

---

# FAST-OAD\_CS25

unknown

May 17, 2024



# CONTENTS

<b>1</b>	<b>Contents</b>	<b>3</b>
<b>2</b>	<b>Indices and tables</b>	<b>93</b>
	<b>Bibliography</b>	<b>95</b>
	<b>Python Module Index</b>	<b>97</b>
	<b>Index</b>	<b>101</b>



WIP



## CONTENTS

## 1.1 License

## 1.2 Contributors

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



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

**Returns** Drag coefficient w.r.t. a surface of area  $\text{length} \times 1 \text{ m}^2$

## 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 overridden 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

Initialize 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 overridden 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 through analytical models.

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

**setup()**

Build this group.



This method should be overridden 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 overridden 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 overridden 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 overridden 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.ComputeCnBetaFuselage`  
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**(\*\*kwargs)

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(**kwargs)
```

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 overridden 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_thrust:
                                                    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.**ComputeNacellePylons**

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**(\*\*kwargs)

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**(\*\*kwargs)

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 overridden 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**(\*\*kwargs)  
 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**(\*\*kwargs)  
 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**(\*\*kwargs)

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 overridden 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(**kwargs)
```

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 overridden 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(\*\*kwargs)

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 overridden 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.ComputeChordGlobalPosition

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/subtraction 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 overridden 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 overridden 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(**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

**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**(*\*\*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

**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 overridden 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(\*\*kwargs)

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*-

*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,
over-
all_pressure_ratio:
float,
tur-
bine_inlet_temperature:
float,
mto_thrust:
float,
max-
i-
mum_mach:
float,
de-
sign_altitude:
float,
delta_t4_cruise:
float
=
-

50,
delta_t4_cruise:
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  $\frac{SFC(F)}{SFC(F_{max})}$ , given altitude and thrust\_rate  $\frac{F}{F_{max}}$ .

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 overridden 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.MaxCGRatiosForLoadCases  
 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 overridden by your Group’s method. The reason for using this method to add sub-system 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 overridden 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 overridden 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.FlightControlsWeight

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(**kwargs)
```

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 overridden 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(\*\*kwargs)  
 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.UnconsumablesWeight  
 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 overridden 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.NavigationSystem

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.TransmissionS

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.**FixedOper**

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 overridden by your Group’s method. The reason for using this method to add sub-system 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.CargoConfigurationWeight`

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.PassengerSeatsWeight

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(\*\*kwargs)

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 overridden 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 overridden 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 overridden 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 overridden 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**





## 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](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](http://depozit.isae.fr/theses/2005/2005_Roux_Elodie.pdf).



## PYTHON MODULE INDEX

f

fastoad_cs25, 91	16	fastoad_cs25.models.aerodynamics.components.high_lift_aero
fastoad_cs25.configurations, 6	17	fastoad_cs25.models.aerodynamics.components.initialize_cl
fastoad_cs25.models, 91	18	fastoad_cs25.models.aerodynamics.components.oswald,
fastoad_cs25.models.aerodynamics, 24	18	fastoad_cs25.models.aerodynamics.components.utils,
fastoad_cs25.models.aerodynamics.aerodynamics_high_speed,	8	fastoad_cs25.models.aerodynamics.components.utils.cd0_lift
20	6	fastoad_cs25.models.aerodynamics.components.utils.friction
fastoad_cs25.models.aerodynamics.aerodynamics_landing,	7	fastoad_cs25.models.aerodynamics.constants,
21	23	fastoad_cs25.models.aerodynamics.external, 20
fastoad_cs25.models.aerodynamics.aerodynamics_low_speed,	20	fastoad_cs25.models.aerodynamics.external.xfoil,
22	8	fastoad_cs25.models.aerodynamics.external.xfoil.xfoil699,
fastoad_cs25.models.aerodynamics.aerodynamics_takeoff,	20	fastoad_cs25.models.aerodynamics.external.xfoil.xfoil_pola
23	20	fastoad_cs25.models.constants, 90
fastoad_cs25.models.aerodynamics.components,	46	fastoad_cs25.models.geometry, 46
19	44	fastoad_cs25.models.geometry.compute_aero_center,
fastoad_cs25.models.aerodynamics.components.cd0,	45	fastoad_cs25.models.geometry.constants, 45
8	43	fastoad_cs25.models.geometry.geom_components,
fastoad_cs25.models.aerodynamics.components.cd0_fuselage,	42	fastoad_cs25.models.geometry.geom_components.compute_wette
8	26	fastoad_cs25.models.geometry.geom_components.fuselage,
fastoad_cs25.models.aerodynamics.components.cd0_ht,	24	fastoad_cs25.models.geometry.geom_components.fuselage.comp
9	25	fastoad_cs25.models.geometry.geom_components.fuselage.comp
fastoad_cs25.models.aerodynamics.components.cd0_nacelles_pylons,	28	fastoad_cs25.models.geometry.geom_components.ht,
10	29	fastoad_cs25.models.geometry.geom_components.ht.components
fastoad_cs25.models.aerodynamics.components.cd0_total,	28	fastoad_cs25.models.geometry.geom_components.ht.components
10	28	fastoad_cs25.models.geometry.geom_components.ht.components
fastoad_cs25.models.aerodynamics.components.cd0_vt,	28	fastoad_cs25.models.geometry.geom_components.ht.components
11	28	fastoad_cs25.models.geometry.geom_components.ht.components
fastoad_cs25.models.aerodynamics.components.cd0_wing,	28	fastoad_cs25.models.geometry.geom_components.ht.components
12	28	fastoad_cs25.models.geometry.geom_components.ht.components
fastoad_cs25.models.aerodynamics.components.cd_compressibility,	28	fastoad_cs25.models.geometry.geom_components.ht.components
12	28	fastoad_cs25.models.geometry.geom_components.ht.components
fastoad_cs25.models.aerodynamics.components.cd_trim,	28	fastoad_cs25.models.geometry.geom_components.ht.components
13	28	fastoad_cs25.models.geometry.geom_components.ht.components
fastoad_cs25.models.aerodynamics.components.compute_alpha,	28	fastoad_cs25.models.geometry.geom_components.ht.components
14	28	fastoad_cs25.models.geometry.geom_components.ht.components
fastoad_cs25.models.aerodynamics.components.compute_ci_alpha,	28	fastoad_cs25.models.geometry.geom_components.ht.components
14	28	fastoad_cs25.models.geometry.geom_components.ht.components
fastoad_cs25.models.aerodynamics.components.compute_max_cl_landing,	28	fastoad_cs25.models.geometry.geom_components.ht.components
15	28	fastoad_cs25.models.geometry.geom_components.ht.components
fastoad_cs25.models.aerodynamics.components.compute_polar,	28	fastoad_cs25.models.geometry.geom_components.ht.components
16	28	fastoad_cs25.models.geometry.geom_components.ht.components
fastoad_cs25.models.aerodynamics.components.compute_reynolds,	28	fastoad_cs25.models.geometry.geom_components.ht.components



---

