FAST-OAD_CS25

unknown

May 17, 2024

CONTENTS

WIP

CHAPTER

ONE

CONTENTS

1.1 License

1.2 Contributors

- Christophe DAVID <christophe.david@onera.fr>
- Scott DELBECQ <scott.delbecq@isae-supaero.fr>
- Martin DELAVENNE <martin.delavenne@isae-supaero.fr>

1.3 How to cite us

Please cite this article when using FAST-OAD in your research works:

C. David, S. Delbecq, S. Defoort, P. Schmollgruber, E. Benard and V. Pommier-Budinger: *"From FAST to FAST-OAD: An open source framework for rapid Overall Aircraft Design"*, IOP Conference Series: Materials Science and Engineering, vol. 1024, n. 1, DOI: 10.1088/1757-899x/1024/1/012062

```
@article{David2021,
    doi = {10.1088/1757-899x/1024/1/012062},
    url = {https://doi.org/10.1088/1757-899x/1024/1/012062},
    year = 2021,
    month = {jan},
    publisher = {{IOP} Publishing},
    volume = {1024},
    number = {1},
    pages = {012062},
    author = {Christophe David and Scott Delbecq and Sebastien Defoort and Peter␣
↪Schmollgruber and Emmanuel Benard and Valerie Pommier-Budinger},
    title = {From {FAST} to {FAST}-{OAD}: An open source framework for rapid Overall␣
↪Aircraft Design},
    journal = {{IOP} Conference Series: Materials Science and Engineering}}
```

1.4 Changelog

1.4.1 Version 0.5.0

Changed:

- Allowing deactivation of inner solvers. (#33)
- Having ISA temperature offset as input of engine deck. (#31)

1.4.2 Version 0.4.0

Changed:

- FAST-OAD-CS25 is now officially compatible with Python 3.10. Support of Python 3.7 has been abandoned. (#26)
- Trailing edge of wing inner part can now have a non-zero sweep angle.(#27)
- Now the criteria for computation of wing can be controlled through options. (#29)
- Computation of wing geometry now uses sub-models and has now an option for (de)activating the computation of wing thickness. (#29)

Fixed: - Fixed component for computing global positions of wing chords, which is now a sub-model of geometry module (#28)

1.4.3 Version 0.3.0

Changed:

- OpenMDAO 3.18+ is now required. (#18)
- Submodels have been introduced in module "fastoad.loop.wing_area". (#7)
- Load factors are now explicitly in output data. (#23)

Fixed:

- Unit of variable "data:weight:aircraft:additional_fuel_capacity" has been corrected to "kg". (#18)
- Unit of "*:CL_alpha" variables is now consistently "1/rad". (#21)
- Unit for "data:load_case:lc2:Vc_EAS" has been corrected to "m/s". (#23)

1.4.4 Version 0.2.0

Added:

- Now polar computation in aerodynamics module computes angle of attack as a linear function of CL. (#16)

1.4.5 Version 0.1.4

Fixed:

- Bundled notebooks have been modified to adapt to FAST-OAD 1.4.1, which is now the minimum required version for FAST-OAD-core. (#14)

1.4.6 Version 0.1.3

Fixed:

- in bundled notebooks:
  - Generation of configuration file would fail if several FAST-OAD plugins were installed.
  - Link to CeRAS website has been fixed

1.4.7 Version 0.1.2

Changed:

- Now allowing wing geometry with no kink (#3)

Fixed:

- Fixed deprecation warnings (#4)
- Now allowing versions greater than 0.1 for StdAtm

1.4.8 Version 0.1.1

- Fixed dependency to FAST-OAD

1.4.9 Version 0.1.0

- FAST-OAD CS-25 related models are now in this separate package

1.5 General documentation

Here you will find the first things to know about FAST-OAD CS25.

1.5.1 Installation procedure

WIP

1.6 fastoad_cs25

1.6.1 fastoad_cs25 package

Subpackages

fastoad_cs25.configurations package

Module contents

fastoad_cs25.models package

Subpackages

fastoad_cs25.models.aerodynamics package

Subpackages

fastoad_cs25.models.aerodynamics.components package

Subpackages

fastoad_cs25.models.aerodynamics.components.utils package

Submodules

fastoad_cs25.models.aerodynamics.components.utils.cd0_lifting_surface module

Computation of CD0 for a lifting surface.

class fastoad_cs25.models.aerodynamics.components.utils.cd0_lifting_surface.**LiftingSurfaceGeometry**(*thickn*
float,
*MAC_*
float,
sweep
float,
cam-
bered
bool,
wet_a
float,
in-
ter-
ac-
tion_c
float)

Bases: `object`

Minimum geometry data for computation of CD0 of lifting surfaces.

thickness_ratio: *float*
average thickness ratio

MAC_length: *float*
length of Mean Aerodynamic Chord

sweep_angle_25: *float*
sweep angle at 25% chord, in degrees

cambered: *bool*
True if airfoil is cambered

wet_area: *float*
wet surface area of the lifting surface

interaction_coeff: *float*
ratio of additional drag due to interaction effects

fastoad_cs25.models.aerodynamics.components.utils.cd0_lifting_surface.**compute_cd0_lifting_surface**(*geomet* fas- toad_c *mach: float*, *reynol float*, *wing_a float*, *lift_coe float* = *0.0*)

Computes CD0 for a lifting surface.

Friction coefficient is assessed from [Ray99] (Eq 12.27). Corrections for lifting surfaces are from [DCAC14].

Parameters

- **geometry** – definition of lifting surface geometry
- **mach** – Mach number
- **reynolds** – Reynolds number
- **wing_area** – wing area (will be used for getting CD specific to wing area
- **lift_coefficient** – needed if wing is cambered

Returns CD0 value

fastoad_cs25.models.aerodynamics.components.utils.friction_drag module

Computation of friction drag.

fastoad_cs25.models.aerodynamics.components.utils.friction_drag.**get_flat_plate_friction_drag_coefficien**

Parameters

- **length** – flat plate length in meters

- **mach** – Mach number
- **reynolds** – Reynolds number

Returns Drag coefficient w.r.t. a surface of area length*1 m**2

Module contents

Submodules

fastoad_cs25.models.aerodynamics.components.cd0 module

Computation of form drag for whole aircraft.

class fastoad_cs25.models.aerodynamics.components.cd0.**CD0**(***kwargs*)

Bases: openmdao.core.group.Group

Computation of form drag for whole aircraft.

Computes and sums the drag coefficients of all components. Interaction drag is assumed to be taken into account at component level.

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

initialize()

Perform any one-time initialization run at instantiation.

setup()

Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

fastoad_cs25.models.aerodynamics.components.cd0_fuselage module

Computation of form drag for fuselage.

class fastoad_cs25.models.aerodynamics.components.cd0_fuselage.**Cd0Fuselage**(***kwargs*)

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computation of form drag for fuselage.

Store some bound methods so we can detect runtime overrides.

initialize()

Perform any one-time initialization run at instantiation.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.cd0_ht module

Computation of form drag for Horizontal Tail Plane.

class fastoad_cs25.models.aerodynamics.components.cd0_ht.**Cd0HorizontalTail**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computation of form drag for Horizontal Tail Plane.

See *cd0_lifting_surface()* for used method.

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.cd0_nacelles_pylons module

Computation of form drag for nacelles and pylons.

class fastoad_cs25.models.aerodynamics.components.cd0_nacelles_pylons.**Cd0NacellesAndPylons**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computation of form drag for nacelles and pylons.

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.cd0_total module

Sum of form drags from aircraft components.

class fastoad_cs25.models.aerodynamics.components.cd0_total.**Cd0Total**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computes the sum of form drags from aircraft components.

