Component Drag Buildup

Aerodynamic methods (`Drag.aero`)

`import math`

Math module required for:

- `exp()`
- `log10()`
- `sqrt()`

`aero.reynolds(rho, v, l ,mu)`

Calculates the Reynolds number

Parameters

- `rho` : float - air density (slug/ft^3)
- `v` : float - freestream velocity (ft/s)
- `l` : float - characteristic length (ft)
- `mu` : float - dynamic viscosity (lbf s/ft^2)

Returns

- `out` : float - Reynolds number (dimensionless)

`aero.cf_lam(re)`

Calculates the skin friction coefficient for laminar flow

Parameters

- `re` : float - Reynolds number (dimensionless)

Returns

- `out` : float - laminar skin friction coefficient

`aero.cf_turb(re, m)`

Calculates the skin friction coefficient for turbulent flow

Parameters

- `re` : float - Reynolds number (dimensionless)
- `m` : float - Mach number (dimensionless)

Returns

- `out` : float - turbulent skin friction coefficient

`aero.atm_temp(h)`

Calculates the temperature of standard atmosphere at altitude

Parameters

- `h` : float - altitude (ft)

Returns

- `out` : float - temperature (F)

`aero.atm_pressure(h)`

Calculates the pressure of standard atmosphere at altitude

Parameters

- `h` : float - altitude (ft)

Returns

- `out` : float - pressure (psf)

`aero.atm_dynamic_viscosity(h)`

Calculates the dynamic viscosity of standard atmosphere at altitude

Parameters

- `h` : float - altitude (ft)

Returns

- `out` : float - dynamic viscosity (lbf s/ft^2)

`aero.a(h)`

Calculates the speed of sound in dry air at the temperature of standard atmosphere at altitude

Parameters

- `h` : float - altitude (ft)

Returns

- `out` : float - speed of sound (fps)

`aero.mach(v, h)`

Calculates the mach number of a given flow at altitude

Parameters

- `v` : float - freestream velocity (fps)
- `h` : float - altitude (ft)

Returns

- `out` : float - Mach number (dimensionless)

`aero.dynamic_pressure(v, h)`

Calculates the dynamic pressure of a given flow at altitude

Parameters

- `v` : float - freestream velocity (fps)
- `h` : float - altitude (ft)

Returns

- `out` : float - dynamic pressure (psf)

Input Module (`Drag.drag_in`)

```
from Drag.Aircraft import Aircraft
from Drag.Component import Component
```

`from Drag.Diverter import Diverter`

`from Drag.Flap import Flap`

`from Drag.Foil import Foil`

`from Drag.Fuselage import Fuselage`

`from Drag.Spoiler import Spoiler`

`from Drag.Store import Spoiler`

Import the constructors from each of the classes in the `Drag` package.

`define_diverter(v, h, data)`

Defines a diverter component given the flight conditions and the specs.

Parameters

- `V` : float - freestream velocity (fps)
- `h` : float - altitude (ft)
- `data` : list - list of data with specs for the component

Returns

- `out` : Diverter - diverter object created using the parameters

---

`define_flap(aircraft, data)`

Defines a flap component given the aircraft and the specs.

Parameters

- `aircraft` : Aircraft - aircraft the flap is on (fps)
- `data` : list - list of data with specs for the component

Returns

- `out` : Flap - flap object created using the parameters

`define_foil(v, h, data)`

Defines a foil component given the flight conditions and the specs.

Parameters

- `V` : float - freestream velocity (fps)
- `h` : float - altitude (ft)
- `data` : list - list of data with specs for the component

Returns

- `out` : Foil - foil object created using the parameters

`define_fuselage(v, h, data)`

Defines a fuselage component given the flight conditions and the specs.

Parameters

- `V` : float - freestream velocity (fps)
- `h` : float - altitude (ft)
- `data` : list - list of data with specs for the component

Returns

- `out` : Fuselage - fuselage object created using the parameters

---

`define_spoiler(aircraft, data)`

Defines a spoiler component given the aircraft and the specs.

