Default 1000>
  Number of iterations
- **nMultiGridLevels = int**. <Default 1>
  number of multigrid levels in the mesh (1 is a single mesh)
- **MultiGridCycleType = int**. <Default 2>
  MultiGrid cycletype: 1 = "V-cycle", 2 = "W-cycle" 'sawtooth' cycle is: V-cycle with MultiGridPreSmoothing = 1, MultiGridPostSmoothing = 0
- **MultiGridPreSmoothing = int**. <Default 1>
  number of pre-smoothing passes in multigrid
- **MultiGridPostSmoothing = int**. <Default 1>
  number of post-smoothing passes in multigrid
- **CFL = double**. <Default 1.2>
  CFL number typically between 0.9 and 1.4
- **Limiter = int**. <Default 2>
  organized in order of increasing dissipation.
  0 = no Limiter, 1 = Barth-Jespersen, 2 = van Leer, 3 = sin limiter, 4 = van Albada, 5 = MinMod
- **FluxFun = int**. <Default 0>
  0 = van Leer, 1 = van Leer-Hanel, 2 = Colella 1998, 3 = HLLC (alpha test)
- **iForce = int**. <Default 10>
  Report force & mom. information every iForce cycles
- **iHist = int**. <Default 1>
  Update 'history.dat' every iHist cycles

- **nOrders = int**. <Default 8>
  Num of orders of Magnitude reduction in residual

- **nAdaptCycles = 0**
  Number of adaptation cycles.

- **Adapt_Functional = NULL**
  Single valued tuple that defines the functional used to drive mesh adaptation, see CFD Functional for additional details on functionals.

- **Design_Variable = NULL**
  The design variable tuple is used to input design variable information for optimization, see CFD Design Variable for additional details.

- **Design_Functional = NULL**
  The design objective tuple is used to input objective information for optimization, see CFD Functional for additional details. The value of the design functionals become available as Dynamic Output Value Objects using the "name" of the functionals.

- **Design_Sensitivity = False**
  Create geometric sensitivities Fun3D input files needed to compute Design_Functional sensitivities w.r.↩
  t Design_Variable.

- **Design_Adapt = NULL**
  String name of a Design_Functional to be used for adjoint based mesh adaptation.

- **Design_Run_Config = "production"**
  run_config = debug || archive || standard || production
  debug: no files are deleted; easily traceable but needs large disk-space
  archive: compresses and tars critical files and deletes temp files;
  becomes slower for cases with large number of design variables
  but fully traceable
  standard: similar to production, but keeps more files, in particular adjoint
  solution(s) on the finest mesh
  production: keeps critical files; reasonable storage and max speed

- **Design_Gradient_Memory_Budget = 32**
  This flag controls the parallel evaluation of components of the gradient. For example, gradient of objective function J with respect to design variables x, y, z is [dJ/dx dJ/dy dJ/dz]^T. The gradient components are independent and hence, can be evaluated in parallel. Evaluating all the components simultaneously is ideal, but you are limited by the number of design variables and the size of your mesh (memory limit). The framework can compute efficient job partitioning automatically. To do this, you just need to specify the memory limit for the run (in GB) via the flag

- **Xslices = double** or **[double, ... , double]**
  X slice locations created in output.

- **Yslices = double** or **[double, ... , double]**
  Y slice locations created in output.

- **Zslices = double** or **[double, ... , double]**
  Z slice locations created in output.

- **y_is_spanwise = bool** <Default false>
  If false, then alpha is defined in the x-y plane,
  otherwise alpha is in the x-z plane

- **Model_X_axis = "-Xb"**
  Model_X_axis defines x-axis orientation.

- **Model_Y_axis = "Yb"**
  Model_Y_axis defines y-axis orientation.

- **Model_Z_axis = "-Zb"**
  Model_Z_axis defines z-axis orientation.
- **Restart = False**
  Use the "restart" option for aero.csh or not
- **aerocsh = NULL**
  List of strings that can be used to override defaults in the Cart3D aero.csh script. Please refer to the Cart3D documentation for all available aero.csh inputs.

0.6 AIM Execution

If auto execution is enabled when creating an Cart3D AIM, the AIM will execute Cart3D just-in-time with the command line:

```
./aero.csh > aero.out
```

in the "inputs" analysis directory or

```
c3d_objGrad.csh
```

in the "design" analysis directory when computing functional sensitvities w.r.t. design variables. The AIM preAnalysis generates the inputs/aero.csh script required to execute Cart3D by copying it from $CART3D/bin/aero.csh and inserting various analysis input modifications.