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.cd0_vt module

Computation of form drag for Vertical Tail Plane.

class fastoad_cs25.models.aerodynamics.components.cd0_vt.**Cd0VerticalTail**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computation of form drag for Vertical Tail Plane.

See *cd0_lifting_surface()* for used method.

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.cd0_wing module

Computation of form drag for wing.

class fastoad_cs25.models.aerodynamics.components.cd0_wing.**Cd0Wing**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computation of form drag for wing.

See *cd0_lifting_surface()* for used method.

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.cd_compressibility module

Compressibility drag computation.

class fastoad_cs25.models.aerodynamics.components.cd_compressibility.**CdCompressibility**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computation of drag increment due to compressibility effects.

Formula from §4.2.4 of [DCAC14]. This formula can be used for aircraft before year 2000.

Earlier aircraft have more optimized wing profiles that are expected to limit the compressibility drag below 2 drag counts. Until a better model can be provided, the variable *tuning:aerodynamics:aircraft:cruise:CD:compressibility:characteristic_mach_increment* allows to move the characteristic Mach number, thus moving the CD divergence to higher Mach numbers.

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.cd_trim module

Computation of trim drag.

class fastoad_cs25.models.aerodynamics.components.cd_trim.**CdTrim**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computation of trim drag.

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.compute_alpha module

Computation of CL characteristics at low speed.

class fastoad_cs25.models.aerodynamics.components.compute_alpha.**ComputeAlpha**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computes a linear evolution of angle of attack from CL scale and CL gradient.

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.compute_cl_alpha module

Computation of CL characteristics at low speed.

class fastoad_cs25.models.aerodynamics.components.compute_cl_alpha.**ComputeCLAlpha**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computes CL gradient.

CL gradient from [Ray99] Eq 12.6

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.compute_max_cl_landing module

Computation of max CL in landing conditions.

class fastoad_cs25.models.aerodynamics.components.compute_max_cl_landing.**ComputeMaxClLanding**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computation of max CL in landing conditions.

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.compute_polar module

Computation of CL and CD for whole aircraft.

class fastoad_cs25.models.aerodynamics.components.compute_polar.**ComputePolar**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computation of CL and CD for whole aircraft.

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.compute_polar.**get_optimum_ClCd**(*ClCd*)

fastoad_cs25.models.aerodynamics.components.compute_reynolds module

Computation of Reynolds number

class fastoad_cs25.models.aerodynamics.components.compute_reynolds.**ComputeReynolds**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computation of Reynolds number

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.high_lift_aero module

Computation of lift and drag increment due to high-lift devices

class fastoad_cs25.models.aerodynamics.components.high_lift_aero.**ComputeDeltaHighLift**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Provides lift and drag increments due to high-lift devices

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.initialize_cl module

Initialization of CL vector.

class fastoad_cs25.models.aerodynamics.components.initialize_cl.**InitializeClPolar**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Initialization of CL vector.

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.components.oswald module

Computation of Oswald coefficient

class fastoad_cs25.models.aerodynamics.components.oswald.**InducedDragCoefficient**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computes the coefficient that should be multiplied by CL**2 to get induced drag.

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

class fastoad_cs25.models.aerodynamics.components.oswald.**OswaldCoefficient**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computes Oswald efficiency number

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

Module contents

fastoad_cs25.models.aerodynamics.external package

Subpackages

fastoad_cs25.models.aerodynamics.external.xfoil package

Subpackages

fastoad_cs25.models.aerodynamics.external.xfoil.xfoil699 package

Module contents

Submodules

fastoad_cs25.models.aerodynamics.external.xfoil.xfoil_polar module

This module launches XFOIL computations

class fastoad_cs25.models.aerodynamics.external.xfoil.xfoil_polar.**XfoilPolar**(***kwargs*)
Bases: openmdao.components.external_code_comp.ExternalCodeComp

Runs a polar computation with XFOIL and returns the 2D max lift coefficient

Intialize the ExternalCodeComp component.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

compute(*inputs*, *outputs*)
Run this component.

User should call this method from their overriden compute method.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].

Module contents

Module for OpenMDAO-embedded XFOIL

Module contents

Submodules

fastoad_cs25.models.aerodynamics.aerodynamics_high_speed module

Computation of aerodynamic polar in cruise conditions.

class fastoad_cs25.models.aerodynamics.aerodynamics_high_speed.**AerodynamicsHighSpeed**(***kwargs*)
Bases: openmdao.core.group.Group

Computes aerodynamic polar of the aircraft in cruise conditions.

Drag contributions of each part of the aircraft are computed though analytical models.

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

fastoad_cs25.models.aerodynamics.aerodynamics_landing module

Aero computation for landing phase

class fastoad_cs25.models.aerodynamics.aerodynamics_landing.**AerodynamicsLanding**(***kwargs*)
Bases: openmdao.core.group.Group

Computes aerodynamic characteristics at landing.

- Computes CL and CD increments due to high-lift devices at landing.
- Computes maximum CL of the aircraft in landing conditions.

Maximum 2D CL without high-lift is computed using XFoil (or provided as input if option use_xfoil is set to False). 3D CL is deduced using sweep angle.

Contribution of high-lift devices is modelled according to their geometry (span and chord ratio) and their deflection angles.

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

class fastoad_cs25.models.aerodynamics.aerodynamics_landing.**ComputeMachReynolds**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Mach and Reynolds computation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

class fastoad_cs25.models.aerodynamics.aerodynamics_landing.**Compute3DMaxCL**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computes 3D max CL from 2D CL (XFOIL-computed) and sweep angle

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.aerodynamics.aerodynamics_low_speed module

Computation of aerodynamic polar in low speed conditions.

class fastoad_cs25.models.aerodynamics.aerodynamics_low_speed.**AerodynamicsLowSpeed**(***kwargs*)
Bases: openmdao.core.group.Group

Models for low speed aerodynamics

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

fastoad_cs25.models.aerodynamics.aerodynamics_takeoff module

Computation of aerodynamic characteristics at takeoff.

class fastoad_cs25.models.aerodynamics.aerodynamics_takeoff.**AerodynamicsTakeoff**(***kwargs*)

Bases: openmdao.core.group.Group

Computes aerodynamic characteristics at takeoff.

- Computes CL and CD increments due to high-lift devices at takeoff.

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()

Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

fastoad_cs25.models.aerodynamics.constants module

Constants for aerodynamics models.

class fastoad_cs25.models.aerodynamics.constants.**PolarType**(*value=<no_arg>*, *names=None*, *module=None*, *type=None*, *start=1*, *boundary=None*)

Bases: aenum.Enum

Enumeration of polar types to be computed.

HIGH_SPEED = 'high_speed'

LOW_SPEED = 'low_speed'

TAKEOFF = 'takeoff'

LANDING = 'landing'

Module contents

fastoad_cs25.models.geometry package

Subpackages

fastoad_cs25.models.geometry.geom_components package

Subpackages

fastoad_cs25.models.geometry.geom_components.fuselage package

Submodules

fastoad_cs25.models.geometry.geom_components.fuselage.compute_cnbeta_fuselage module

Estimation of yawing moment due to sideslip

class fastoad_cs25.models.geometry.geom_components.fuselage.compute_cnbeta_fuselage.**ComputeCnBetaFusela**

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Yawing moment due to sideslip estimation

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.fuselage.compute_fuselage module

Estimation of geometry of fuselase part A - Cabin (Commercial)

class fastoad_cs25.models.geometry.geom_components.fuselage.compute_fuselage.**ComputeFuselageGeometryBas**
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Geometry of fuselage part A - Cabin (Commercial) estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

class fastoad_cs25.models.geometry.geom_components.fuselage.compute_fuselage.**ComputeFuselageGeometryCab**
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Geometry of fuselage part A - Cabin (Commercial) estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

Module contents

Estimation of fuselage geometry

fastoad_cs25.models.geometry.geom_components.ht package

Subpackages

fastoad_cs25.models.geometry.geom_components.ht.components package

Submodules

fastoad_cs25.models.geometry.geom_components.ht.components.compute_ht_chords module

Estimation of horizontal tail chords and span

class fastoad_cs25.models.geometry.geom_components.ht.components.compute_ht_chords.**ComputeHTChord**(**kwa

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Horizontal tail chords and span estimation

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.ht.components.compute_ht_cl_alpha module

Estimation of horizontal tail lift coefficient

class fastoad_cs25.models.geometry.geom_components.ht.components.compute_ht_cl_alpha.**ComputeHTClalpha**(*

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Horizontal tail lift coefficient estimation

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.ht.components.compute_ht_mac module

Estimation of horizontal tail mean aerodynamic chords

class fastoad_cs25.models.geometry.geom_components.ht.components.compute_ht_mac.**ComputeHTMAC**(***kwargs*)

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Horizontal tail mean aerodynamic chord estimation

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].

- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.ht.components.compute_ht_sweep module

Estimation of horizontal tail sweeps

class fastoad_cs25.models.geometry.geom_components.ht.components.compute_ht_sweep.**ComputeHTSweep**(**kwarg

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Horizontal tail sweeps estimation

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

Module contents

Estimation of horizontal tail geometry (components)

Submodules

fastoad_cs25.models.geometry.geom_components.ht.compute_horizontal_tail module

Estimation of geometry of horizontal tail

class fastoad_cs25.models.geometry.geom_components.ht.compute_horizontal_tail.**ComputeHorizontalTailGeom**

Bases: openmdao.core.group.Group

Horizontal tail geometry estimation

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

Module contents

Estimation of horizontal tail geometry (global)

fastoad_cs25.models.geometry.geom_components.nacelle_pylons package

Subpackages

Submodules

fastoad_cs25.models.geometry.geom_components.nacelle_pylons.compute_nacelle_pylons module

Estimation of nacelle and pylon geometry

class fastoad_cs25.models.geometry.geom_components.nacelle_pylons.compute_nacelle_pylons.**Chord**(*x: float*, *y: float*, *length: float*)

Bases: object

Container for storing chord length and x,y coordinates of leading edge.

x: float

y: float

length: float

class fastoad_cs25.models.geometry.geom_components.nacelle_pylons.compute_nacelle_pylons.**Nacelle**(*max_thr*
float)

Bases: object

Simple class for computing nacelle geometry

max_thrust: float

property diameter
Nacelle diameter in m.

property length
Nacelle length in m.

property wetted_area
Wetted area for one nacelle in m**2.

class fastoad_cs25.models.geometry.geom_components.nacelle_pylons.compute_nacelle_pylons.ComputeNacelle
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Nacelle and pylon geometry estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

Module contents

Estimation of nacelle and pylons

fastoad_cs25.models.geometry.geom_components.vt package

Subpackages

fastoad_cs25.models.geometry.geom_components.vt.components package

Submodules

fastoad_cs25.models.geometry.geom_components.vt.components.compute_vt_chords module

Estimation of vertical tail chords and span

class fastoad_cs25.models.geometry.geom_components.vt.components.compute_vt_chords.ComputeVTChords(**kw
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Vertical tail chords and span estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.vt.components.compute_vt_clalpha module

Estimation of vertical tail lift coefficient

class fastoad_cs25.models.geometry.geom_components.vt.components.compute_vt_clalpha.**ComputeVTClalpha**(**
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Vertical tail lift coefficient estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.vt.components.compute_vt_distance module

Estimation of vertical tail distance

class fastoad_cs25.models.geometry.geom_components.vt.components.compute_vt_distance.**ComputeVTDistance**(
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Vertical tail distance estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.vt.components.compute_vt_mac module

Estimation of vertical tail mean aerodynamic chords

class fastoad_cs25.models.geometry.geom_components.vt.components.compute_vt_mac.**ComputeVTMAC**(**kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Vertical tail mean aerodynamic chord estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].

- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.vt.components.compute_vt_sweep module

Estimation of vertical tail sweeps

class fastoad_cs25.models.geometry.geom_components.vt.components.compute_vt_sweep.**ComputeVTSweep**(**kwarg

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Vertical tail sweeps estimation

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

Module contents

Estimation of vertical tail geometry (components)

Submodules

fastoad_cs25.models.geometry.geom_components.vt.compute_vertical_tail module

Estimation of geometry of vertical tail

class fastoad_cs25.models.geometry.geom_components.vt.compute_vertical_tail.**ComputeVerticalTailGeometry**

Bases: openmdao.core.group.Group

Vertical tail geometry estimation

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

Module contents

Estimation of vertical tail geometry (global)

fastoad_cs25.models.geometry.geom_components.wing package

Subpackages

fastoad_cs25.models.geometry.geom_components.wing.components package

Subpackages

Submodules

fastoad_cs25.models.geometry.geom_components.wing.components.compute_b_50 module

Estimation of wing B50

class fastoad_cs25.models.geometry.geom_components.wing.components.compute_b_50.ComputeB50(**kwargs)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Wing B50 estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].

- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.wing.components.compute_l1_l4 module

Estimation of wing chords (l1 and l4)

class fastoad_cs25.models.geometry.geom_components.wing.components.compute_l1_l4.**ComputeL1AndL4Wing**(**kw

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Wing chords (l1 and l4) estimation

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.wing.components.compute_l2_l3 module

Estimation of wing chords (l2 and l3)

class fastoad_cs25.models.geometry.geom_components.wing.components.compute_l2_l3.**ComputeL2AndL3Wing**(**kw

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Wing chords (l2 and l3) estimation

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.wing.components.compute_mac_wing module

Estimation of wing mean aerodynamic chord

class fastoad_cs25.models.geometry.geom_components.wing.components.compute_mac_wing.**ComputeMACWing**(**kw
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Wing mean aerodynamic chord estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.wing.components.compute_mfw module

Estimation of max fuel weight

class fastoad_cs25.models.geometry.geom_components.wing.components.compute_mfw.**ComputeMFW**(**kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Max fuel weight estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.wing.components.compute_planform module

Submodel for computing wing planform.

class fastoad_cs25.models.geometry.geom_components.wing.components.compute_planform.**ComputeWingGeometry**
Bases: fastoad.openmdao.base_model_classes.CycleGroup

Computation of wing planform

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

fastoad_cs25.models.geometry.geom_components.wing.components.compute_sweep_wing module

Estimation of wing sweeps

class fastoad_cs25.models.geometry.geom_components.wing.components.compute_sweep_wing.**ComputeSweepWing**(
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Wing sweeps estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.wing.components.compute_toc_wing module

Estimation of wing ToC

class fastoad_cs25.models.geometry.geom_components.wing.components.compute_toc_wing.**ComputeToCWing**(***kw*
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Wing ToC estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.wing.components.compute_wet_area_wing module

Estimation of wing wet area

class fastoad_cs25.models.geometry.geom_components.wing.components.compute_wet_area_wing.**ComputeWetArea**
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Wing wet area estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.wing.components.compute_x_wing module

Estimation of wing Xs

class fastoad_cs25.models.geometry.geom_components.wing.components.compute_x_wing.**ComputeXWing**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Wing Xs estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.geom_components.wing.components.compute_y_wing module

Estimation of wing Ys (sections span)

class fastoad_cs25.models.geometry.geom_components.wing.components.compute_y_wing.**ComputeYWing**(***kwargs*)

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Wing Ys estimation

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

Module contents

Estimation of wing geometry (components)

Submodules

fastoad_cs25.models.geometry.geom_components.wing.compute_wing module

Estimation of wing geometry

class fastoad_cs25.models.geometry.geom_components.wing.compute_wing.**ComputeWingGeometry**(***kwargs*)

Bases: openmdao.core.group.Group

Wing geometry estimation

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

fastoad_cs25.models.geometry.geom_components.wing.constants module

Constants for identifiers of wing geometry submodel requirements.

fastoad_cs25.models.geometry.geom_components.wing.wing_global_positions module

Convenience module for computing leading edge X positions of wing chords.

class fastoad_cs25.models.geometry.geom_components.wing.wing_global_positions.**ChordGlobalPositions**(**kw

Bases: openmdao.core.group.Group

Computes leading edge X positions of wing chords (oot, kink, tip) with respect to aircraft nose.