```

fastoad_cs25.models.weight.cg.cg_components.compute_cg73control_surfaces,
61                                fastoad_cs25.models.weight.mass_breakdown.a_airframe.sum,
fastoad_cs25.models.weight.cg.cg_components.compute_cg74others,
62                                fastoad_cs25.models.weight.mass_breakdown.b_propulsion,
fastoad_cs25.models.weight.cg.cg_components.compute_cg77ratio_aft,
62                                fastoad_cs25.models.weight.mass_breakdown.b_propulsion.b1,
fastoad_cs25.models.weight.cg.cg_components.compute_cg74tanks,
64                                fastoad_cs25.models.weight.mass_breakdown.b_propulsion.b2,
fastoad_cs25.models.weight.cg.cg_components.compute_cg75wing,
64                                fastoad_cs25.models.weight.mass_breakdown.b_propulsion.b3,
fastoad_cs25.models.weight.cg.cg_components.compute_global_cg,
65                                fastoad_cs25.models.weight.mass_breakdown.b_propulsion.cor,
fastoad_cs25.models.weight.cg.cg_components.compute_hf76cg,
65                                fastoad_cs25.models.weight.mass_breakdown.b_propulsion.sum,
fastoad_cs25.models.weight.cg.cg_components.compute_max_cg_ratio,
66                                fastoad_cs25.models.weight.mass_breakdown.c_systems,
fastoad_cs25.models.weight.cg.cg_components.compute_v78cg,
66                                fastoad_cs25.models.weight.mass_breakdown.c_systems.c1_pow,
fastoad_cs25.models.weight.cg.cg_components.load_cases7
61                                fastoad_cs25.models.weight.mass_breakdown.c_systems.c2_li,
fastoad_cs25.models.weight.cg.cg_components.load_cases78compute_cg_loadcase1,
58                                fastoad_cs25.models.weight.mass_breakdown.c_systems.c3_nav,
fastoad_cs25.models.weight.cg.cg_components.load_cases79compute_cg_loadcase2,
59                                fastoad_cs25.models.weight.mass_breakdown.c_systems.c4_tra,
fastoad_cs25.models.weight.cg.cg_components.load_cases79compute_cg_loadcase3,
59                                fastoad_cs25.models.weight.mass_breakdown.c_systems.c5_fix,
fastoad_cs25.models.weight.cg.cg_components.load_cases80compute_cg_loadcase4,
59                                fastoad_cs25.models.weight.mass_breakdown.c_systems.c6_fli,
fastoad_cs25.models.weight.cg.cg_components.load_cases81compute_cg_loadcase_base,
60                                fastoad_cs25.models.weight.mass_breakdown.c_systems.consta,
fastoad_cs25.models.weight.cg.cg_components.load_cases81compute_cg_loadcases,
60                                fastoad_cs25.models.weight.mass_breakdown.c_systems.sum,
fastoad_cs25.models.weight.cg.cg_components.update_ml81
67                                fastoad_cs25.models.weight.mass_breakdown.constants,
fastoad_cs25.models.weight.cg.constants, 69                                87
fastoad_cs25.models.weight.constants, 90                                fastoad_cs25.models.weight.mass_breakdown.cs25,
fastoad_cs25.models.weight.mass_breakdown, 90                                87
fastoad_cs25.models.weight.mass_breakdown.a_airframe, 74                                fastoad_cs25.models.weight.mass_breakdown.d_furniture,
69                                86
fastoad_cs25.models.weight.mass_breakdown.a_airframe, 70                                fastoad_cs25.models.weight.mass_breakdown.d_furniture.cons,
69                                82
fastoad_cs25.models.weight.mass_breakdown.a_airframe, 70                                fastoad_cs25.models.weight.mass_breakdown.d_furniture.d1_c,
70                                82
fastoad_cs25.models.weight.mass_breakdown.a_airframe, 70                                fastoad_cs25.models.weight.mass_breakdown.d_furniture.d2_p,
70                                83
fastoad_cs25.models.weight.mass_breakdown.a_airframe, 71                                fastoad_cs25.models.weight.mass_breakdown.d_furniture.d3_f,
71                                83
fastoad_cs25.models.weight.mass_breakdown.a_airframe, 72                                fastoad_cs25.models.weight.mass_breakdown.d_furniture.d4_s,
72                                84
fastoad_cs25.models.weight.mass_breakdown.a_airframe, 72                                fastoad_cs25.models.weight.mass_breakdown.d_furniture.d5_t,
72                                84
fastoad_cs25.models.weight.mass_breakdown.a_airframe, 73                                fastoad_cs25.models.weight.mass_breakdown.d_furniture.sum,
73                                85
fastoad_cs25.models.weight.mass_breakdown.a_airframe, 73                                fastoad_cs25.models.weight.mass_breakdown.e_crew,
73                                85

```

---

[86](#)  
fastoad\_cs25.models.weight.mass\_breakdown.e\_crew.crew\_weight,  
[86](#)  
fastoad\_cs25.models.weight.mass\_breakdown.mass\_breakdown,  
[87](#)  
fastoad\_cs25.models.weight.mass\_breakdown.payload,  
[88](#)  
fastoad\_cs25.models.weight.mass\_breakdown.update\_mlw\_and\_mzfw,  
[89](#)  
fastoad\_cs25.models.weight.weight, [90](#)



## INDEX

### A

**AerodynamicsHighSpeed** (class in fas-toad\_cs25.models.aerodynamics.aerodynamics\_high\_speed), 20

**AerodynamicsLanding** (class in fas-toad\_cs25.models.aerodynamics.aerodynamics\_landing), 21

**AerodynamicsLowSpeed** (class in fas-toad\_cs25.models.aerodynamics.aerodynamics\_low\_speed), 22

**AerodynamicsTakeoff** (class in fas-toad\_cs25.models.aerodynamics.aerodynamics\_takeoff), 23

**AirframeWeight** (class in fas-toad\_cs25.models.weight.mass\_breakdown.a\_airframe.sum), 74

**CdCompressibility** (class in fas-toad\_cs25.models.aerodynamics.components.cd\_compressibility), 12

**CdTrim** (class in fas-toad\_cs25.models.aerodynamics.components.cd\_trim), 13

**CG** (class in fastoad\_cs25.models.weight.cg.cg), 68

**CGRatio** (class in fas-toad\_cs25.models.weight.cg.cg\_components.compute\_cg\_ratio\_a), 63

**CGRatiosForLoadCases** (class in fas-toad\_cs25.models.weight.cg.cg\_components.load\_cases.compute), 60

**Chord** (class in fastoad\_cs25.models.geometry.geom\_components.nacelle\_profile), 29

**chord\_length** (fastoad\_cs25.models.geometry.profiles.profile.Profile attribute), 43

### C

**cambered** (fastoad\_cs25.models.aerodynamics.components.units.cao\_lifting\_surface.LiftingSurfaceGeometry attribute), 7

**CargoConfigurationWeight** (class in fas-toad\_cs25.models.weight.mass\_breakdown.d\_furniture.d1\_cargo\_configuration\_weight), 82

**CD0** (class in fastoad\_cs25.models.aerodynamics.components.cd0), 8

**Cd0Fuselage** (class in fas-toad\_cs25.models.aerodynamics.components.cd0\_fuselage), 8

**Cd0HorizontalTail** (class in fas-toad\_cs25.models.aerodynamics.components.cd0\_ht), 9

**Cd0NacellesAndPylons** (class in fas-toad\_cs25.models.aerodynamics.components.cd0\_nacelles\_pylons), 10

**Cd0Total** (class in fas-toad\_cs25.models.aerodynamics.components.cd0\_total), 10

**Cd0VerticalTail** (class in fas-toad\_cs25.models.aerodynamics.components.cd0\_vt), 11

**Cd0Wing** (class in fas-toad\_cs25.models.aerodynamics.components.cd0\_wing), 11

**ChordGlobalPositions** (class in fas-toad\_cs25.models.geometry.geom\_components.wing\_global\_positions), 41

**compute()** (fastoad\_cs25.models.aerodynamics.aerodynamics\_landing.Co method), 22

**compute()** (fastoad\_cs25.models.aerodynamics.aerodynamics\_landing.Co method), 22

**compute()** (fastoad\_cs25.models.aerodynamics.components.cd0\_fuselage.Co method), 9

**compute()** (fastoad\_cs25.models.aerodynamics.components.cd0\_ht.Co method), 9

**compute()** (fastoad\_cs25.models.aerodynamics.components.cd0\_nacelles\_pylons.Co method), 10

**compute()** (fastoad\_cs25.models.aerodynamics.components.cd0\_total.Co method), 10

**compute()** (fastoad\_cs25.models.aerodynamics.components.cd0\_vt.Co method), 11

**compute()** (fastoad\_cs25.models.aerodynamics.components.cd0\_wing.Co method), 12

**compute()** (fastoad\_cs25.models.aerodynamics.components.cd\_compressibility.Co method), 13

**compute()** (fastoad\_cs25.models.aerodynamics.components.cd\_trim.Co method), 13

**compute()** (fastoad\_cs25.models.aerodynamics.components.compute\_alpha), 13

[illegible]



41	36
ComputeCLAlpha (class in fas-toad_cs25.models.aerodynamics.components.compute_cl_alpha),	ComputeMaxCGRatio (class in fas-toad_cs25.models.weight.cg.cg_components.compute_max_cg_ratio),
14	66