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 Cart3D AIM object. In this mode, caps_execute and Analysis.runAnalysis can be used to run the analysis, or Cart3D can be executed by calling preAnalysis, system, and posAnalysis as demonstrated below with a pyCAPS example:

```
print ("\n\preAnalysis......")
cart.preAnalysis()
print ("\n\nRunning......")
cart.system("./aero.csh", "inputs"); # Run via system call in inputs analysis directory
print ("\n\postAnalysis......")
cart.postAnalysis()
```

0.7 AIM Outputs

Integrated force outputs on the entire body are available as outputs from the loadsCC.dat output file.

- **C_A**. entire Axial Force
- **C_Y**. entire Lateral Force
- **C_N**. entire Normal Force
- **C_D**. entire Drag Force
- **C_S**. entire Side Force
- **C_L**. entire Lift Force
- **C_l**. entire Rolling Moment
- **C_m**. Pitching Moment
- **C_n**. Yawing Moment
- **C_M_x**. X Aero Moment
- **C_M_y**. Y Aero Moment
- **C_M_z**. Z Aero Moment
- **alpha**. Angle of attach (may change using TargetCL)

0.8 CFD Boundary Conditions

Structure for the boundary condition tuple = ("CAPS Group Name", "Value"). "CAPS Group Name" defines the capsGroup on which the boundary condition 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)

0.8.1 JSON String Dictionary

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

- **bcType = "Inviscid"**
  Boundary condition type. Options:
  - Inviscid
  - Viscous
  - Farfield
  - Extrapolate
  - Freestream
  - BackPressure
  - Symmetry
  - SubsonicInflow
  - SubsonicOutflow
  - MassflowIn
  - MassflowOut
  - MachOutflow
  - FixedInflow
  - FixedOutflow

0.8.1.1 Wall Properties

- **wallTemperature = 0.0**
  Temperature on inviscid and viscous surfaces.

- **wallHeatFlux = 0.0**
  Heat flux on inviscid and viscous surfaces.

0.8.1.2 Stagnation Properties

- **totalPressure = 0.0**
  Total pressure on a boundary surface.

- **totalTemperature = 0.0**
  Total temperature on a boundary surface.

- **totalDensity = 0.0**
  Total density of boundary.

0.8.1.3 Static Properties

- **staticPressure = 0.0**
  Static pressure of boundary.

- **staticTemperature = 0.0**
  Static temperature of boundary.

- **staticDensity = 0.0**
  Static density of boundary.

0.8.1.4 Velocity Components

- **uVelocity = 0.0**
  X-velocity component on boundary.

- **vVelocity = 0.0**
  Y-velocity component on boundary.

- **wVelocity = 0.0**
  Z-velocity component on boundary.

- **machNumber = 0.0**
  Mach number on boundary.

0.8.1.5 Massflow Properties

- **massflow = 0.0**
  Massflow through the boundary.

0.8.2 Single Value String

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

- "Inviscid" (default)
- "Viscous"
- "Farfield"
- "Extrapolate"
- "Freestream"
- "SymmetryX"
- "SymmetryY"
- "SymmetryZ"

0.9 CFD Design Variable

Structure for the design variable tuple = ("DesignVariable Name", "Value"). "DesignVariable Name" defines the reference name for the design variable being specified. The "Value" may be a JSON String dictionary (see Section JSON String Dictionary) or just a blank string (see Section Single Value String).

Note that any JSON string inputs are written to the input files as information only. They are only use if the analysis is executed with the analysis specific design framework.

0.9.1 JSON String Dictionary

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

- **lowerBound = 0.0 or [0.0, 0.0,...]**
  Lower bound for the design variable.

- **upperBound = 0.0 or [0.0, 0.0,...]**
  Upper bound for the design variable.

- **typicalSize = 0.0 or [0.0, 0.0,...]**
  Typical size for the design variable.

0.9.2 Single Value String

If "Value" is a string, the string value will be ignored.

0.10 CFD Functional

Structure for the design functional tuple = {"Functional Name": "Expression"}. "Functional Name" is a user specified unique name, and the Expression is a mathematical expression parsed by Cart3D symbolically.

The variables that can be used to form an expression are:

Variables	Description
"cl", "cd"	Lift, drag coefficients
"cmx", "cmy", "cmz"	x/y/z-axis moment coefficients
"cx", "cy", "cz"	x/y/z-axis force coefficients

0.10.1 JSON String Dictionary

If "Value" is a JSON string dictionary (eg. "Value" = "Composite":[{"function": "cl", "weight": 3.0, "target": 10.7}, {"function": "cd", "weight": 2.0, "power": 2.0}])
which represents the composite functional: $Composite = 3(c_l - 10.7) + 2c_d^2$
The following keywords (= default values) may be used:

- **capsGroup = "GroupName"**
  Name of boundary to apply for the function C_i.

- **function = NULL**
  The name of the function C_i, e.g. "cl", "cd", etc.

- **weight = 1.0**
  This weighting w_i of the function.

- **target = 0.0**
  This is the target value C_i^* of the function.

- **power = 1.0**
  This is the user defined power operator p_i for the function.