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

class fastoad_cs25.models.geometry.geom_components.wing.wing_global_positions.**ComputeChordGlobalPositio**

Bases: openmdao.components.add_subtract_comp.AddSubtractComp

Computes leading edge X positions of wing chords with respect to aircraft nose.

Allow user to create an addition/subtracton system with one-liner.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

Module contents

Estimation of wing (global)

Submodules

fastoad_cs25.models.geometry.geom_components.compute_wetted_area module

Estimation of total aircraft wet area

class fastoad_cs25.models.geometry.geom_components.compute_wetted_area.**ComputeWettedArea**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Total aircraft wet area estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

Module contents

Estimation of geometry components

fastoad_cs25.models.geometry.profiles package

Subpackages

Submodules

fastoad_cs25.models.geometry.profiles.profile module

Management of 2D wing profiles

class fastoad_cs25.models.geometry.profiles.profile.**Coordinates2D**(*x*, *y*)
Bases: tuple

Create new instance of Coordinates2D(x, y)

x
Alias for field number 0

y
Alias for field number 1

class fastoad_cs25.models.geometry.profiles.profile.**Profile**(*chord_length: float = 0.0*)
Bases: object

Class for managing 2D wing profiles :param chord_length:

chord_length: float
in meters

property thickness_ratio: float
thickness-to-chord ratio

set_points(*x: Sequence*, *z: Sequence*, *keep_chord_length: bool = True*, *keep_relative_thickness: bool = True*)
Sets points of the 2D profile.

Provided points are expected to be in order around the profile (clockwise or anti-clockwise).

Parameters

- **x** – in meters
- **z** – in meters
- **keep_relative_thickness** –
- **keep_chord_length** –

get_mean_line() → pandas.core.frame.DataFrame
Point set of mean line of the profile.

DataFrame keys are 'x' and 'z', given in meters.

get_relative_thickness() → pandas.core.frame.DataFrame
Point set of relative thickness of the profile.

DataFrame keys are 'x' and 'thickness' and are relative to chord_length. 'x' is from 0. to 1.

get_upper_side() → pandas.core.frame.DataFrame
Point set of upper side of the profile.

DataFrame keys are 'x' and 'z', given in meters.

get_lower_side() → pandas.core.frame.DataFrame
Point set of lower side of the profile.

DataFrame keys are 'x' and 'z', given in meters.

get_sides() → pandas.core.frame.DataFrame
Point set of the whole profile

Points are given from trailing edge to trailing edge, starting by upper side.

fastoad_cs25.models.geometry.profiles.profile_getter module

Airfoil reshape function

fastoad_cs25.models.geometry.profiles.profile_getter.**get_profile**(*file_name: str = 'BACJ.txt'*, *chord_length=1.0*, *thickness_ratio=None*) → *fastoad_cs25.models.geometry.profiles.profile.Profile*

Reads profile from indicated resource file and returns it after resize

Parameters

- **file_name** – name of resource
- **chord_length** – set to None to get original chord length
- **thickness_ratio** –

Returns the Profile instance

Module contents

Management of wing profiles

Submodules

fastoad_cs25.models.geometry.compute_aero_center module

Estimation of aerodynamic center

class fastoad_cs25.models.geometry.compute_aero_center.**ComputeAeroCenter**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Aerodynamic center estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.geometry.constants module

Constants for geometry submodels.

fastoad_cs25.models.geometry.geometry module

FAST - Copyright (c) 2016 ONERA ISAE

class fastoad_cs25.models.geometry.geometry.**Geometry**(***kwargs*)
Bases: openmdao.core.group.Group

Computes geometric characteristics of the (tube-wing) aircraft:

- fuselage size can be computed from payload requirements
- wing dimensions are computed from global parameters (area, taper ratio…)
- tail planes are dimensioned from HQ requirements

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

Module contents

Estimation of global geometry components

fastoad_cs25.models.handling_qualities package

Subpackages

fastoad_cs25.models.handling_qualities.tail_sizing package

Submodules

fastoad_cs25.models.handling_qualities.tail_sizing.compute_ht_area module

Estimation of horizontal tail area

class fastoad_cs25.models.handling_qualities.tail_sizing.compute_ht_area.**ComputeHTArea**(***kwargs*)

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computes area of horizontal tail plane

Area is computed to fulfill aircraft balance requirement at rotation speed

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.handling_qualities.tail_sizing.compute_tail_areas module

Computation of tail areas w.r.t. HQ criteria

class fastoad_cs25.models.handling_qualities.tail_sizing.compute_tail_areas.**ComputeTailAreas**(***kwargs*)
Bases: openmdao.core.group.Group

Computes areas of vertical and horizontal tail.

- Horizontal tail area is computed so it can balance pitching moment of aircraft at rotation speed.
- Vertical tail area is computed so aircraft can have the CNbeta in cruise conditions

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

fastoad_cs25.models.handling_qualities.tail_sizing.compute_vt_area module

Estimation of vertical tail area

class fastoad_cs25.models.handling_qualities.tail_sizing.compute_vt_area.**ComputeVTArea**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computes area of vertical tail plane

Area is computed to fulfill lateral stability requirement (with the most aft CG) as stated in :cite:raymer:1992.

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.

- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

Module contents

Submodules

fastoad_cs25.models.handling_qualities.compute_static_margin module

Estimation of static margin

class fastoad_cs25.models.handling_qualities.compute_static_margin.**ComputeStaticMargin**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computation of static margin i.e. difference between CG ratio and neutral point.

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

Module contents

fastoad_cs25.models.loops package

Subpackages

fastoad_cs25.models.loops.wing_area_component package

Subpackages

Submodules

fastoad_cs25.models.loops.wing_area_component.update_wing_area_aero module

Computation of wing area following aerodynamic constraints

class fastoad_cs25.models.loops.wing_area_component.update_wing_area_aero.**UpdateWingAreaAero**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computes wing area for having enough lift at required approach speed.

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

compute_partials(*inputs*, *partials*, *discrete_inputs=None*)
Compute sub-jacobian parts. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **partials** (*Jacobian*) – Sub-jac components written to partials[output_name, input_name]..
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.

class fastoad_cs25.models.loops.wing_area_component.update_wing_area_aero.**WingAreaConstraintsAero**(***kwa*
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.

- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

compute_partials(*inputs*, *partials*, *discrete_inputs=None*)
Compute sub-jacobian parts. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **partials** (*Jacobian*) – Sub-jac components written to partials[output_name, input_name]..
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.

fastoad_cs25.models.loops.wing_area_component.update_wing_area_geom module

Computation of wing area following geometric constraints

class fastoad_cs25.models.loops.wing_area_component.update_wing_area_geom.**UpdateWingAreaGeom**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computes wing area for being able to load enough fuel to achieve the sizing mission.