Parameters

- `aircraft` : Aircraft - aircraft the spoiler is on (fps)
- `data` : list - list of data with specs for the component

Returns

- `out` : Spoiler - spoiler object created using the parameters

---

`define_store(v, h, data)`

Defines a store component given the flight conditions and the specs.

Parameters

- `V` : float - freestream velocity (fps)
- `h` : float - altitude (ft)
- `data` : list - list of data with specs for the component

Returns

- `out` : Store - store object created using the parameters

`read_aircraft_file()`

Obtains user input for the input file location. The input file is a csv file (format specified in example.csv). Reads the flight conditions from the csv file (freestream, altitude, reference area). Creates an aircraft object. Reads the data for each component, creates components, and adds components to the aircraft object.

Returns

- `out` : Aircraft - aircraft with all components specified in file

---

Output Module (`Drag.drag_out`)

`from Drag.Aircraft import Aircraft`

`from Drag.Component import Component`

`from Drag.Diverter import Diverter`

`from Drag.Flap import Flap`

`from Drag.Foil import Foil`

`from Drag.Fuselage import Fuselage`

`from Drag.Spoiler import Spoiler`

`from Drag.Store import Spoiler`

Import the constructors from each of the classes in the `Drag` package.

`write_output(aircraft)`

Creates a new csv file in the directory `./drag_results/` with the name of the aircraft and the flight conditions. Writes the drag data for each component of the aircraft. Writes the total parasite drag for the aircraft.

Parameters

- `out` : Aircraft - aircraft to write output for

Aircraft Class Definition (`Drag.Aircraft`)

`from . import aero`

aero module required for:

- `aero.mach`

`class Aircraft`

Definition of the aircraft class. An aircraft object contains information regarding flight conditions and aircraft geometry. The geometry is defined by component objects. An aircraft object can calculate its parasite drag.

`Aircraft.Aircraft.__init__(self, name, V, h, S_ref)`

Constructor for the aircraft class. Defines fields for the aircraft object: name, freestream velocity, altitude, reference area. Creates an empty dictionary for aircraft components.

Parameters

- `name`: string - aircraft name (e.g. B-2)
- `V`: float - freestream velocity (fps)
- `h`: float - altitude (ft)
- `S_ref`: float - reference area (sq. ft)

`Aircraft.Aircraft.add_component(self, name, component)`

Method adds specified component to the dictionary of aircraft components. The dictionary key is a string with the name of the component.

Parameters

- `name`: string - component name (e.g. wing)
- `component`: Component - component being added to list

`Aircraft.Aircraft.total_Cd0(self)`

Method calculates the total parasite drag coefficient by iterating through the components and calling the `get_Cd0` component method.

Returns

- `out` : float - total parasite drag coefficient (dimensionless)

`Aircraft.Aircraft.list_contributions(self)`

Method returns a dictionary of the contributions to parasite drag of each of the components.

Returns

- `out` : dictionary - parasite drag contributions for each component of the aircraft

`Aircraft.Aircraft.list_percent(self)`

Method returns a dictionary of the percent contributions to parasite drag of each of the components.

Returns

- `out` : dictionary - parasite drag percent contributions for each component of the aircraft

---

Component Class Definition (`Drag.Component`)

`from . import aero`

aero module required for:

- `aero.atm_density`
- `aero.atm_dynamic_viscosity`
- `aero.reynolds`
- `aero.mach`
- `aero.cf_lam`
- `aero.cf_turb`

`class Component`

Definition of the Component class. This is a generic component, which is inherited by more specific classes. A component object contains information regarding flight conditions and component geometry. A component object can calculate aerodynamic quantities including parasite drag.

`Component.Component.__init__(self, V, h, L_c, S_wet, percent_lam)`

Constructor for the component class. Defines fields for the freestream velocity, the altitude, the characteristic length, the wetter area, and the percent laminar flow over the component. Assigns the value of `self.Q` to `1.0`. Calculates the values of density and viscosity using the defined fields.