ComputeCnBetaFuselage (class in fas-toad_cs25.models.geometry.geom_components.fuselage.compute_cn_beta_fuselage),	ComputeMaxCLLanding (class in fas-toad_cs25.models.aerodynamics.components.compute_max_cl_landing),
24	15
ComputeControlSurfacesCG (class in fas-toad_cs25.models.weight.cg.cg_components.compute_cg_control_surfaces),	ComputeMFW (class in fas-toad_cs25.models.geometry.geom_components.wing.components.compute_mfw),
61	36
ComputeDeltaHighLift (class in fas-toad_cs25.models.aerodynamics.components.high_lift_aero),	ComputeNacelleAndPylonsGeometry (class in fas-toad_cs25.models.geometry.geom_components.nacelle_pylons.compute_nacelle_and_pylons_geometry),
17	30
ComputeFuselageGeometryBasic (class in fas-toad_cs25.models.geometry.geom_components.fuselage.compute_fuselage_geometry_basic),	ComputeOthersCG (class in fas-toad_cs25.models.weight.cg.cg_components.compute_cg_others),
25	62
ComputeFuselageGeometryCabinSizing (class in fas-toad_cs25.models.geometry.geom_components.fuselage.compute_fuselage_geometry_cabin_sizing),	ComputePayload (class in fas-toad_cs25.models.weight.mass_breakdown.payload),
25	88
ComputeGlobalCG (class in fas-toad_cs25.models.weight.cg.cg_components.compute_global_cg),	ComputePolar (class in fas-toad_cs25.models.aerodynamics.components.compute_polar),
65	16
ComputeHorizontalTailGeometry (class in fas-toad_cs25.models.geometry.geom_components.ht.compute_horizontal_tail_geometry),	ComputeReynolds (class in fas-toad_cs25.models.aerodynamics.components.compute_reynolds),
28	16
ComputeHTArea (class in fas-toad_cs25.models.handling_qualities.tail_sizing.compute_ht_area),	ComputeStaticMargin (class in fas-toad_cs25.models.handling_qualities.compute_static_margin),
46	48
ComputeHTcg (class in fas-toad_cs25.models.weight.cg.cg_components.compute_ht_cg),	ComputeSweepWing (class in fas-toad_cs25.models.geometry.geom_components.wing.components.compute_sweep_wing),
65	37
ComputeHTChord (class in fas-toad_cs25.models.geometry.geom_components.ht.components.compute_ht_chord),	ComputeTailAreas (class in fas-toad_cs25.models.handling_qualities.tail_sizing.compute_tail_areas),
26	47
ComputeHTClalpha (class in fas-toad_cs25.models.geometry.geom_components.ht.components.compute_ht_cl_alpha),	ComputeTanksCG (class in fas-toad_cs25.models.weight.cg.cg_components.compute_cg_tanks),
27	64
ComputeHTMAC (class in fas-toad_cs25.models.geometry.geom_components.ht.components.compute_ht_mac),	ComputeToCWing (class in fas-toad_cs25.models.geometry.geom_components.wing.components.compute_to_cwing),
27	38
ComputeHTSweep (class in fas-toad_cs25.models.geometry.geom_components.ht.components.compute_ht_sweep),	ComputeVerticalTailGeometry (class in fas-toad_cs25.models.geometry.geom_components.vt.compute_vertical_tail_geometry),
28	33
ComputeL1AndL4Wing (class in fas-toad_cs25.models.geometry.geom_components.wing.components.compute_l1_and_l4_wing),	ComputeVTArea (class in fas-toad_cs25.models.handling_qualities.tail_sizing.compute_vt_area),
35	47
ComputeL2AndL3Wing (class in fas-toad_cs25.models.geometry.geom_components.wing.components.compute_l2_and_l3_wing),	ComputeVTCg (class in fas-toad_cs25.models.weight.cg.cg_components.compute_vt_cg),
35	66
ComputeMachReynolds (class in fas-toad_cs25.models.aerodynamics.aerodynamics_landing),	ComputeVTChords (class in fas-toad_cs25.models.geometry.geom_components.vt.components.compute_vt_chords),
21	30
ComputeMACWing (class in fas-toad_cs25.models.geometry.geom_components.wing.components.compute_mac_wing),	ComputeVTClalpha (class in fas-toad_cs25.models.geometry.geom_components.vt.components.compute_vt_cl_alpha),



31	74
ComputeVTDistance (class in fas-	FastoadCs25Components.compute_vt_distance),
toad_cs25.models.geometry.geom_components.vt.components.compute_vt_distance),	
32	fastoad_cs25
ComputeVTMAC (class in fas-	module, 91
toad_cs25.models.geometry.geom_components.vt.components.compute_vt_mac),	
32	fastoad_cs25.models.aerodynamics
ComputeVTSweep (class in fas-	fastoad_cs25.models
toad_cs25.models.geometry.geom_components.vt.components.compute_vt_sweep),	module, 91
33	fastoad_cs25.models.aerodynamics
ComputeWetAreaWing (class in fas-	module, 24
toad_cs25.models.geometry.geom_components.wing.components.compute_wet_area_wing),	
39	fastoad_cs25.models.aerodynamics.aerodynamics_high_speed
ComputeWettedArea (class in fas-	module, 20
toad_cs25.models.geometry.geom_components.compute_wetted_area),	
42	fastoad_cs25.models.aerodynamics.aerodynamics_landing
ComputeWingArea (class in fas-	module, 21
toad_cs25.models.loops.compute_wing_area),	
51	fastoad_cs25.models.aerodynamics.aerodynamics_low_speed
ComputeWingCG (class in fas-	module, 22
toad_cs25.models.weight.cg.cg_components.compute_wing_cg),	
64	fastoad_cs25.models.aerodynamics.aerodynamics_takeoff
ComputeWingGeometry (class in fas-	module, 23
toad_cs25.models.weight.cg.cg_components.compute_wing_geometry),	
64	fastoad_cs25.models.aerodynamics.components
ComputeWingGeometry (class in fas-	fastoad_cs25.models.aerodynamics.components.cd0
toad_cs25.models.geometry.geom_components.wing.components.compute_wing_geometry),	
37	fastoad_cs25.models.aerodynamics.components.cd0_fuselage
ComputeWingGeometry (class in fas-	module, 8
toad_cs25.models.geometry.geom_components.wing.components.compute_wing_geometry),	
40	fastoad_cs25.models.aerodynamics.components.cd0_ht
ComputeWingPosition (class in fas-	module, 10
toad_cs25.models.loops.compute_wing_position),	
52	fastoad_cs25.models.aerodynamics.components.cd0_nacelles_p
ComputeXWing (class in fas-	module, 10
toad_cs25.models.geometry.geom_components.wing.components.compute_x_wing),	
39	fastoad_cs25.models.aerodynamics.components.cd0_total
ComputeYWing (class in fas-	module, 10
toad_cs25.models.geometry.geom_components.wing.components.compute_y_wing),	
40	fastoad_cs25.models.aerodynamics.components.cd0_vt
Coordinates2D (class in fas-	module, 11
toad_cs25.models.geometry.profiles.profile),	
43	fastoad_cs25.models.aerodynamics.components.cd0_wing
CrewWeight (class in fas-	module, 12
toad_cs25.models.weight.mass_breakdown.e_crew_weight),	
86	fastoad_cs25.models.aerodynamics.components.cd_compressibi
D	module, 12
diameter (fastoad_cs25.models.geometry.geom_components.diameter property),	fastoad_cs25.models.aerodynamics.components.cd_trim
29	module, 13
E	fastoad_cs25.models.aerodynamics.components.compute_alpha
EmpennageWeight (class in fas-	module, 14
toad_cs25.models.weight.mass_breakdown.a_airframe.empennage_weight),	
70	fastoad_cs25.models.aerodynamics.components.compute_cl_alpha
EngineWeight (class in fas-	module, 14
toad_cs25.models.weight.mass_breakdown.b_propulsion.engine_weight),	
	fastoad_cs25.models.aerodynamics.components.compute_max_cl
	module, 15
	fastoad_cs25.models.aerodynamics.components.compute_polar
	module, 16
	fastoad_cs25.models.aerodynamics.components.compute_reynol
	module, 16
	fastoad_cs25.models.aerodynamics.components.high_lift_aero
	module, 17
	fastoad_cs25.models.aerodynamics.components.initialize_cl
	module, 18
	fastoad_cs25.models.aerodynamics.components.oswald

module, 18	module, 34
fastoad_cs25.models.aerodynamics.components.util	fastoad_cs25.models.geometry.geom_components.vt.components
module, 8	module, 33
fastoad_cs25.models.aerodynamics.components.util	fastoad_cs25.models.geometry.geom_components.vt.components
module, 6	module, 30
fastoad_cs25.models.aerodynamics.components.util	fastoad_cs25.models.geometry.geom_components.vt.components
module, 7	module, 31
fastoad_cs25.models.aerodynamics.constants	fastoad_cs25.models.geometry.geom_components.vt.components
module, 23	module, 32
fastoad_cs25.models.aerodynamics.external	fastoad_cs25.models.geometry.geom_components.vt.components
module, 20	module, 32
fastoad_cs25.models.aerodynamics.external.xfoil	fastoad_cs25.models.geometry.geom_components.vt.components
module, 20	module, 33
fastoad_cs25.models.aerodynamics.external.xfoil	fastoad_cs25.models.geometry.geom_components.vt.compute_v
module, 20	module, 33
fastoad_cs25.models.aerodynamics.external.xfoil	fastoad_cs25.models.geometry.geom_components.wing
module, 20	module, 42
fastoad_cs25.models.constants	fastoad_cs25.models.geometry.geom_components.wing.componer
module, 90	module, 40
fastoad_cs25.models.geometry	fastoad_cs25.models.geometry.geom_components.wing.componer
module, 46	module, 34
fastoad_cs25.models.geometry.compute_aero_center	fastoad_cs25.models.geometry.geom_components.wing.componer
module, 44	module, 35
fastoad_cs25.models.geometry.constants	fastoad_cs25.models.geometry.geom_components.wing.componer
module, 45	module, 35
