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

5.1.1 Detailed Description

Author

Jan Olucak

Date

25.11.2017

Version

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 (AerodynamicStruct &AeroData, AircraftStruct &AircraftDatas)
  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

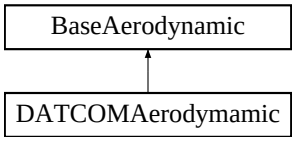
Version

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 &Aero↩ Data, AirframeStruct &AirframeData, ThrustStruct &ThrustData)
  The update function from the selected aerodynamic model is called by a pointer.
- void initAerodynamic (AerodynamicStruct &AeroData, AircraftStruct &AircraftData)
  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** (AerodynamicStruct &AeroData, AircraftStruct &AircraftData)

5.1.2.3 class DATCOMAerodymamic

Author

Jan Olucak

Date

25.11.2017

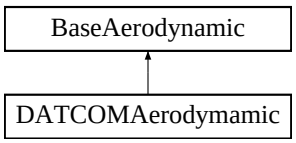
Version

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 21 of file DATCOMAerodynamic.h.

Inheritance diagram for DATCOMAerodymamic:



Public Member Functions

- DATCOMAerodymamic ()
  constructor
- ∼DATCOMAerodymamic ()
  destructor
- void initializeAerodynamic (AerodynamicStruct &AeroData, AircraftStruct &AircraftData)
  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

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 &Aircraft↩ Data, AirframeStruct &AirframeData)
  rotational equations of motion rotation body accelerations are calculated. Euler angle derivatives,too.

5.2.2.1.1 Member Function Documentation

5.2.2.1.1.1 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

AeroData	Aerodynamic moments and angles
ThrustData	Thrust forces and moments
AircraftData	aircraft mass

Returns

Data stored in AirframeStruct

Definition at line 58 of file Airframe.cpp.

5.2.2.1.1.2 updateTranslational()

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

translational equations of motion translational body accelerations are calculated

Parameters

AeroData	Aerodynamic forces,moments and angles
ThrustData	Thrust forces and moments
AircraftData	aircraft mass

Returns

Data stored in AirframeStruct

Definition at line 38 of file Airframe.cpp.

5.3 Atmosphere

Classes

- class Atmopshere

5.3.1 Detailed Description

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.

5.3.2 Class Documentation

5.3.2.1 class Atmopshere

Definition at line 20 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

5.3.2.1.1 Member Function Documentation

5.3.2.1.1.1 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

Version

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

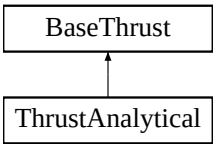
Version

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 (ThrustStruct &ThrustData, AircraftStruct &AircraftData)
- virtual void calcThrust (Float64 FlightTime, AtmosphereStruct &AtmoData, AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData)

  calculate thrust forces and moments
- virtual void **initializeThrust** (ThrustStruct &ThrustData, AircraftStruct &AircraftData)

5.5.2.1.1 Member Function Documentation

5.5.2.1.1.1 calcThrust()

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

calculate thrust forces and moments

Parameters

FlightTime	
AtmoData	get current atmospheric data
AeroData	get mach number
AirframeData	get current throttle stick position

Returns

current thrust data is stored in ThrustStruct

Reimplemented in ThrustAnalytical.

Definition at line 24 of file BaseThrust.cpp.

5.5.2.1.1.2 initThrust()

```
void BaseThrust::initThrust (
            ThrustStruct & ThrustData,
            AircraftStruct & AircraftData )
```

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

Definition at line 11 of file BaseThrust.cpp.

5.5.2.1.1.3 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 34 of file BaseThrust.cpp.

5.5.2.2 class Engine

Definition at line 19 of file Engine.h.

Public Member Functions

- Engine ()
  constructor
- ~Engine ()
  destructor
- void selectEngineType (int type)
  select Engine Type depending on input file
- void initEngine (ThrustStruct &ThrustData, AircraftStruct &AircraftData)
  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 13 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

FlightTime	
AtmoData	get current atmospheric data
AeroData	get mach number
AirframeData	get current throttle stick position

Returns

current thrust data is stored in ThrustStruct

Definition at line 35 of file Engine.cpp.

5.5.2.3 class ThrustAnalytical

Author

Jan Olucak

Date

25.11.2017

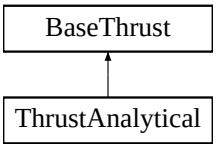
Version

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 (ThrustStruct &ThrustData, AircraftStruct &AircraftData)
  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

FlightTime	
AtmoData	get current atmospheric data
AeroData	get mach number
AirframeData	get current throttle stick position

Returns

current thrust data is stored in ThrustStruct

Reimplemented from BaseThrust.

Definition at line 35 of file ThrustAnalytical.cpp.

5.5.2.3.2.2 initializeThrust()

```
void ThrustAnalytical::initializeThrust (
            ThrustStruct & ThrustData,
            AircraftStruct & AircraftData )  [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.

5.6 Tools

Classes

- class DataLogger

- const Float64 GAMMA = 1.4
- const Float64 **GAS_CONSTANT** = 287
- const Float64 **RHO_0** = 1.225
- const Float64 **PI** = 3.14159265359
- const Float64 **GRAVITATIONAL_CONSTANT** = 9.80665

- typedef double Float64

5.6.1 Detailed Description

Author

Jan Olucak

Date

25.11.2017

Version

1.0

Data Logger is a class to write simulation data to an outputfile.

5.6.2 Class Documentation

5.6.2.1 class DataLogger

Definition at line 18 of file DataLogger.h.

Public Member Functions

- DataLogger (std::string aPath, int aWidth, std::string aDelimiter)
  constructor
- ~DataLogger ()
  Destructor.
- void add (std::string aHeader, double &aVar)
  defines variable which is stored in outputfile (double)
- void add (std::string aHeader, int &aVar)
  defines variable which is stored in outputfile (integer)
- void print ()
  writes defined variable to an outputfile
- void printHeader ()
  defines header of specific variable

5.6.3 Typedef Documentation

5.6.3.1 Float64