Parameters

- `V`: float - freestream velocity (fps)
- `h`: float - altitude (ft)
- `L_c`: characteristic length (ft)

- `S_wet` : wetted area (sq. ft)
- `percent_lam` : percent laminar flow (%)

`Component.Component.get_reynolds(self)`

Method calculates the reynolds number for the component using the `aero.reynolds` method and the applicable fields.

Returns

- `out` : float - reynolds number (dimensionless)

`Component.Component.get_mach(self)`

Method calculates the mach number for the component using the `aero.mach` method and the applicable fileds.

Returns

- `out` : float - mach number (dimensionless)

`Component.Component.get_cf(self)`

Method calculates the turbulent and laminar skin friction coefficient. Using the value fo `self.percent_lam`, the combined skin friction coefficient is calculated.

Returns

- `out` : float - skin friction coefficient (dimensionless)

`Component.Component.get_Cd0(self, S_ref)`

Method calculates the contribution of the component to the overall parasite drag using applicable methods.

Parameters

- `S_ref` : float - reference area (sq. ft)

Returns

- `out` : float - parasite drag contribution (dimensionless)

Diverter Class Definition (`Drag.Diverter`)

`from Drag.Component import Component`

Component class required to define Diverter class.

`class Diverter(Component)`

Definition of the Diverter class, which inherits the Component class. A Diverter object contains information to define a component and information about the geometry of the diverter. A Diverter object can perform component methods and calculate its form factor.

`Diverter.Diverter.__init__(self, V, h, L_c, S_wet, percent_lam, d, config)`

Constructor for the Diverter class. Defines the fields for a component object and the width and configuration of the diverter. It defines the interference factor for a diverter.

Parameters

- `d` : float - diverter width (ft)

- `config` : string - configuration of the diverter (single or double)

`Diverter.Diverter.get_FF(self)`

Method calculates the form factor of a diverter using the characteristic length, width, and configuration.

Returns

- `out` : float - form factor (dimensionless)

Flap Class Definition (`Drag.Flap`)

`from Drag.Component import Component`

Component class required to define Diverter class.

`class Flap(Component)`

Definition of the Flap class, which inherits the Component class. A Flap object contains information about the geometry and deflection of the flap as well as the wing the flap is connected to. A Flap object can calculate its parasite drag contribution.

`Flap.Flap.__init__(self, span, deflection, S_wet, wing)`

Constructor for the Flap class (which overrides the component constructor). Defines fields for the flap span, flap deflection, flap wetted area, and wing.

Parameters

- `span` : float - flap span (ft)

- `deflection` : float - flap deflection (degrees)
- `S_wet` : float - flap wetted area (sq. ft)
- `wing` : Foil - wing the flap is on

`Flap.Flap.get_Cd0(self.S_ref)`

Method calculates the parasite drag contribution of the flap. Uses equation found in Raymer's text, which is a rough approximation of the drag.

Parameters

- `S_ref` : float - reference area (sq. ft)

Returns

- `out` : float - parasite drag contribution (dimensionless)

Foil Class Definition (`Drag.Foil`)

`from . import aero`

aero module required for:

- aero.mach

`from Drag.Component import Component`

Component class required to define Foil class.

`class Foil(Component)`

Definition of the Foil class, which inherits the Component class. This class is used to define wings and stabilizers. A Foil object contains information to define a component and information regarding the geometry and configuration of the airfoil. A foil object can perform the necessary calculations to calculate its contribution to parasite drag.

`Foil.Foil.__init__(self, V, h, L_c, S_wet, percent_lam, t_c, x_c, sweep, span, config)`

Constructor for the foil class. Defines the fields for a component object and the geometry and configuration of the airfoil. It defines the interference factor using the configuration.

Parameters

- `t_c` : float - thickness/chord (dimensionless)
- `x_c` : float - chordwise location of maximum thickness (dimensionless)
- `sweep` : float - wing sweep (radian)
- `span` : float - span of the airfoil (feet)
- `config` : string - configuration of the wing

`Foil.Foil.define_Q(self)`

Method finds the value of the interference factor of the airfoil based on the configuration. Called in the constructor.