fastoad_cs25.models.geometry.geom_components	fastoad_cs25.models.geometry.geom_components.wing.componer
module, 43	module, 36
fastoad_cs25.models.geometry.geom_components.computed_wing	fastoad_cs25.models.geometry.geom_components.wing.componer
module, 42	module, 36
fastoad_cs25.models.geometry.geom_components.fuselage	fastoad_cs25.models.geometry.geom_components.wing.componer
module, 26	module, 37
fastoad_cs25.models.geometry.geom_components.fuselage	fastoad_cs25.models.geometry.geom_components.wing.componer
module, 24	module, 37
fastoad_cs25.models.geometry.geom_components.fuselage	fastoad_cs25.models.geometry.geom_components.wing.componer
module, 25	module, 38
fastoad_cs25.models.geometry.geom_components.hf	fastoad_cs25.models.geometry.geom_components.wing.componer
module, 29	module, 39
fastoad_cs25.models.geometry.geom_components.hf	fastoad_cs25.models.geometry.geom_components.wing.componer
module, 28	module, 39
fastoad_cs25.models.geometry.geom_components.hf	fastoad_cs25.models.geometry.geom_components.wing.componer
module, 26	module, 40
fastoad_cs25.models.geometry.geom_components.hf	fastoad_cs25.models.geometry.geom_components.wing.compute
module, 27	module, 40
fastoad_cs25.models.geometry.geom_components.hf	fastoad_cs25.models.geometry.geom_components.wing.constant
module, 27	module, 41
fastoad_cs25.models.geometry.geom_components.hf	fastoad_cs25.models.geometry.geom_components.wing.wing_glo
module, 28	module, 41
fastoad_cs25.models.geometry.geom_components.hf	fastoad_cs25.models.geometry.geom_components.wing.wing_glo
module, 28	module, 41
fastoad_cs25.models.geometry.geom_components.hf	fastoad_cs25.models.geometry.geom_components.wing.wing_glo
module, 28	module, 45
fastoad_cs25.models.geometry.geom_components.naca	fastoad_cs25.models.geometry.profiles
module, 30	module, 44
fastoad_cs25.models.geometry.geom_components.naca	fastoad_cs25.models.geometry.profiles.profile
module, 29	module, 43
fastoad_cs25.models.geometry.geom_components.vf	fastoad_cs25.models.geometry.profiles.profile_getter

module, 44	module, 62
fastoad_cs25.models.handling_qualities	fastoad_cs25.models.weight.cg.cg_components.compute_cg_tare
module, 48	module, 64
fastoad_cs25.models.handling_qualities.compute_cg_tare	fastoad_cs25.models.weight.cg.cg_components.compute_cg_wing
module, 48	module, 64
fastoad_cs25.models.handling_qualities.tail_size	fastoad_cs25.models.weight.cg.cg_components.compute_global_cg
module, 48	module, 65
fastoad_cs25.models.handling_qualities.tail_size_area	fastoad_cs25.models.weight.cg.cg_components.compute_ht_cg
module, 46	module, 65
fastoad_cs25.models.handling_qualities.tail_size_area_max	fastoad_cs25.models.weight.cg.cg_components.compute_max_cg
module, 47	module, 66
fastoad_cs25.models.handling_qualities.tail_size_area_vt	fastoad_cs25.models.weight.cg.cg_components.compute_vt_cg
module, 47	module, 66
fastoad_cs25.models.loops	fastoad_cs25.models.weight.cg.cg_components.load_cases
module, 52	module, 61
fastoad_cs25.models.loops.compute_wing_area	fastoad_cs25.models.weight.cg.cg_components.load_cases.compute
module, 51	module, 58
fastoad_cs25.models.loops.compute_wing_position	fastoad_cs25.models.weight.cg.cg_components.load_cases.compute
module, 52	module, 59
fastoad_cs25.models.loops.constants	fastoad_cs25.models.weight.cg.cg_components.load_cases.compute
module, 52	module, 59
fastoad_cs25.models.loops.wing_area_component	fastoad_cs25.models.weight.cg.cg_components.load_cases.compute
module, 51	module, 59
fastoad_cs25.models.loops.wing_area_component.compute	fastoad_cs25.models.weight.cg.cg_components.load_cases.compute
module, 49	module, 60
fastoad_cs25.models.loops.wing_area_component.compute	fastoad_cs25.models.weight.cg.cg_components.load_cases.compute
module, 50	module, 60
fastoad_cs25.models.propulsion	fastoad_cs25.models.weight.cg.cg_components.update_mlg
module, 58	module, 67
fastoad_cs25.models.propulsion.fuel_propulsion	fastoad_cs25.models.weight.cg.constants
module, 58	module, 69
fastoad_cs25.models.propulsion.fuel_propulsion	fastoad_cs25.models.weight.constants
module, 58	module, 90
fastoad_cs25.models.propulsion.fuel_propulsion	fastoad_cs25.models.mass_breakdown
module, 53	module, 90
fastoad_cs25.models.propulsion.fuel_propulsion	fastoad_cs25.models.mass_breakdown.a_airframe
module, 53	module, 74
fastoad_cs25.models.propulsion.fuel_propulsion	fastoad_cs25.models.mass_breakdown.a_airframe.a1_wing
module, 53	module, 69
fastoad_cs25.models.propulsion.fuel_propulsion	fastoad_cs25.models.mass_breakdown.a_airframe.a2_fuselage
module, 54	module, 70
fastoad_cs25.models.weight	fastoad_cs25.models.mass_breakdown.a_airframe.a3_engine
module, 90	module, 70
fastoad_cs25.models.weight.cg	fastoad_cs25.models.mass_breakdown.a_airframe.a4_fuel
module, 69	module, 71
fastoad_cs25.models.weight.cg.cg	fastoad_cs25.models.mass_breakdown.a_airframe.a5_landing
module, 68	module, 72
fastoad_cs25.models.weight.cg.cg_components	fastoad_cs25.models.mass_breakdown.a_airframe.a6_payload
module, 68	module, 72
fastoad_cs25.models.weight.cg.cg_components.compute	fastoad_cs25.models.mass_breakdown.a_airframe.a7_passenger
module, 61	module, 73
fastoad_cs25.models.weight.cg.cg_components.compute	fastoad_cs25.models.mass_breakdown.a_airframe.constants
module, 62	module, 73
fastoad_cs25.models.weight.cg.cg_components.compute	fastoad_cs25.models.mass_breakdown.a_airframe.sum
module, 62	module, 73





`get_relative_thickness()` (fastoad\_cs25.models.geometry.profiles.profile.Profile method), 43  
`get_sides()` (fastoad\_cs25.models.geometry.profiles.profile.Profile method), 44  
`get_upper_side()` (fastoad\_cs25.models.geometry.profiles.profile.Profile method), 43  
`get_wrapper()` (fastoad\_cs25.models.propulsion.fuel\_propulsion.fuel\_propulsion.rubber\_engine.rubber\_engine.ComputeV method), 54  
**H**  
**HIGH\_SPEED** (fastoad\_cs25.models.aerodynamics.constants.PolarType attribute), 23  
**I**  
**InducedDragCoefficient** (class in fastoad\_cs25.models.aerodynamics.components.oswald), 18  
`initialize()` (fastoad\_cs25.models.aerodynamics.aerodynamics\_landing.AerodynamicsLanding method), 21  
`initialize()` (fastoad\_cs25.models.aerodynamics.components.cd0.cd0 method), 8  
`initialize()` (fastoad\_cs25.models.aerodynamics.components.cd0\_installed\_weight.Cd0InstalledWeight method), 8  
`initialize()` (fastoad\_cs25.models.aerodynamics.components.cd0\_interaction.Cd0Interaction method), 9  
`initialize()` (fastoad\_cs25.models.aerodynamics.components.cd0\_nacelles\_and\_pylons.Cd0NacellesAndPylons method), 10  
`initialize()` (fastoad\_cs25.models.aerodynamics.components.cd0\_total.Cd0Total method), 10  
`initialize()` (fastoad\_cs25.models.aerodynamics.components.cd0\_wing.Cd0Wing method), 11  
`initialize()` (fastoad\_cs25.models.aerodynamics.components.cd\_trim.CdTrim method), 13  
`initialize()` (fastoad\_cs25.models.aerodynamics.components.compute\_alpha.ComputeAlpha method), 14  
`initialize()` (fastoad\_cs25.models.aerodynamics.components.compute\_cl\_alpha.ComputeCLAlpha method), 14  
`initialize()` (fastoad\_cs25.models.aerodynamics.components.compute\_polar.ComputePolar method), 16  
`initialize()` (fastoad\_cs25.models.aerodynamics.components.compute\_reynolds.ComputeReynolds method), 16  
`initialize()` (fastoad\_cs25.models.aerodynamics.components.initialize\_cl.InitializeCLPolar method), 18  
`initialize()` (fastoad\_cs25.models.aerodynamics.components.oswald.InducedDragCoefficient method), 18  
`initialize()` (fastoad\_cs25.models.aerodynamics.components.oswald.OswaldCoefficient method), 19  
`initialize()` (fastoad\_cs25.models.aerodynamics.external.xfoil.xfoil\_polar.XfoilPolar method), 20  
`initialize()` (fastoad\_cs25.models.geometry.geom\_components.wing.Wing method), 20  
`initialize()` (fastoad\_cs25.models.geometry.geometry.Geometry method), 45  
`initialize()` (fastoad\_cs25.models.handling\_qualities.compute\_static\_moment.ComputeStaticMoment method), 48  
`initialize()` (fastoad\_cs25.models.mass\_balance.mass\_balance.ComputeV method), 51  
`initialize()` (fastoad\_cs25.models.weight.cg.cg\_components.compute\_cg.ComputeCG method), 63  
`initialize()` (fastoad\_cs25.models.weight.cg.cg\_components.compute\_cg.ComputeCG method), 64  
`initialize()` (fastoad\_cs25.models.weight.cg.cg\_components.load\_cases.LoadCases method), 60  
`initialize()` (fastoad\_cs25.models.weight.mass\_breakdown.mass\_breakdown.mass\_breakdown method), 88  
