Generic Aircraft Simulation

Generated by Doxygen 1.8.13

Contents

Chapter 1

GenericAircraftSimulation

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

Module Documentation

5.1 Aerodynamic

Classes

- class Aerodynamics
- class BaseAerodynamic
- class DATCOMAerodymamic

Functions

- Aerodynamics::Aerodynamics ()
  constructor
- Aerodynamics::∼Aerodynamics ()
  destructor
- void Aerodynamics::selectAerodynamicType (int type)
  set pointer to desired class
- void Aerodynamics::initAerodynamic ()
  initialize aerodynamic paramters
- void Aerodynamics::updateAerodynamic (Float64 FlightTime, AtmosphereStruct &AtmoData, Aerodynamic↩ Struct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData)
  calculate aero forces and moments

5.1.1 Detailed Description

Author

Jan Olucak

Date

25.11.2017 1.0

Aerodynamic class is used to call the desired aerodynamic model. The engine model is selected from General.dat input file.

5.1.2 Class Documentation

5.1.2.1 class Aerodynamics

Definition at line 23 of file Aerodynamic.h.

Public Member Functions

- Aerodynamics ()
  constructor
- ~Aerodynamics ()
  destructor
- void selectAerodynamicType (int type)
  set pointer to desired class
- void initAerodynamic ()
  initialize aerodynamic paramters
- void updateAerodynamic (Float64 FlightTime, AtmosphereStruct &AtmoData, AerodynamicStruct &Aero↩ Data, AirframeStruct &AirframeData, ThrustStruct &ThrustData)
  calculate aero forces and moments

5.1.2.2 class BaseAerodynamic

Author

Jan Olucak

Date

25.11.2017

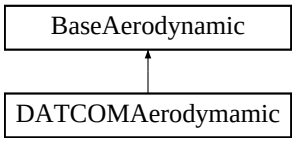
Version

1.0

Base Aerodynamic class is the superclass for all aerodynamic models. Using pointer to base init and update function allows the user to extend the aerodynamic module with new models.

Definition at line 22 of file BaseAerodynamic.h.

Inheritance diagram for BaseAerodynamic:



Public Member Functions

- BaseAerodynamic ()
  constructor
- ~BaseAerodynamic ()
  destructor
- void updateAerodynamic (Float64 FlightTime, AtmosphereStruct &AtmoData, AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData)
  The update function from the selected aerodynamic model is called by a pointer.
- void initAerodynamic ()
  The init function from the selected aerodynamic model is called by a pointer.
- virtual void **calcAerodynamic** (Float64 FlightTime, AtmosphereStruct &AtmoData, AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData)
- virtual void **initializeAerodynamic** ()

5.1.2.3 class DATCOMAerodymamic

Author

Jan Olucak

Date

25.11.2017

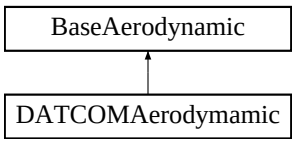
Version

1.0

DATCOm aerodynamic class is a child class from BaseAerodynamic. This class calculates aerodynamic forces and moments with tables from DATCOM. Tables of derivative are read in from specific file.

Definition at line 20 of file DATCOMAerodynamic.h.

Inheritance diagram for DATCOMAerodymamic:



Public Member Functions

- DATCOMAerodymamic ()
  constructor
- ~DATCOMAerodymamic ()
  destructor
- void initializeAerodynamic ()
  read in tables of derivatives
- void calcAerodynamic (Float64 FlightTime, AtmosphereStruct &AtmoData, AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData)
  current flight state is used to interpolated derivatives and a linear aerodynamic model calculates forces and moments

5.2 Airframe

Classes

- class Airframe

Functions

- Airframe::Airframe ()

  constructor
- Airframe::∼Airframe ()

  destructor
- void Airframe::initAirframe (AircraftStruct &AircraftData, AirframeStruct &AirframeData)

  Airframe initialization Airframe and Aircraft Data are initialized. Parameters from Aircraft.dat are read in and stored in their specific structure.
- void Airframe::updateTranslational (AerodynamicStruct &AeroData, ThrustStruct &ThrustData, AircraftStruct &AircraftData, AirframeStruct &AirframeData)

  translational equations of motion translational body accelerations are calculated
- void Airframe::updateRotational (AerodynamicStruct &AeroData, ThrustStruct &ThrustData, AircraftStruct &AircraftData, AirframeStruct &AirframeData)

  rotational equations of motion rotation body accelerations are calculated. Euler angle derivatives,too.

5.2.1 Detailed Description

Author

Jan Olucak

Date

27.11.2017

Version

1.0

Airframe class calculates body fixed acceleration

5.2.2 Class Documentation

5.2.2.1 class Airframe

Definition at line 18 of file Airframe.h.