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

compute_partials(*inputs*, *partials*, *discrete_inputs=None*)
Compute sub-jacobian parts. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **partials** (*Jacobian*) – Sub-jac components written to partials[output_name, input_name]..
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.

class fastoad_cs25.models.loops.wing_area_component.update_wing_area_geom.**WingAreaConstraintsGeom**(***kwa*
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

Module contents

Submodules

fastoad_cs25.models.loops.compute_wing_area module

Computation of wing area

class fastoad_cs25.models.loops.compute_wing_area.**ComputeWingArea**(***kwargs*)
Bases: openmdao.core.group.Group

Computes needed wing area for:

- having enough lift at required approach speed
- being able to load enough fuel to achieve the sizing mission

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

fastoad_cs25.models.loops.compute_wing_position module

Computation of wing position

class fastoad_cs25.models.loops.compute_wing_position.**ComputeWingPosition**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computes the wing position for a static margin target

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.loops.constants module

Constants for loops submodels

Module contents

fastoad_cs25.models.propulsion package

Subpackages

fastoad_cs25.models.propulsion.fuel_propulsion package

Subpackages

fastoad_cs25.models.propulsion.fuel_propulsion.rubber_engine package

Subpackages

Submodules

fastoad_cs25.models.propulsion.fuel_propulsion.rubber_engine.constants module

Constants for rubber engine analytical models

fastoad_cs25.models.propulsion.fuel_propulsion.rubber_engine.exceptions module

Exceptions for rubber_engine package.

exception fastoad_cs25.models.propulsion.fuel_propulsion.rubber_engine.exceptions.
FastRubberEngineInconsistentInputParametersError

Bases: Exception

Raised when provided parameter combination is incorrect.

fastoad_cs25.models.propulsion.fuel_propulsion.rubber_engine.openmdao module

OpenMDAO wrapping of RubberEngine.

class fastoad_cs25.models.propulsion.fuel_propulsion.rubber_engine.openmdao.
OMRubberEngineWrapper

Bases: fastoad.model_base.propulsion.IOMPropulsionWrapper

Wrapper class of for rubber engine model.

It is made to allow a direct call to RubberEngine in an OpenMDAO component.

Example of usage of this class:

```
import openmdao.api as om

class MyComponent(om.ExplicitComponent):
    def initialize():
        self._engine_wrapper = OMRubberEngineWrapper()

    def setup():
        # Adds OpenMDAO variables that define the engine
        self._engine_wrapper.setup(self)

        # Do the normal setup
        self.add_input("my_input")
        [finish the setup...]

    def compute(self, inputs, outputs, discrete_inputs=None, discrete_outputs=None):
        [do something]

        # Get the engine instance, with parameters defined from OpenMDAO inputs
        engine = self._engine_wrapper.get_model(inputs)

        # Run the engine model. This is a pure Python call. You have to define
        # its inputs before, and to use its outputs according to your needs
        sfc, thrust_rate, thrust = engine.compute_flight_points(
            mach,
            altitude,
            engine_setting,
```

(continues on next page)

(continued from previous page)

```
            use_thrust_rate,
            thrust_rate,
            thrust
            )

        [do something else]

)
```

setup(*component: openmdao.core.component.Component*)
Defines the needed OpenMDAO inputs for propulsion instantiation as done in *get_model()*

Use *add_inputs* and *declare_partials* methods of the provided *component*

Parameters **component** –

static get_model(*inputs*) → fastoad.model_base.propulsion.IPropulsion

Parameters **inputs** – input parameters that define the engine

Returns a RubberEngine instance

class fastoad_cs25.models.propulsion.fuel_propulsion.rubber_engine.openmdao.**OMRubberEngineComponent**(**kw

Bases: fastoad.model_base.propulsion.BaseOMPropulsionComponent

Parametric engine model as OpenMDAO component

See RubberEngine for more information.

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

static get_wrapper() → *fastoad_cs25.models.propulsion.fuel_propulsion.rubber_engine.openmdao.OMRubberEngineWrapper*
This method defines the used IOMPropulsionWrapper instance.

Returns an instance of OpenMDAO wrapper for propulsion model

fastoad_cs25.models.propulsion.fuel_propulsion.rubber_engine.rubber_engine module

Parametric turbofan engine.

class fastoad_cs25.models.propulsion.fuel_propulsion.rubber_engine.rubber_engine.**RubberEngine**(*bypass_ratio*: *float*, *overall_pressure*: *float*, *turbine_inlet_t*: *float*, *mto_thrust*: *float*, *maximum_mach*: *float*, *design_altitud*: *float*, *delta_t4_cli*: *float* = -50, *delta_t4_cru*: *float* = -100, *k_sfc_sl*: *float* = *1.0*, *k_sfc_cr*: *float* = *1.0*)

Bases: fastoad.model_base.propulsion.AbstractFuelPropulsion

Parametric turbofan engine.

It computes engine characteristics using analytical model from following sources:

Parameters

- **bypass_ratio** –
- **overall_pressure_ratio** –
- **turbine_inlet_temperature** – (unit=K) also noted T4
- **mto_thrust** – (unit=N) Maximum TakeOff thrust, i.e. maximum thrust on ground at speed 0, also noted F0
- **maximum_mach** –

- **design_altitude** – (unit=m)
- **delta_t4_climb** – (unit=K) difference between T4 during climb and design T4
- **delta_t4_cruise** – (unit=K) difference between T4 during cruise and design T4
- **k_sfc_sl** – SFC correction at sea level and below
- **k_sfc_cr** – SFC correction at 43000ft and above in cruise

compute_flight_points(*flight_points: Union[fastoad.model_base.flight_point.FlightPoint, pandas.core.frame.DataFrame]*)

Computes Specific Fuel Consumption according to provided conditions.

See `FlightPoint` for available fields that may be used for computation. If a DataFrame instance is provided, it is expected that its columns match field names of FlightPoint (actually, the DataFrame instance should be generated from a list of FlightPoint instances).

Note: About thrust_is_regulated, thrust_rate and thrust

`thrust_is_regulated` tells if a flight point should be computed using `thrust_rate` (when False) or `thrust` (when True) as input. This way, the method can be used in a vectorized mode, where each point can be set to respect a **thrust** order or a **thrust rate** order.

- if `thrust_is_regulated` is not defined, the considered input will be the defined one between `thrust_rate` and `thrust` (if both are provided, `thrust_rate` will be used)
- if `thrust_is_regulated` is `True` or `False` (i.e., not a sequence), the considered input will be taken accordingly, and should of course be defined.
- if there are several flight points, `thrust_is_regulated` is a sequence or array, `thrust_rate` and `thrust` should be provided and have the same shape as `thrust_is_regulated:code:`. The method will consider for each element which input will be used according to `thrust_is_regulated`.

Parameters **flight_points** – FlightPoint or DataFram instance

Returns None (inputs are updated in-place)

compute_flight_points_from_dt4(*mach: Union[float, Sequence]*, *altitude: Union[float, Sequence]*, *delta_t4: Union[float, Sequence]*, *isa_offset: Union[float, Sequence] = 0*, *thrust_is_regulated: Optional[Union[bool, Sequence]] = None*, *thrust_rate: Optional[Union[float, Sequence]] = None*, *thrust: Optional[Union[float, Sequence]] = None*) → Tuple[Union[float, Sequence], Union[float, Sequence], Union[float, Sequence]]

Same as *compute_flight_points()* except that delta_t4 is used directly instead of specifying flight engine_setting.

Parameters

- **mach** – Mach number
- **altitude** – (unit=m) altitude w.r.t. to sea level
- **delta_t4** – (unit=K) difference between operational and design values of turbine inlet temperature in K
- **isa_offset** – (unit=degK) temperature difference from isa conditions
- **thrust_is_regulated** – tells if thrust_rate or thrust should be used (works element-wise)

- **thrust_rate** – thrust rate (unit=none)
- **thrust** – required thrust (unit=N)

Returns SFC (in kg/s/N), thrust rate, thrust (in N)

sfc_at_max_thrust(*atmosphere: stdatm.atmosphere.Atmosphere*, *mach: Union[float, Sequence[float]]*) → numpy.ndarray

Computation of Specific Fuel Consumption at maximum thrust.