`initialize()` (fastoad\_cs25.models.weight.weight.Weight method), 90  
`InitializeCIPolar` (class in fastoad\_cs25.models.aerodynamics.components.initialize\_cl), 18  
`installed_weight()` (fastoad\_cs25.models.aerodynamics.components.cd0\_installed\_weight.Cd0InstalledWeight method), 57  
`interaction_coeff` (fastoad\_cs25.models.aerodynamics.components.cd0\_interaction.Cd0Interaction attribute), 7  
**LANDING** (fastoad\_cs25.models.aerodynamics.constants.PolarType attribute), 23  
**LandingGearWeight** (class in fastoad\_cs25.models.weight.mass\_breakdown.a\_airframe.a5\_landing\_gear.LandingGearWeight), 73  
`length` (fastoad\_cs25.models.geometry.geom\_components.nacelle\_pylons.NacellePylons attribute), 28  
`length` (fastoad\_cs25.models.geometry.geom\_components.nacelle\_pylons.NacellePylons attribute), 29  
`length()` (fastoad\_cs25.models.propulsion.fuel\_propulsion.rubber\_engine.rubber\_engine.ComputePolar method), 57  
**LifeSupportSystemsWeight** (class in fastoad\_cs25.models.weight.mass\_breakdown.c\_systems.c2\_life\_support.LifeSupportSystemsWeight), 78  
**LiftingSurfaceGeometry** (class in fastoad\_cs25.models.aerodynamics.components.utils.cd0\_lifting\_surface.LiftingSurfaceGeometry), 61  
**Loads** (class in fastoad\_cs25.models.weight.mass\_breakdown.cs25.Loads), 87  
**LOW\_SPEED** (fastoad\_cs25.models.aerodynamics.constants.PolarType attribute), 23

## M

`MAC_length(fastoad_cs25.models.aerodynamics.components.utils.cd_lifting_surface.LiftingSurfaceGeometry attribute), 7`  
`MassBreakdown (class in fastoad_cs25.models.weight.mass_breakdown.mass_breakdown), 17`  
`max_thrust(fastoad_cs25.models.geometry.geom_components.nacelle_pylons.compute_nacelle_pylons.Nacelle attribute), 29`  
`max_thrust() (fastoad_cs25.models.propulsion.fuel_propulsion.rubber_engine.RubberEngine method), 57`  
`MaxCGRatiosForLoadCases (class in fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcases), 61`  
`module`  
`fastoad_cs25, 91`  
`fastoad_cs25.configurations, 6`  
`fastoad_cs25.models, 91`  
`fastoad_cs25.models.aerodynamics, 24`  
`fastoad_cs25.models.aerodynamics.aerodynamics_high_speed, 20`  
`fastoad_cs25.models.aerodynamics.aerodynamics_landing, 21`  
`fastoad_cs25.models.aerodynamics.aerodynamics_low_speed, 22`  
`fastoad_cs25.models.aerodynamics.aerodynamics_takeoff, 23`  
`fastoad_cs25.models.aerodynamics.components, 19`  
`fastoad_cs25.models.aerodynamics.components.cd0, 8`  
`fastoad_cs25.models.aerodynamics.components.cd0_fuselage, 8`  
`fastoad_cs25.models.aerodynamics.components.cd0_ht, 9`  
`fastoad_cs25.models.aerodynamics.components.cd0_nacelles_pylons, 10`  
`fastoad_cs25.models.aerodynamics.components.cd0_total, 10`  
`fastoad_cs25.models.aerodynamics.components.cd0_vt, 11`  
`fastoad_cs25.models.aerodynamics.components.cd0_wing, 12`  
`fastoad_cs25.models.aerodynamics.components.cd_compressibility, 12`  
`fastoad_cs25.models.aerodynamics.components.cd_trim, 13`  
`fastoad_cs25.models.aerodynamics.components.compute_alpha, 14`  
`fastoad_cs25.models.aerodynamics.components.compute_cl_alpha, 14`  
`fastoad_cs25.models.aerodynamics.components.compute_max_cl_landing, 15`  
`fastoad_cs25.models.aerodynamics.components.compute_polar, 16`  
`fastoad_cs25.models.aerodynamics.components.compute_re, 16`  
`fastoad_cs25.models.aerodynamics.components.high_lift, 17`  
`fastoad_cs25.models.aerodynamics.components.initialize, 18`  
`fastoad_cs25.models.aerodynamics.components.oswald, 18`  
`fastoad_cs25.models.aerodynamics.components.utils, 8`  
`fastoad_cs25.models.aerodynamics.components.utils.cd0, 6`  
`fastoad_cs25.models.aerodynamics.components.utils.fric, 7`  
`fastoad_cs25.models.aerodynamics.constants, 23`  
`fastoad_cs25.models.aerodynamics.external, 20`  
`fastoad_cs25.models.aerodynamics.external.xfoil, 20`  
`fastoad_cs25.models.aerodynamics.external.xfoil.xfoil6, 20`  
`fastoad_cs25.models.aerodynamics.external.xfoil.xfoil_10, 20`  
`fastoad_cs25.models.aerodynamics.external.xfoil.xfoil_15, 20`  
`fastoad_cs25.models.constants, 90`  
`fastoad_cs25.models.geometry, 46`  
`fastoad_cs25.models.geometry.compute_aero_center, 44`  
`fastoad_cs25.models.geometry.constants, 45`  
`fastoad_cs25.models.geometry.geom_components, 43`  
`fastoad_cs25.models.geometry.geom_components.compute_wing, 42`  
`fastoad_cs25.models.geometry.geom_components.fuselage, 26`  
`fastoad_cs25.models.geometry.geom_components.fuselage.2d, 24`  
`fastoad_cs25.models.geometry.geom_components.fuselage.3d, 25`  
`fastoad_cs25.models.geometry.geom_components.ht, 29`  
`fastoad_cs25.models.geometry.geom_components.ht.component, 28`  
`fastoad_cs25.models.geometry.geom_components.ht.component.2d, 26`  
`fastoad_cs25.models.geometry.geom_components.ht.component.3d, 27`  
`fastoad_cs25.models.geometry.geom_components.ht.component.compute_alpha, 27`  
`fastoad_cs25.models.geometry.geom_components.ht.component.compute_cl_alpha, 27`  
`fastoad_cs25.models.geometry.geom_components.ht.component.compute_max_cl_landing, 28`  
`fastoad_cs25.models.geometry.geom_components.ht.component.compute_polar, 28`

fastoad\_cs25.models.geometry.geom\_components.nfastoad\_cs25.models.geometry.profiles.profile,  
 30 43  
 fastoad\_cs25.models.geometry.geom\_components.nfastoad\_cs25.models.geometry.profiles.profile\_getter,  
 29 44  
 fastoad\_cs25.models.geometry.geom\_components.vfastoad\_cs25.models.handling\_qualities,  
 34 48  
 fastoad\_cs25.models.geometry.geom\_components.vfastoad\_cs25.models.handling\_qualities.compute\_static\_  
 33 48  
 fastoad\_cs25.models.geometry.geom\_components.vfastoad\_cs25.models.handling\_qualities.tail\_sizing,  
 30 48  
 fastoad\_cs25.models.geometry.geom\_components.vfastoad\_cs25.models.handling\_qualities.tail\_sizing.com  
 31 46  
 fastoad\_cs25.models.geometry.geom\_components.vfastoad\_cs25.models.handling\_qualities.tail\_sizing.com  
 32 47  
 fastoad\_cs25.models.geometry.geom\_components.vfastoad\_cs25.models.handling\_qualities.tail\_sizing.com  
 32 47  
 fastoad\_cs25.models.geometry.geom\_components.vfastoad\_cs25.models.loops.sweep,  
 33 fastoad\_cs25.models.loops.compute\_wing\_area,  
 fastoad\_cs25.models.geometry.geom\_components.vt.compute\_vertical\_tail,  
 33 fastoad\_cs25.models.loops.compute\_wing\_position,  
 fastoad\_cs25.models.geometry.geom\_components.wing,52  
 42 fastoad\_cs25.models.loops.constants, 52  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components,51  
 40 51  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components.update\_wing\_area\_component,50  
 34 49  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components.update\_wing\_area\_component,50  
 35 50  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components.update\_wing\_area\_component,58  
 35 fastoad\_cs25.models.propulsion.fuel\_propulsion,  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components.compute\_mac\_wing,  
 36 fastoad\_cs25.models.propulsion.fuel\_propulsion.rubber\_  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components.compute\_mfw,  
 36 fastoad\_cs25.models.propulsion.fuel\_propulsion.rubber\_  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components.compute\_planform,  
 37 fastoad\_cs25.models.propulsion.fuel\_propulsion.rubber\_  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components.compute\_sweep\_wing,  
 37 fastoad\_cs25.models.propulsion.fuel\_propulsion.rubber\_  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components.compute\_toc\_wing,  
 38 fastoad\_cs25.models.propulsion.fuel\_propulsion.rubber\_  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components.compute\_wet\_area\_wing,  
 39 fastoad\_cs25.models.weight, 90  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components.compute\_weight\_wing, 69  
 39 fastoad\_cs25.models.weight.cg.cg, 68  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components.compute\_weight\_wing, cg\_components,  
 40 68  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components.compute\_weight\_wing, cg\_components, compute\_cg  
 40 61  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components.compute\_weight\_wing, cg\_components, compute\_cg  
 41 62  
 fastoad\_cs25.models.geometry.geom\_components.wing\_components.compute\_weight\_wing, cg\_components, compute\_cg  
 41 62  
 fastoad\_cs25.models.geometry.geometry, 45 fastoad\_cs25.models.weight.cg.cg\_components.compute\_cg  
 fastoad\_cs25.models.geometry.profiles, 44 64