Returns

- `out` : float - interference factor (dimensionless)

`Foil.Foil.get_FF(self)`

Method calculates the form factor of an airfoil using the geometry of the airfoil. Uses the `aero.mach` method.

Returns

- `out` : float - form factor (dimensionless)

Fuselage Class Definition (`Drag.Fuselage`)

`import math`

Math module required for:

- `math.sqrt()`
- `math.pi`

`from Drag.Component import Component`

Component class required to define Fuselage class.

`class Fuselage(Component)`

Definition of the Fuselage class, which inherits the Component class. A Fuselage object contains information to define a component and information regarding the geometry and configuration of the fueslage. A fueslage object can perform necessary calculations to calculate the contribution to parasite drag.

`Fuselage.Fuselage.__init__(self, V, h, L_c, S_wet,`

`percent_lam, A_max, u, config)`

Constructor for the fuselage class. Defines the fields for a component object and the geometry and configuration of the fuselage. It defines the interference factor as 1.0.

Parameters

- `A_max` : float - maximum cross-sectional area (sq. ft)
- `u` : float - tail upsweep (radian)
- `config` : string - fuselage configuration

`Fuselage.Fuselage.get_f(self)`

Method calculates the value of `f` .

Returns

- `out` : float - f (dimensionless)

`Fuselage.Fuselage.get_FF(self)`

Method calculates the form factor of the fueslage. Adjusts the value of the form factor based on the configuration.

Returns

- `out` : float - form factor (dimensionless)

`Fuselage.Fuselage.get_Cd0(self, S_ref)`

Method calculates the contribution to parasite drag. Method takes the drag from the tail upsweep into account.

Returns

- `out` : float - parasite drag contribution (dimensionless)

Spoiler Class Definition (`Drag.Spoiler`)

`from . import aero`

aero module required for

- `aero.mach`

`from Drag.Component import Component`

Component class required to define Spoiler class.

`class Spoiler(Component)`

Definition of the Spoiler class, which inherits from the Component class. A Spoiler object contains information regarding the geometry of the spoiler and the wing it is on. A spoiler object can calculate its parasite drag contribution.

`Spoiler.Spoiler.__init__(self, A_base, wing)`

Constructor for the spoiler class. Defines the fields for the area and wing the spoiler is on.

Parameters

- `A_base` : float - area exposed to flow (sq. ft)
- `wing` : Wing - wing spoiler is on

`Spoiler.Spoiler.get_Cd0(self, S_ref)`

Method calculates the parasite drag contribution. Using the equation found in Raymer's text.

Returns

- `out` : float - parasite drag contribution (dimensionless)

Store Class Definition (`Drag.Spoiler`)

`import math`

Math module required for:

- `math.sqrt()`
- `math.pi`

`from Drag.Component import Component`

Component class required to define Store class.

`class Store(Component)`

Definition of the Store class, which inherits from the Component class. A Store object can define a necelle, an attack store, or any other body on the exterior of an aircraft. A Store object contains information to define a component and information regarding the geometry of the store. A store object can perform necessary calculations to calculate the contribution to parasite drag.

`Store.Store.__init__(self, V, h, L_c, S_wet, percent_lam, A_max, config)`

Constructor for the store class. Define the fields for a component object and the maximum cross-sectionl area. The interference factor is defined based on the configuration.

Parameters

- `A_max` : float - maximum cross-sectional area (sq. ft)
- `config` : string - configuration of the store

`Store.Store.define_Q(self)`

Method finds the value of the interference factor of the store based on the configuration. Called in the constructor.

Returns

- `out` : float - interference factor (dimensionless)

`Store.Store.get_f(self)`

Method calculates the value of `f`.

Returns

- `out` : float - f (dimensionless)

`Store.Store.get_FF(self)`

Method calculates the form factor of the store.

Returns

- `out` : float - form factor (dimensionless)