Uses model described in [Rou05], p.41.

Parameters

- **atmosphere** – Atmosphere instance at intended altitude
- **mach** – Mach number(s)

Returns SFC (in kg/s/N)

sfc_ratio(*altitude: Union[float, Sequence[float]]*, *thrust_rate: Union[float, Sequence[float]]*, *mach: Union[float, Sequence[float]] = 0.8*) → numpy.ndarray

Computation of ratio $\frac{SFC(F)}{SFC(Fmax)}$, given altitude and thrust_rate $\frac{F}{Fmax}$.

Uses a patched version of model described in [Rou02], p.85.

Warning: this model is very limited

Parameters

- **altitude** –
- **thrust_rate** –
- **mach** – only used for logger checks as model is made for Mach~0.8

Returns SFC ratio

max_thrust(*atmosphere: stdatm.atmosphere.Atmosphere*, *mach: Union[float, Sequence[float]]*, *delta_t4: Union[float, Sequence[float]]*) → numpy.ndarray

Computation of maximum thrust.

Uses model described in [Rou05], p.57-58

Parameters

- **atmosphere** – Atmosphere instance at intended altitude (should be <=20km)
- **mach** – Mach number(s) (should be between 0.05 and 1.0)
- **delta_t4** – (unit=K) difference between operational and design values of turbine inlet temperature in K

Returns maximum thrust (in N)

installed_weight() → float

Computes weight of installed engine, depending on MTO thrust (F0).

Uses model described in [Rou05], p.74

Returns installed weight (in kg)

length() → float

Computes engine length from MTO thrust and maximum Mach.

Model from [Ray99], p.74

Returns engine length (in m)

nacelle_diameter() → float
Computes nacelle diameter from MTO thrust and bypass ratio.

Model of engine diameter from [Ray99], p.235. Nacelle diameter is considered 10% greater ([kro01])

Returns nacelle diameter (in m)

Module contents

Provides a parametric model for turbofan:

- as a pure Python
- as OpenMDAO modules

Module contents

Module contents

Package for propulsion modules

fastoad_cs25.models.weight package

Subpackages

fastoad_cs25.models.weight.cg package

Subpackages

fastoad_cs25.models.weight.cg.cg_components package

Subpackages

fastoad_cs25.models.weight.cg.cg_components.load_cases package

Submodules

fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase1 module

Estimation of center of gravity for load case 1

class fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase1.**ComputeCGLoadCase1**(**
Bases: *fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase_base.ComputeCGLoadCase*

Center of gravity estimation for load case 1

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase2 module

Estimation of center of gravity for load case 2

class fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase2.**ComputeCGLoadCase2**(**
Bases: *fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase_base.ComputeCGLoadCase*

Center of gravity estimation for load case 3

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase3 module

Estimation of center of gravity for load case 3

class fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase3.**ComputeCGLoadCase3**(**
Bases: *fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase_base.ComputeCGLoadCase*

Center of gravity estimation for load case 3

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase4 module

Estimation of center of gravity for load case 4

class fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase4.**ComputeCGLoadCase4**(**
Bases: *fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase_base.ComputeCGLoadCase*

Center of gravity estimation for load case

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase_base module

CG calculation for load cases.

class fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase_base.**ComputeCGLoadCase**
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Base class for computing load cases for CG calculations.

Store some bound methods so we can detect runtime overrides.

property output_name
Builds name of the unique output from option "case_number".

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcases module

Computes and aggregates CG ratios for load cases.

class fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcases.**CGRatiosForLoadCases**(
Bases: openmdao.core.group.Group

Aggregation of CG ratios from load case calculations.

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

class fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcases.**MaxCGRatiosForLoadCas**
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Maximum center of gravity ratio from load cases.

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

Module contents

Submodules

fastoad_cs25.models.weight.cg.cg_components.compute_cg_control_surfaces module

Estimation of control surfaces center of gravity

class fastoad_cs25.models.weight.cg.cg_components.compute_cg_control_surfaces.**ComputeControlSurfacesCG**(
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Control surfaces center of gravity estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.cg.cg_components.compute_cg_others module

Estimation centers of gravity of other components

class fastoad_cs25.models.weight.cg.cg_components.compute_cg_others.**ComputeOthersCG**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Other components center of gravities estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.cg.cg_components.compute_cg_ratio_aft module

Estimation of center of gravity ratio with aft

class fastoad_cs25.models.weight.cg.cg_components.compute_cg_ratio_aft.**ComputeCGRatioAft**(***kwargs*)
Bases: openmdao.core.group.Group

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

class fastoad_cs25.models.weight.cg.cg_components.compute_cg_ratio_aft.**ComputeCG**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

class fastoad_cs25.models.weight.cg.cg_components.compute_cg_ratio_aft.**CGRatio**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.

- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.cg.cg_components.compute_cg_tanks module

Estimation of tanks center of gravity

class fastoad_cs25.models.weight.cg.cg_components.compute_cg_tanks.**ComputeTanksCG**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Tanks center of gravity estimation

Store some bound methods so we can detect runtime overrides.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.cg.cg_components.compute_cg_wing module

Estimation of wing center of gravity

class fastoad_cs25.models.weight.cg.cg_components.compute_cg_wing.**ComputeWingCG**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Wing center of gravity estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.cg.cg_components.compute_global_cg module

Estimation of global center of gravity

class fastoad_cs25.models.weight.cg.cg_components.compute_global_cg.**ComputeGlobalCG**(***kwargs*)
Bases: openmdao.core.group.Group

Global center of gravity estimation

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

fastoad_cs25.models.weight.cg.cg_components.compute_ht_cg module

Estimation of horizontal tail center of gravity

class fastoad_cs25.models.weight.cg.cg_components.compute_ht_cg.**ComputeHTcg**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Horizontal tail center of gravity estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.cg.cg_components.compute_max_cg_ratio module

Estimation of maximum center of gravity ratio

class fastoad_cs25.models.weight.cg.cg_components.compute_max_cg_ratio.**ComputeMaxCGratio**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Maximum center of gravity ratio estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.cg.cg_components.compute_vt_cg module

Estimation of vertical tail center of gravity

class fastoad_cs25.models.weight.cg.cg_components.compute_vt_cg.**ComputeVTcg**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Vertical tail center of gravity estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.cg.cg_components.update_mlg module

Estimation of main landing gear center of gravity

class fastoad_cs25.models.weight.cg.cg_components.update_mlg.**UpdateMLG**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Main landing gear center of gravity estimation

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

Module contents

Estimation of centers of gravity

Submodules

fastoad_cs25.models.weight.cg.cg module

FAST - Copyright (c) 2016 ONERA ISAE

class fastoad_cs25.models.weight.cg.cg.**CG**(***kwargs*)
Bases: fastoad.openmdao.base_model_classes.CycleGroup

Model that computes the global center of gravity

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

class fastoad_cs25.models.weight.cg.cg.**ComputeAircraftCG**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Compute position of aircraft CG from CG ratio

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.cg.constants module

Constants for CG submodels.

Module contents

fastoad_cs25.models.weight.mass_breakdown package

Subpackages

fastoad_cs25.models.weight.mass_breakdown.a_airframe package

Submodules

fastoad_cs25.models.weight.mass_breakdown.a_airframe.a1_wing_weight module

Estimation of wing weight

class fastoad_cs25.models.weight.mass_breakdown.a_airframe.a1_wing_weight.**WingWeight**(***kwargs*)

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Wing weight estimation

This is done by summing following estimations:

- mass from sizing to flexion forces
- mass from sizing to shear forces
- mass of ribs
- mass of reinforcements
- mass of secondary parts

Based on [DCAC14], mass contribution A1

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.

- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.a_airframe.a2_fuselage_weight module

Estimation of fuselage weight

class fastoad_cs25.models.weight.mass_breakdown.a_airframe.a2_fuselage_weight.**FuselageWeight**(***kwargs*)

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Fuselage weight estimation

Based on a statistical analysis. See [DCAC14], mass contribution A2

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.a_airframe.a3_empennage_weight module

Estimation of empennage weight

class fastoad_cs25.models.weight.mass_breakdown.a_airframe.a3_empennage_weight.**EmpennageWeight**(***kwargs*)

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for tail planes

Based on formulas in [DCAC14], mass contribution A3

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.a_airframe.a4_flight_control_weight module

Estimation of flight controls weight

class fastoad_cs25.models.weight.mass_breakdown.a_airframe.a4_flight_control_weight.**FlightControlsWeigh**
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Flight controls weight estimation

Based on formulas in [DCAC14], mass contribution A4

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.a_airframe.a5_landing_gear_weight module

Estimation of landing gear weight

class fastoad_cs25.models.weight.mass_breakdown.a_airframe.a5_landing_gear_weight.**LandingGearWeight**(**kw

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for landing gears

Based on formulas in [DCAC14], mass contribution A5

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.a_airframe.a6_pylons_weight module

Estimation of pylons weight

class fastoad_cs25.models.weight.mass_breakdown.a_airframe.a6_pylons_weight.**PylonsWeight**(**kwargs*)

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for pylons

Based on formula in [DCAC14], mass contribution A6

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.a_airframe.a7_paint_weight module

Estimation of paint weight

class fastoad_cs25.models.weight.mass_breakdown.a_airframe.a7_paint_weight.**PaintWeight**(***kwargs*)

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for paint

Based on formula in [DCAC14], mass contribution A7

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.a_airframe.constants module

Constants for airframe mass submodels.

fastoad_cs25.models.weight.mass_breakdown.a_airframe.sum module

Computation of airframe mass.

class fastoad_cs25.models.weight.mass_breakdown.a_airframe.sum.**AirframeWeight**(***kwargs*)
Bases: openmdao.core.group.Group

Computes mass of airframe.

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

Module contents

Estimation of airframe weight

fastoad_cs25.models.weight.mass_breakdown.b_propulsion package

Submodules

fastoad_cs25.models.weight.mass_breakdown.b_propulsion.b1_engine_weight module

Estimation of engine weight

class fastoad_cs25.models.weight.mass_breakdown.b_propulsion.b1_engine_weight.**EngineWeight**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Engine weight estimation

Uses model described in [Rou05], p.74

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.b_propulsion.b2_fuel_lines_weight module

Estimation of fuel lines weight

class fastoad_cs25.models.weight.mass_breakdown.b_propulsion.b2_fuel_lines_weight.**FuelLinesWeight**(***kwa*

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for fuel lines

Based on formula in [DCAC14], mass contribution B2

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.b_propulsion.b3_unconsumables_weight module

Estimation of fuel lines weight

class fastoad_cs25.models.weight.mass_breakdown.b_propulsion.b3_unconsumables_weight.**UnconsumablesWeigh**

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for oil and unusable fuel

Based on formula in [DCAC14], mass contribution B3

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.b_propulsion.constants module

Constants for propulsion mass submodels.

fastoad_cs25.models.weight.mass_breakdown.b_propulsion.sum module

Computation of propulsion mass.

class fastoad_cs25.models.weight.mass_breakdown.b_propulsion.sum.**PropulsionWeight**(***kwargs*)
Bases: openmdao.core.group.Group

Computes mass of propulsion.

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

Module contents

Estimation of propulsion weight

fastoad_cs25.models.weight.mass_breakdown.c_systems package

Submodules

fastoad_cs25.models.weight.mass_breakdown.c_systems.c1_power_systems_weight module

Estimation of power systems weight

class fastoad_cs25.models.weight.mass_breakdown.c_systems.c1_power_systems_weight.**PowerSystemsWeight**(**

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for power systems (generation and distribution)

This includes:

- Auxiliary Power Unit (APU)
- electric systems
- hydraulic systems

Based on formulas in [DCAC14], mass contribution C1

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.c_systems.c2_life_support_systems_weight module

Estimation of life support systems weight

class fastoad_cs25.models.weight.mass_breakdown.c_systems.c2_life_support_systems_weight.LifeSupportSys
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for life support systems

This includes:

- insulation
- air conditioning / pressurization
- de-icing
- internal lighting system
- seats and installation of crew
- fixed oxygen
- permanent security kits

Based on formulas in [DCAC14], mass contribution C2

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.c_systems.c3_navigation_systems_weight module

Estimation of navigation systems weight

class fastoad_cs25.models.weight.mass_breakdown.c_systems.c3_navigation_systems_weight.**NavigationSystems**
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for navigation systems

Based on figures in [DCAC14], mass contribution C3

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.c_systems.c4_transmissions_systems_weight module

Estimation of transmissions systems weight

class fastoad_cs25.models.weight.mass_breakdown.c_systems.c4_transmissions_systems_weight.**TransmissionSy**
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for transmission systems

Based on figures in [DCAC14], mass contribution C4

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.c_systems.c5_fixed_operational_systems_weight module

Estimation of fixed operational systems weight

class fastoad_cs25.models.weight.mass_breakdown.c_systems.c5_fixed_operational_systems_weight.**FixedOpera**
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for fixed operational systems (weather radar, flight recorder, ...)

Based on formulas in [DCAC14], mass contribution C5

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.c_systems.c6_flight_kit_weight module

Estimation of flight kit weight

class fastoad_cs25.models.weight.mass_breakdown.c_systems.c6_flight_kit_weight.**FlightKitWeight**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for flight kit (tools that are always on board)

Based on figures in [DCAC14], mass contribution C6

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.c_systems.constants module

Constants for systems mass submodels.

fastoad_cs25.models.weight.mass_breakdown.c_systems.sum module

Computation of mass of systems.

class fastoad_cs25.models.weight.mass_breakdown.c_systems.sum.**SystemsWeight**(***kwargs*)
Bases: openmdao.core.group.Group

Computes mass of systems.

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

Module contents

Estimation of weight of all-mission systems

fastoad_cs25.models.weight.mass_breakdown.d_furniture package

Submodules

fastoad_cs25.models.weight.mass_breakdown.d_furniture.constants module

Constants for systems mass submodels.

fastoad_cs25.models.weight.mass_breakdown.d_furniture.d1_cargo_configuration_weight module

Estimation of cargo configuration weight

class fastoad_cs25.models.weight.mass_breakdown.d_furniture.d1_cargo_configuration_weight.CargoConfigura

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for equipments for freight transport (applies only for freighters)

Based on [DCAC14], mass contribution D1

Store some bound methods so we can detect runtime overrides.

setup()

Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()

Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)

Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.d_furniture.d2_passenger_seats_weight module

Estimation of passenger seats weight

class fastoad_cs25.models.weight.mass_breakdown.d_furniture.d2_passenger_seats_weight.**PassengerSeatsWei**
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for passenger seats

Based on [DCAC14], mass contribution D2

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.d_furniture.d3_food_water_weight module

Estimation of food water weight

class fastoad_cs25.models.weight.mass_breakdown.d_furniture.d3_food_water_weight.**FoodWaterWeight**(**kwarg
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for food and water

Includes trolleys, trays, cutlery…

Based on [DCAC14], mass contribution D3

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.d_furniture.d4_security_kit_weight module

Estimation of security kit weight

class fastoad_cs25.models.weight.mass_breakdown.d_furniture.d4_security_kit_weight.**SecurityKitWeight**(**

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for security kit

Based on [DCAC14], mass contribution D4

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.d_furniture.d5_toilets_weight module

Estimation of toilets weight

class fastoad_cs25.models.weight.mass_breakdown.d_furniture.d5_toilets_weight.**ToiletsWeight**(**kwargs*)

Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for toilets

Based on [DCAC14], mass contribution D5

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.d_furniture.sum module

Computation of furniture mass.

class fastoad_cs25.models.weight.mass_breakdown.d_furniture.sum.**FurnitureWeight**(***kwargs*)
Bases: openmdao.core.group.Group

Computes mass of furniture.