```

fastoad_cs25.models.weight.cg.cg_components.compute_cg_wing,
64 fastoad_cs25.models.weight.mass_breakdown.b_propulsion
fastoad_cs25.models.weight.cg.cg_components.compute_global_cg,
65 fastoad_cs25.models.weight.mass_breakdown.b_propulsion
fastoad_cs25.models.weight.cg.cg_components.compute_ht_cg,
65 fastoad_cs25.models.weight.mass_breakdown.b_propulsion
fastoad_cs25.models.weight.cg.cg_components.compute_max_cg_ratio,
66 fastoad_cs25.models.weight.mass_breakdown.b_propulsion
fastoad_cs25.models.weight.cg.cg_components.compute_vt_cg,
66 fastoad_cs25.models.weight.mass_breakdown.c_systems,
fastoad_cs25.models.weight.cg.cg_components.load_cases,
61 fastoad_cs25.models.weight.mass_breakdown.c_systems.c1
fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase1,
58 fastoad_cs25.models.weight.mass_breakdown.c_systems.c2
fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase2,
59 fastoad_cs25.models.weight.mass_breakdown.c_systems.c3
fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase3,
59 fastoad_cs25.models.weight.mass_breakdown.c_systems.c4
fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase4,
59 fastoad_cs25.models.weight.mass_breakdown.c_systems.c5
fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcase_base,
60 fastoad_cs25.models.weight.mass_breakdown.c_systems.c6
fastoad_cs25.models.weight.cg.cg_components.load_cases.compute_cg_loadcases,
60 fastoad_cs25.models.weight.mass_breakdown.c_systems.co
fastoad_cs25.models.weight.cg.cg_components.update_inlg,
67 fastoad_cs25.models.weight.mass_breakdown.c_systems.su
fastoad_cs25.models.weight.cg.constants,
69 fastoad_cs25.models.weight.mass_breakdown.constants,
fastoad_cs25.models.weight.constants, 90 87
fastoad_cs25.models.weight.mass_breakdown, fastoad_cs25.models.weight.mass_breakdown.cs25,
90 87
fastoad_cs25.models.weight.mass_breakdown.a_iframed fastoad_cs25.models.weight.mass_breakdown.d_furniture,
74 86
fastoad_cs25.models.weight.mass_breakdown.a_iframed a1s25 modelight fastoad_cs25.models.weight.mass_breakdown.d_furniture,
69 82
fastoad_cs25.models.weight.mass_breakdown.a_iframed a2s25 modelight fastoad_cs25.models.weight.mass_breakdown.d_furniture,
70 82
fastoad_cs25.models.weight.mass_breakdown.a_iframed a3s25 modelight fastoad_cs25.models.weight.mass_breakdown.d_furniture,
70 83
fastoad_cs25.models.weight.mass_breakdown.a_iframed a4s25 modelight fastoad_cs25.models.weight.mass_breakdown.d_furniture,
71 83
fastoad_cs25.models.weight.mass_breakdown.a_iframed a5s25 modelight fastoad_cs25.models.weight.mass_breakdown.d_furniture,
72 84
fastoad_cs25.models.weight.mass_breakdown.a_iframed a6s25 modelight fastoad_cs25.models.weight.mass_breakdown.d_furniture,
72 84
fastoad_cs25.models.weight.mass_breakdown.a_iframed a7s25 modelight fastoad_cs25.models.weight.mass_breakdown.d_furniture,
73 85
fastoad_cs25.models.weight.mass_breakdown.a_iframed cs25 modelight fastoad_cs25.models.weight.mass_breakdown.e_crew,
73 86
fastoad_cs25.models.weight.mass_breakdown.a_iframed cs25 modelight fastoad_cs25.models.weight.mass_breakdown.e_crew.crew,
74 86
fastoad_cs25.models.weight.mass_breakdown.b_propulsion fastoad_cs25.models.weight.mass_breakdown.mass_breakdo
77 87
fastoad_cs25.models.weight.mass_breakdown.b_propulsion cs25 modelight fastoad_cs25.models.weight.mass_breakdown.payload,

```

88  
 fastoad\_cs25.models.weight.mass\_breakdown.update\_mlw\_and\_mzfw, (class in fas-  
 89 RubberEngine  
 fastoad\_cs25.models.weight.weight, 90 toad\_cs25.models.propulsion.fuel\_propulsion.rubber\_engine.rub  
 54

**N** **S**

Nacelle (class in fas- SecurityKitWeight (class in fas-  
 toad\_cs25.models.geometry.geom\_components.nacelle\_pylons, nacelle\_pylons),  
 29 toad\_cs25.models.weight.mass\_breakdown.d\_furniture.d4\_securi  
 84  
 nacelle\_diameter() (fas- set\_points() (fastoad\_cs25.models.geometry.profiles.profile.Profile  
 toad\_cs25.models.propulsion.fuel\_propulsion.rubber\_engine.rubber\_engine.RubberEngine  
 method), 57 method), 438  
 NavigationSystemsWeight (class in fas- setup() (fastoad\_cs25.models.aerodynamics.aerodynamics\_high\_speed.Ae  
 toad\_cs25.models.weight.mass\_breakdown.c\_systems.c3\_navigation\_systems\_weight),  
 79 setup() (fastoad\_cs25.models.aerodynamics.aerodynamics\_landing.Aerod  
 method), 21  
 setup() (fastoad\_cs25.models.aerodynamics.aerodynamics\_landing.Comp  
 method), 22  
 OMRubberEngineComponent (class in fas- setup() (fastoad\_cs25.models.aerodynamics.aerodynamics\_landing.Comp  
 toad\_cs25.models.propulsion.fuel\_propulsion.rubber\_engine.openloop),  
 54 method), 20  
 OMRubberEngineWrapper (class in fas- setup() (fastoad\_cs25.models.aerodynamics.aerodynamics\_low\_speed.Ae  
 toad\_cs25.models.propulsion.fuel\_propulsion.rubber\_engine.openloop),  
 53 setup() (fastoad\_cs25.models.aerodynamics.aerodynamics\_takeoff.Aerod  
 method), 23  
 OperatingWeightEmpty (class in fas- setup() (fastoad\_cs25.models.aerodynamics.components.cd0.CD0  
 toad\_cs25.models.weight.mass\_breakdown.mass\_breakdown),  
 88 method), 8  
 OswaldCoefficient (class in fas- setup() (fastoad\_cs25.models.aerodynamics.components.cd0\_fuselage.Cd  
 toad\_cs25.models.aerodynamics.components.oswald),  
 19 method), 8  
 output\_name (fastoad\_cs25.models.weight.cg.cg\_components.load\_case\_components.load\_case\_base.ComputeCGLoadCase  
 property), 60 setup() (fastoad\_cs25.models.aerodynamics.components.cd0\_nacelles\_py  
 method), 10  
 setup() (fastoad\_cs25.models.aerodynamics.components.cd0\_total.Cd0To  
 method), 10

**P**

PaintWeight (class in fas- setup() (fastoad\_cs25.models.aerodynamics.components.cd0\_vt.Cd0Vertic  
 toad\_cs25.models.weight.mass\_breakdown.a\_airframe.a7\_paint\_weight),  
 73 method), 10  
 setup() (fastoad\_cs25.models.aerodynamics.components.cd0\_wing.Cd0W  
 method), 12  
 PassengerSeatsWeight (class in fas- setup() (fastoad\_cs25.models.aerodynamics.components.cd\_compressibil  
 toad\_cs25.models.weight.mass\_breakdown.d\_furniture.d4\_passenger\_seats\_weight),  
 83 method), 12  
 PolarType (class in fas- setup() (fastoad\_cs25.models.aerodynamics.components.cd\_trim.CdTrim  
 toad\_cs25.models.aerodynamics.constants),  
 23 method), 13  
 setup() (fastoad\_cs25.models.aerodynamics.components.compute\_alpha.  
 method), 14  
 PowerSystemsWeight (class in fas- setup() (fastoad\_cs25.models.aerodynamics.components.compute\_cl\_alph  
 toad\_cs25.models.weight.mass\_breakdown.c\_systems.c3\_power\_systems\_weight),  
 77 method), 14  
 Profile (class in fas- setup() (fastoad\_cs25.models.aerodynamics.components.compute\_max\_cl  
 toad\_cs25.models.geometry.profiles.profile),  
 43 method), 15  
 setup() (fastoad\_cs25.models.aerodynamics.components.compute\_polar.C  
 method), 16  
 PropulsionWeight (class in fas- setup() (fastoad\_cs25.models.aerodynamics.components.compute\_reynol  
 toad\_cs25.models.weight.mass\_breakdown.b\_propulsion.b1\_weight),  
 76 method), 16  
 PylonsWeight (class in fas- setup() (fastoad\_cs25.models.aerodynamics.components.high\_lift\_aero.C  
 toad\_cs25.models.weight.mass\_breakdown.a\_airframe.a6\_pylons\_weight),  
 72 method), 17