Public Member Functions

- Airframe ()

  constructor
- ~Airframe ()

  destructor
- void initAirframe (AircraftStruct &AircraftData, AirframeStruct &AirframeData)

  Airframe initialization Airframe and Aircraft Data are initialized. Parameters from Aircraft.dat are read in and stored in their specific structure.
- void updateTranslational (AerodynamicStruct &AeroData, ThrustStruct &ThrustData, AircraftStruct &AircraftData, AirframeStruct &AirframeData)

  translational equations of motion translational body accelerations are calculated
- void updateRotational (AerodynamicStruct &AeroData, ThrustStruct &ThrustData, AircraftStruct &AircraftData, AirframeStruct &AirframeData)

  rotational equations of motion rotation body accelerations are calculated. Euler angle derivatives,too.

5.2.3 Function Documentation

5.2.3.1 updateRotational()

```
void Airframe::updateRotational (
            AerodynamicStruct & AeroData,
            ThrustStruct & ThrustData,
            AircraftStruct & AircraftData,
            AirframeStruct & AirframeData )
```

rotational equations of motion rotation body accelerations are calculated. Euler angle derivatives,too.

Parameters

AerodynamicStruct	Aerodynamic moments and angles
ThrustStruct	Thrust forces and moments
AircraftStruct	aircraft mass

Returns

Data stored in AirframeStruct

Definition at line 58 of file Airframe.cpp.

5.2.3.2 updateTranslational()

```
void Airframe::updateTranslational (
            AerodynamicStruct & AeroData,
            ThrustStruct & ThrustData,
```

```
            AircraftStruct & AircraftData,
            AirframeStruct & AirframeData )
```

translational equations of motion translational body accelerations are calculated

Parameters

AerodynamicStruct	Aerodynamic forces,moments and angles
ThrustStruct	Thrust forces and moments
AircraftStruct	aircraft mass

Returns

Data stored in AirframeStruct

Definition at line 38 of file Airframe.cpp.

5.3 Atmosphere

Classes

- class Atmopshere

Typedefs

- typedef double **Float64**

Functions

- Atmopshere::Atmopshere ()
  constructor
- Atmopshere::∼Atmopshere ()
  destructor
- void Atmopshere::initAtmosphere ()
  initialize atmospheric paramters
- void Atmopshere::updateAtmosphere (Float64 &Altitude, AtmosphereStruct &AtmoData)
  calculates atmospheric data depending on altitude

5.3.1 Detailed Description

Author

Jan Olucak

Date

25.11.2017 1.0

DataCloud is a global data storage for structures. It serves the purpose to provide data for several applications like the simulation itself, module and unit tests.

5.3.2 Class Documentation

5.3.2.1 class Atmopshere

Definition at line 21 of file Atmosphere.h.

Public Member Functions

- Atmopshere ()
  constructor
- ∼Atmopshere ()
  destructor
- void initAtmosphere ()
  initialize atmospheric paramters
- void updateAtmosphere (Float64 &Altitude, AtmosphereStruct &AtmoData)
  calculates atmospheric data depending on altitude

5.3.3 Function Documentation

5.3.3.1 updateAtmosphere()

```
void Atmopshere::updateAtmosphere (
            Float64 & Altitude,
            AtmosphereStruct & AtmoData )
```

calculates atmospheric data depending on altitude

Parameters

Altitude	current altitude

Returns

AtmosphericStruc store air density, speed of sound, temperature, pressure

troposphere

lower stratosphere

upper stratosphere–> Altitude >= 25000.0

Definition at line 19 of file Atmosphere.cpp.

5.4 DataCloud

Author

Jan Olucak

Date

25.11.2017

Version

1.0

DataCloud is a global data storage for structures. It serves the purpose to provide data for several applications like the simulation itself, module and unit tests.

5.5 Engine

Classes

- class BaseThrust
- class Engine
- class ThrustAnalytical

5.5.1 Detailed Description

Author

Jan Olucak

Date

25.11.2017 1.0

Engine class is used to call the desired engine model. The engine model is selected from General.dat input file.

5.5.2 Class Documentation

5.5.2.1 class BaseThrust

Author

Jan Olucak

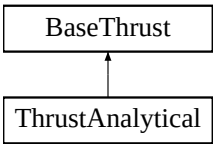
Date

25.11.2017 1.0

Base Thrust class is the superclass for all engine models. Using pointer to base init and update function allows the user to extend the engine module with new engine models.

Definition at line 22 of file BaseThrust.h.

Inheritance diagram for BaseThrust:



Public Member Functions

- BaseThrust ()
  constructor
- ~BaseThrust ()
  destructor
- void updateThrust (Float64 FlightTime, AtmosphereStruct &AtmoData, AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData)
- void initThrust ()
- virtual void **calcThrust** (Float64 FlightTime, AtmosphereStruct &AtmoData, AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData)
- virtual void **initializeThrust** ()

5.5.2.1.1 Member Function Documentation

5.5.2.1.1.1 initThrust()

```
void BaseThrust::initThrust ( )
```

The init function from the selected engine is called by a pointer.

Definition at line 11 of file BaseThrust.cpp.