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

Module contents

Estimation of furniture weight

fastoad_cs25.models.weight.mass_breakdown.e_crew package

Submodules

fastoad_cs25.models.weight.mass_breakdown.e_crew.crew_weight module

Estimation of crew weight

class fastoad_cs25.models.weight.mass_breakdown.e_crew.crew_weight.**CrewWeight**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Weight estimation for aircraft crew

Based on [DCAC14], mass contribution E

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

Module contents

Estimation of crew weight

Submodules

fastoad_cs25.models.weight.mass_breakdown.constants module

Constants for mass submodels.

fastoad_cs25.models.weight.mass_breakdown.cs25 module

Computation of load cases

class fastoad_cs25.models.weight.mass_breakdown.cs25.**Loads**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computes gust load cases

Load case 1: with wings with almost no fuel Load case 2: at maximum take-off weight

Based on formulas in [DCAC14], §6.3

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.mass_breakdown module

Main components for mass breakdown.

class fastoad_cs25.models.weight.mass_breakdown.mass_breakdown.**MassBreakdown**(***kwargs*)
Bases: fastoad.openmdao.base_model_classes.CycleGroup

Computes analytically the mass of each part of the aircraft, and the resulting sum, the Overall Weight Empty (OWE).

Some models depend on MZFW (Max Zero Fuel Weight), MLW (Max Landing Weight) and MTOW (Max TakeOff Weight), which depend on OWE.

This model cycles for having consistent OWE, MZFW and MLW.

Options: - payload_from_npax: If True (default), payload masses will be computed from NPAX, if False

design payload mass and maximum payload mass must be provided.

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

class fastoad_cs25.models.weight.mass_breakdown.mass_breakdown.**OperatingWeightEmpty**(***kwargs*)
Bases: openmdao.core.group.Group

Operating Empty Weight (OEW) estimation.

This group aggregates weight from all components of the aircraft.

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

fastoad_cs25.models.weight.mass_breakdown.payload module

Payload mass computation

class fastoad_cs25.models.weight.mass_breakdown.payload.**ComputePayload**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computes payload from NPAX

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

fastoad_cs25.models.weight.mass_breakdown.update_mlw_and_mzfw module

Main component for mass breakdown

class fastoad_cs25.models.weight.mass_breakdown.update_mlw_and_mzfw.**UpdateMLWandMZFW**(***kwargs*)
Bases: openmdao.core.explicitcomponent.ExplicitComponent

Computes Maximum Landing Weight and Maximum Zero Fuel Weight from Overall Empty Weight and Maximum Payload.

Store some bound methods so we can detect runtime overrides.

setup()
Declare inputs and outputs.

Available attributes: name pathname comm options

setup_partials()
Declare partials.

This is meant to be overridden by component classes. All partials should be declared here since this is called after all size/shape information is known for all variables.

compute(*inputs*, *outputs*, *discrete_inputs=None*, *discrete_outputs=None*)
Compute outputs given inputs. The model is assumed to be in an unscaled state.

Parameters

- **inputs** (*Vector*) – Unscaled, dimensional input variables read via inputs[key].
- **outputs** (*Vector*) – Unscaled, dimensional output variables read via outputs[key].
- **discrete_inputs** (*dict or None*) – If not None, dict containing discrete input values.
- **discrete_outputs** (*dict or None*) – If not None, dict containing discrete output values.

Module contents

Estimation of Aircraft Weight

Submodules

fastoad_cs25.models.weight.constants module

Constants for weight submodels.

fastoad_cs25.models.weight.weight module

Weight computation (mass and CG)

class fastoad_cs25.models.weight.weight.**Weight**(***kwargs*)
Bases: openmdao.core.group.Group

Computes masses and Centers of Gravity for each part of the empty operating aircraft, among these 5 categories: airframe, propulsion, systems, furniture, crew

This model uses MTOW as an input, as it allows to size some elements, but resulting OWE do not aim at being consistent with MTOW.

Consistency between OWE and MTOW can be achieved by cycling with a model that computes MTOW from OWE, which should come from a mission computation that will assess needed block fuel.

Set the solvers to nonlinear and linear block Gauss–Seidel by default.

initialize()
Perform any one-time initialization run at instantiation.

setup()
Build this group.

This method should be overidden by your Group's method. The reason for using this method to add subsystem is to save memory and setup time when using your Group while running under MPI. This avoids the creation of systems that will not be used in the current process.

You may call 'add_subsystem' to add systems to this group. You may also issue connections, and set the linear and nonlinear solvers for this group level. You cannot safely change anything on children systems; use the 'configure' method instead.

Available attributes: name pathname comm options

Module contents

Submodules

fastoad_cs25.models.constants module

Module for management of options and factorizing their definition.

Module contents

This package contains the OAD models of FAST-OAD.

It has to be declared as FAST-OAD plugin.

These models are based on following references:

Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[kro01] 2001. URL: https://web.archive.org/web/20010307121417/http://adg.stanford.edu/aa241/propulsion/nacelledesign.html.

[Ray99] Daniel P. Raymer. *Aircraft Design: A Conceptual Approach, Third edition*. AIAA (American Institute of Aeronautics & Astronautics, 1999. ISBN 1563473437.

[Rou02] Elodie Roux. *Modèles Moteurs... Réacteurs double flux civils et réacteurs militaires à faible taux de dilution avec Post-Combustion*. INSA-SupAéro-ONÉRA, 2002. URL: http://elodieroux.com/ReportFiles/ModelesMoteurVersionPublique.pdf.

[Rou05] Elodie Roux. *Pour une approche analytique de la Dynamique du Vol*. PhD thesis, SupAéro, 2005. URL: http://depozit.isae.fr/theses/2005/2005_Roux_Elodie.pdf.

[kro01] 2001. URL: https://web.archive.org/web/20010307121417/http://adg.stanford.edu/aa241/propulsion/nacelledesign.html.

[DCAC14] Willy Pierre Dupont, Christian Colongo, Olivier Atinault, and Christophe Cros. *Preliminary Design of a Commercial Transport Aircraft*. ISAE-SUPAERO, 2014.

[Ray99] Daniel P. Raymer. *Aircraft Design: A Conceptual Approach, Third edition*. AIAA (American Institute of Aeronautics & Astronautics, 1999. ISBN 1563473437.

[Rou02] Elodie Roux. *Modèles Moteurs... Réacteurs double flux civils et réacteurs militaires à faible taux de dilution avec Post-Combustion*. INSA-SupAéro-ONÉRA, 2002. URL: http://elodieroux.com/ReportFiles/ModelesMoteurVersionPublique.pdf.

[Rou05] Elodie Roux. *Pour une approche analytique de la Dynamique du Vol*. PhD thesis, SupAéro, 2005. URL: http://depozit.isae.fr/theses/2005/2005_Roux_Elodie.pdf.

PYTHON MODULE INDEX

f

INDEX

D

E

F

G

M

N

O

P

R

S

T

U

W

X

Y