```

setup() (fastoad_cs25.models.aerodynamics.components.insert_inlet_flow method), 18
setup() (fastoad_cs25.models.aerodynamics.components.oscillating_flow method), 18
setup() (fastoad_cs25.models.aerodynamics.components.oscillating_flow method), 19
setup() (fastoad_cs25.models.aerodynamics.external.xfoil_setup method), 20
setup() (fastoad_cs25.models.geometry.compute_aero_center method), 44
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 42
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 24
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 25
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 25
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 26
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 27
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 27
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 28
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 28
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 30
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 30
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 31
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 32
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 32
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 33
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 33
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 34
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 35
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 35
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 36
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 36
setup() (fastoad_cs25.models.geometry.geom_components.setup method), 37

```

setup() (fastoad\_cs25.models.weight.cg.cg\_components.compute\_cg\_components method), 62  
 setup() (fastoad\_cs25.models.weight.cg.cg\_components.compute\_cg\_components method), 64  
 setup() (fastoad\_cs25.models.weight.cg.cg\_components.compute\_cg\_components method), 64  
 setup() (fastoad\_cs25.models.weight.cg.cg\_components.compute\_cg\_components method), 65  
 setup() (fastoad\_cs25.models.weight.cg.cg\_components.compute\_cg\_components method), 65  
 setup() (fastoad\_cs25.models.weight.cg.cg\_components.compute\_cg\_components method), 66  
 setup() (fastoad\_cs25.models.weight.cg.cg\_components.compute\_cg\_components method), 66  
 setup() (fastoad\_cs25.models.weight.cg.cg\_components.load\_case method), 58  
 setup() (fastoad\_cs25.models.weight.cg.cg\_components.load\_case method), 59  
 setup() (fastoad\_cs25.models.weight.cg.cg\_components.load\_case method), 59  
 setup() (fastoad\_cs25.models.weight.cg.cg\_components.load\_case method), 59  
 setup() (fastoad\_cs25.models.weight.cg.cg\_components.load\_case method), 60  
 setup() (fastoad\_cs25.models.weight.cg.cg\_components.load\_case method), 60  
 setup() (fastoad\_cs25.models.weight.cg.cg\_components.load\_case method), 61  
 setup() (fastoad\_cs25.models.weight.cg.cg\_components.update method), 67  
 setup() (fastoad\_cs25.models.weight.mass\_breakdown.a\_compute method), 69  
 setup() (fastoad\_cs25.models.weight.mass\_breakdown.a\_compute method), 70  
 setup() (fastoad\_cs25.models.weight.mass\_breakdown.a\_compute method), 70  
 setup() (fastoad\_cs25.models.weight.mass\_breakdown.a\_compute method), 71  
 setup() (fastoad\_cs25.models.weight.mass\_breakdown.a\_compute method), 72  
 setup() (fastoad\_cs25.models.weight.mass\_breakdown.a\_compute method), 72  
 setup() (fastoad\_cs25.models.weight.mass\_breakdown.a\_compute method), 73  
 setup() (fastoad\_cs25.models.weight.mass\_breakdown.a\_compute method), 74  
 setup() (fastoad\_cs25.models.weight.mass\_breakdown.b\_compute method), 74  
 setup() (fastoad\_cs25.models.weight.mass\_breakdown.b\_compute method), 75  
 setup() (fastoad\_cs25.models.weight.mass\_breakdown.b\_compute method), 75  
 setup() (fastoad\_cs25.models.weight.mass\_breakdown.b\_compute method), 76

<i>method</i> ), 10	<i>method</i> ), 25	
setup_partials( toad_cs25.models.aerodynamics.components.cd0_total.Cd0Total	(fas- setup_partials( toad_cs25.models.aerodynamics.components.cd0_total.Cd0Total	(fas- setup_partials( toad_cs25.models.aerodynamics.components.cd0_total.Cd0Total
<i>method</i> ), 10	<i>method</i> ), 25	<i>method</i> ), 25
setup_partials( toad_cs25.models.aerodynamics.components.cd0_vt.Cd0Vertical	(fas- setup_partials( toad_cs25.models.aerodynamics.components.cd0_vt.Cd0Vertical	(fas- setup_partials( toad_cs25.models.aerodynamics.components.cd0_vt.Cd0Vertical
<i>method</i> ), 11	<i>method</i> ), 26	<i>method</i> ), 26
setup_partials( toad_cs25.models.aerodynamics.components.cd0_wing.Cd0Wing	(fas- setup_partials( toad_cs25.models.aerodynamics.components.cd0_wing.Cd0Wing	(fas- setup_partials( toad_cs25.models.aerodynamics.components.cd0_wing.Cd0Wing
<i>method</i> ), 12	<i>method</i> ), 27	<i>method</i> ), 27
setup_partials( toad_cs25.models.aerodynamics.components.cd_compressibility.CdCompressibility	(fas- setup_partials( toad_cs25.models.aerodynamics.components.cd_compressibility.CdCompressibility	(fas- setup_partials( toad_cs25.models.aerodynamics.components.cd_compressibility.CdCompressibility
<i>method</i> ), 12	<i>method</i> ), 27	<i>method</i> ), 27
setup_partials( toad_cs25.models.aerodynamics.components.cd_trim.CdTrim	(fas- setup_partials( toad_cs25.models.aerodynamics.components.cd_trim.CdTrim	(fas- setup_partials( toad_cs25.models.aerodynamics.components.cd_trim.CdTrim
<i>method</i> ), 13	<i>method</i> ), 28	<i>method</i> ), 28
setup_partials( toad_cs25.models.aerodynamics.components.compute_alpha_compressible	(fas- setup_partials( toad_cs25.models.aerodynamics.components.compute_alpha_compressible	(fas- setup_partials( toad_cs25.models.aerodynamics.components.compute_alpha_compressible
<i>method</i> ), 14	<i>method</i> ), 30	<i>method</i> ), 30
setup_partials( toad_cs25.models.aerodynamics.components.compute_cl_alpha_compressible	(fas- setup_partials( toad_cs25.models.aerodynamics.components.compute_cl_alpha_compressible	(fas- setup_partials( toad_cs25.models.aerodynamics.components.compute_cl_alpha_compressible
<i>method</i> ), 14	<i>method</i> ), 31	<i>method</i> ), 31
setup_partials( toad_cs25.models.aerodynamics.components.compute_max_roll_rate	(fas- setup_partials( toad_cs25.models.aerodynamics.components.compute_max_roll_rate	(fas- setup_partials( toad_cs25.models.aerodynamics.components.compute_max_roll_rate
<i>method</i> ), 15	<i>method</i> ), 31	<i>method</i> ), 31