5.5.2.1.1.2 updateThrust()

```
void BaseThrust::updateThrust (
            Float64 FlightTime,
            AtmosphereStruct & AtmoData,
            AerodynamicStruct & AeroData,
            AirframeStruct & AirframeData,
            ThrustStruct & ThrustData )
```

The update function from the selected engine is called by a pointer.

Definition at line 31 of file BaseThrust.cpp.

5.5.2.2 class Engine

Definition at line 18 of file Engine.h.

Public Member Functions

- Engine ()
  constructor
- ~Engine ()
  destructor
- void selectEngineType (int type)
  select Engine Type depending on input file
- void initEngine ()
  initilization of engine specific data
- void updateEngine (Float64 FlightTime, AtmosphereStruct &AtmoData, AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData)
  calculate thrust forces and moments

5.5.2.2.1 Member Function Documentation

5.5.2.2.1.1 selectEngineType()

```
void Engine::selectEngineType (
            int type )
```

select Engine Type depending on input file

Parameters

type	specific integer to select desired engine

Definition at line 15 of file Engine.cpp.

5.5.2.2.1.2 updateEngine()

```
void Engine::updateEngine (
            Float64 FlightTime,
            AtmosphereStruct & AtmoData,
            AerodynamicStruct & AeroData,
            AirframeStruct & AirframeData,
            ThrustStruct & ThrustData )
```

calculate thrust forces and moments

Parameters

AtmosphereStruct	get current atmospheric data
AerodynamiStruct	get mach number
AirframeStruct	get current throttle stick position

Returns

ThrustStruct current thrust data is stored

Definition at line 34 of file Engine.cpp.

5.5.2.3 class ThrustAnalytical

Author

Jan Olucak

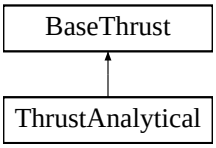
Date

25.11.2017 1.0

Base Thrust class is the superclass for all engine models. Using pointer to base init and update function allows the user to extend the engine module with new engine models.

Definition at line 20 of file ThrustAnalytical.h.

Inheritance diagram for ThrustAnalytical:



Public Member Functions

- ThrustAnalytical ()
  constructor
- ~ThrustAnalytical ()
  destructor
- void initializeThrust ()
  read in Data from Engine.dat
- void calcThrust (Float64 FlightTime, AtmosphereStruct &AtmoData, AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData)
  calculate thrust forces and moments

5.5.2.3.1 Constructor & Destructor Documentation

5.5.2.3.1.1 ~ThrustAnalytical()

```
ThrustAnalytical::~ThrustAnalytical ( )
```

destructor

destrcutor

Definition at line 10 of file ThrustAnalytical.cpp.

5.5.2.3.2 Member Function Documentation

5.5.2.3.2.1 calcThrust()

```
void ThrustAnalytical::calcThrust (
            Float64 FlightTime,
            AtmosphereStruct & AtmoData,
            AerodynamicStruct & AeroData,
            AirframeStruct & AirframeData,
            ThrustStruct & ThrustData )  [virtual]
```

calculate thrust forces and moments

calculation of thrust forces and moments

Parameters

AtmosphereStruct	get current atmospheric data
AerodynamiStruct	get mach number
AirframeStruct	get current throttle stick position

Returns

ThrustStruct current thrust data is stored

Reimplemented from BaseThrust.

Definition at line 30 of file ThrustAnalytical.cpp.

5.5.2.3.2.2 initializeThrust()

```
void ThrustAnalytical::initializeThrust ( )  [virtual]
```

read in Data from Engine.dat

data is read in from Engine.dat and stored in private variables

Reimplemented from BaseThrust.

Definition at line 15 of file ThrustAnalytical.cpp.

Chapter 6

Class Documentation

6.1 DataLogger Class Reference

Public Member Functions

- **DataLogger** (std::string aPath, int aWidth, std::string aDelimiter)
- void **add** (std::string aHeader, double &aVar)
- void **add** (std::string aHeader, int &aVar)
- void **print** ()
- void **printHeader** ()

6.1.1 Detailed Description

Definition at line 9 of file DataLogger.h.

The documentation for this class was generated from the following files:

- DataLogger.h
- DataLogger.cpp

6.2 LinearInterpolation Class Reference

Public Member Functions

- VectorXd **loadTable** (MatrixXd)
- Float64 **searchIndex** (VectorXd Vector, Float64 Value)
- Float64 **biLinearInterpolation** (VectorXd Vector1, VectorXd Vector2, MatrixXd Table, Float64 Value1, Float64 Value2)

6.2.1 Detailed Description

Definition at line 18 of file LinearInterpolation.h.

The documentation for this class was generated from the following files:

- LinearInterpolation.h
- LinearInterpolation.cpp

6.3 readInData Class Reference

Public Member Functions

- Float64 **readInParameter** (std::string CodeWord, std::string Filename)
- MatrixXd **readInTable** (std::string FileName)
- VectorXd **readInVector** (std::string FileName)
- void **setPath** (std::string Pathname)

6.3.1 Detailed Description

Definition at line 37 of file readInData.h.

The documentation for this class was generated from the following files:

- readInData.h
- readInData.cpp