setup_partials( toad_cs25.models.aerodynamics.components.compute_polar_compressible	(fas- setup_partials( toad_cs25.models.aerodynamics.components.compute_polar_compressible	(fas- setup_partials( toad_cs25.models.aerodynamics.components.compute_polar_compressible
<i>method</i> ), 16	<i>method</i> ), 32	<i>method</i> ), 32
setup_partials( toad_cs25.models.aerodynamics.components.compute_reynolds_number	(fas- setup_partials( toad_cs25.models.aerodynamics.components.compute_reynolds_number	(fas- setup_partials( toad_cs25.models.aerodynamics.components.compute_reynolds_number
<i>method</i> ), 16	<i>method</i> ), 32	<i>method</i> ), 32
setup_partials( toad_cs25.models.aerodynamics.components.high_lift_aerodynamic	(fas- setup_partials( toad_cs25.models.aerodynamics.components.high_lift_aerodynamic	(fas- setup_partials( toad_cs25.models.aerodynamics.components.high_lift_aerodynamic
<i>method</i> ), 17	<i>method</i> ), 33	<i>method</i> ), 33
setup_partials( toad_cs25.models.aerodynamics.components.initialize_cl_initialize	(fas- setup_partials( toad_cs25.models.aerodynamics.components.initialize_cl_initialize	(fas- setup_partials( toad_cs25.models.aerodynamics.components.initialize_cl_initialize
<i>method</i> ), 18	<i>method</i> ), 34	<i>method</i> ), 34
setup_partials( toad_cs25.models.aerodynamics.components.oswald_induced_drag	(fas- setup_partials( toad_cs25.models.aerodynamics.components.oswald_induced_drag	(fas- setup_partials( toad_cs25.models.aerodynamics.components.oswald_induced_drag
<i>method</i> ), 18	<i>method</i> ), 35	<i>method</i> ), 35
setup_partials( toad_cs25.models.aerodynamics.components.oswald_lift_coefficient	(fas- setup_partials( toad_cs25.models.aerodynamics.components.oswald_lift_coefficient	(fas- setup_partials( toad_cs25.models.aerodynamics.components.oswald_lift_coefficient
<i>method</i> ), 19	<i>method</i> ), 35	<i>method</i> ), 35
setup_partials( toad_cs25.models.geometry.compute_aero_center.ComputeAeroCenter	(fas- setup_partials( toad_cs25.models.geometry.compute_aero_center.ComputeAeroCenter	(fas- setup_partials( toad_cs25.models.geometry.compute_aero_center.ComputeAeroCenter
<i>method</i> ), 44	<i>method</i> ), 36	<i>method</i> ), 36
setup_partials( toad_cs25.models.geometry.geom_components.compute_wetted_area	(fas- setup_partials( toad_cs25.models.geometry.geom_components.compute_wetted_area	(fas- setup_partials( toad_cs25.models.geometry.geom_components.compute_wetted_area
<i>method</i> ), 42	<i>method</i> ), 37	<i>method</i> ), 37
setup_partials( toad_cs25.models.geometry.geom_components.fuselage.compute_fuselage	(fas- setup_partials( toad_cs25.models.geometry.geom_components.fuselage.compute_fuselage	(fas- setup_partials( toad_cs25.models.geometry.geom_components.fuselage.compute_fuselage
<i>method</i> ), 24	<i>method</i> ), 37	<i>method</i> ), 37
setup_partials( toad_cs25.models.geometry.geom_components.fuselage.compute_fuselage	(fas- setup_partials( toad_cs25.models.geometry.geom_components.fuselage.compute_fuselage	(fas- setup_partials( toad_cs25.models.geometry.geom_components.fuselage.compute_fuselage
<i>method</i> ), 24	<i>method</i> ), 37	<i>method</i> ), 37





method), 80  
 setup\_partials() (fas- TransmissionSystemsWeight (class in fas-  
 toad\_cs25.models.weight.mass\_breakdown.c\_systems.c6\_flight\_kit\_weight.FlightKitWeight  
 mass\_breakdown.c\_systems.c4\_transmission\_weight.TransmissionWeight  
 method), 81  
 setup\_partials() (fas- U  
 toad\_cs25.models.weight.mass\_breakdown.cs25.Loads  
 method), 87  
 setup\_partials() (fas- UnconsumablesWeight (class in fas-  
 toad\_cs25.models.weight.mass\_breakdown.b\_propulsion.b3\_unconsumables\_weight.UnconsumablesWeight  
 mass\_breakdown.d\_furniture.d1\_cargo\_configuration\_weight.CargoConfigurationWeight  
 method), 82  
 setup\_partials() (fas- UpdateMLG (class in fas-  
 toad\_cs25.models.weight.cg.cg\_components.update\_mlg),  
 mass\_breakdown.d\_furniture.d2\_passenger\_seats\_weight.PassengerSeatsWeight  
 method), 83  
 setup\_partials() (fas- UpdateMLWandMZFW (class in fas-  
 toad\_cs25.models.weight.mass\_breakdown.update\_mlw\_and\_mzf  
 mass\_breakdown.d\_furniture.d3\_food\_water\_weight.FoodWaterWeight  
 method), 83  
 setup\_partials() (fas- UpdateWingAreaAero (class in fas-  
 toad\_cs25.models.loops.wing\_area\_component.update\_wing\_area  
 mass\_breakdown.d\_furniture.d4\_security\_kit\_weight.SecurityKitWeight  
 method), 84  
 setup\_partials() (fas- UpdateWingAreaGeom (class in fas-  
 toad\_cs25.models.loops.wing\_area\_component.update\_wing\_area  
 mass\_breakdown.d\_furniture.d5\_toilets\_weight.ToiletsWeight  
 method), 85  
 setup\_partials() (fas- W  
 toad\_cs25.models.weight.mass\_breakdown.e\_crew\_weight.CrewWeight  
 mass\_breakdown.cs25.models.weight.weight), 90  
 method), 86  
 setup\_partials() (fas- wet\_area (fastoad\_cs25.models.aerodynamics.components.utils.cd0\_lifting  
 mass\_breakdown.payload\_compute\_payload  
 method), 88  
 setup\_partials() (fas- wetted\_area (fastoad\_cs25.models.geometry.geom\_components.nacelle\_pylons.comput  
 mass\_breakdown.update\_mlw\_and\_mzf  
 method), 89  
 sfc\_at\_max\_thrust() (fas- WingAreaConstraintsAero (class in fas-  
 toad\_cs25.models.propulsion.fuel\_propulsion.rubber\_engine.rubber\_engine.RubberEngine  
 mass\_breakdown.update\_mlw\_and\_mzf  
 method), 57  
 sfc\_ratio() (fastoad\_cs25.models.propulsion.fuel\_propulsion.rubber\_engine.rubber\_engine.RubberEngine fas-  
 method), 57  
 sweep\_angle\_25 (fas- WingAreaConstraintsGeom (class in fas-  
 toad\_cs25.models.aerodynamics.components.utils.cd0\_lifting\_surface.LiftingSurfaceGeometry  
 attribute), 7  
 SystemsWeight (class in fas- x (fastoad\_cs25.models.geometry.geom\_components.nacelle\_pylons.comput  
 toad\_cs25.models.weight.mass\_breakdown.c\_systems.sum), attribute), 29  
 81  
 T x (fastoad\_cs25.models.geometry.profiles.profile.Coordinates2D  
 attribute), 43  
 TAKEOFF (fastoad\_cs25.models.aerodynamics.constants.PolarType toad\_cs25.models.aerodynamics.external.xfoil.xfoil\_polar),  
 attribute), 23  
 thickness\_ratio (fas- XfoilPolar (class in fas-  
 toad\_cs25.models.aerodynamics.components.utils.cd0\_lifting\_surface.LiftingSurfaceGeometry  
 attribute), 6  
 thickness\_ratio (fas- y (fastoad\_cs25.models.geometry.geom\_components.nacelle\_pylons.comput  
 toad\_cs25.models.geometry.profiles.profile.Profile y (fastoad\_cs25.models.geometry.profiles.profile.Coordinates2D  
 property), 43  
 ToiletsWeight (class in fas-  
 toad\_cs25.models.weight.mass\_breakdown.d\_furniture.d5\_toilets\_weight),