```
typedef double Float64
```

Author

Jan Olucak

Date

25.11.2017

Version

1.0

Provides independet data types

Definition at line 13 of file IndependetDataTypes.h.

5.6.4 Variable Documentation

5.6.4.1 GAMMA

```
const Float64 GAMMA = 1.4
```

Author

Jan Olucak

Date

25.11.2017

Version

1.0

Provides physical and mathematical constants

Definition at line 15 of file Constants.h.

5.7 Trajectory

Classes

- class BaseTrajectory
- class Trajectory
- class Trajectory3Dof

Functions

- void **Trajectory::selectTrajectory** (int type)
- void **Trajectory::updateTrajectory** (Float64 FlightTime, AtmosphereStruct &AtmoData, Aerodynamic↩ Struct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData, AutopilotStruct &AutopilotData, GuidanceStruct &GuidanceData)
- void **Trajectory::initTrajectory** (AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData, AircraftStruct &AircraftData, AutopilotStruct &AutopilotData, GuidanceStruct &GuidanceData)
- void **Trajectory3Dof::initializeTrajectory** (AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData, AircraftStruct &AircraftData, AutopilotStruct &AutopilotData, GuidanceStruct &GuidanceData)
- void **Trajectory3Dof::calcTrajectorythis** (Float64 FlightTime, AtmosphereStruct &AtmoData, Aerodynamic↩ Struct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData, AutopilotStruct &AutopilotData, GuidanceStruct &GuidanceData)

5.7.1 Detailed Description

Author

Jan Olucak

Date

28.11.2017

Version

1.0

BaseTrajectory is the superclass for all trajectory classes.

5.7.2 Class Documentation

5.7.2.1 class BaseTrajectory

Author

Jan Olucak

Date

28.11.2017

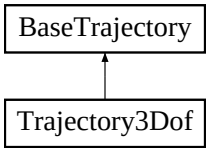
Version

1.0

BaseTrajectory is the superclass for all trajectory classes.

Definition at line 14 of file BaseTrajectory.h.

Inheritance diagram for BaseTrajectory:



Public Member Functions

- void **initTrajectory** (AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &Thrust↩ Data, AircraftStruct &AircraftData, AutopilotStruct &AutopilotData, GuidanceStruct &GuidanceData)
- void **updateTrajectory** (Float64 FlightTime, AtmosphereStruct &AtmoData, AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData, AutopilotStruct &AutopilotData, GuidanceStruct &GuidanceData)
- virtual void **initializeTrajectory** (AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData, AircraftStruct &AircraftData, AutopilotStruct &AutopilotData, GuidanceStruct &GuidanceData)
- virtual void **calcTrajectory** (Float64 FlightTime, AtmosphereStruct &AtmoData, AerodynamicStruct &Aero↩ Data, AirframeStruct &AirframeData, ThrustStruct &ThrustData, AutopilotStruct &AutopilotData, Guidance↩ Struct &GuidanceData)

5.7.2.2 class Trajectory

Definition at line 16 of file Trajectory.h.

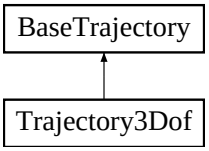
Public Member Functions

- void **selectTrajectory** (int type)
- void **updateTrajectory** (Float64 FlightTime, AtmosphereStruct &AtmoData, AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData, AutopilotStruct &AutopilotData, GuidanceStruct &GuidanceData)
- void **initTrajectory** (AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &Thrust↩ Data, AircraftStruct &AircraftData, AutopilotStruct &AutopilotData, GuidanceStruct &GuidanceData)

5.7.2.3 class Trajectory3Dof

Definition at line 19 of file Trajectory3DoF.h.

Inheritance diagram for Trajectory3Dof:



Public Member Functions

- void **initializeTrajectory** (AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData, AircraftStruct &AircraftData, AutopilotStruct &AutopilotData, GuidanceStruct &Guidance↩ Data)
- void **calcTrajectorythis** (Float64 FlightTime, AtmosphereStruct &AtmoData, AerodynamicStruct &AeroData, AirframeStruct &AirframeData, ThrustStruct &ThrustData, AutopilotStruct &AutopilotData, GuidanceStruct &GuidanceData)

Chapter 6

Class Documentation

6.1 LinearInterpolation Class Reference

Public Member Functions

- LinearInterpolation ()
  constructor
- ~LinearInterpolation ()
  destructor
- Float64 searchIndex (VectorXd Vector, Float64 Value)
  searches index of of a specific value in a vector/matrix
- Float64 linearInterpolation2D (VectorXd Vector1, VectorXd Vector2, MatrixXd Table, Float64 Value1, Float64 Value2)
  2D linear interpolation
- Float64 linearInterpolation1D (VectorXd Vector1, VectorXd Table, Float64 Value)
  1D linear interpolation

6.1.1 Detailed Description

Definition at line 28 of file LinearInterpolation.h.

6.1.2 Member Function Documentation

6.1.2.1 linearInterpolation1D()

```
Float64 LinearInterpolation::linearInterpolation1D (
            VectorXd Vector1,
            VectorXd Table,
            Float64 Value )
```

1D linear interpolation

Parameters

vector	that defines lines of a vector
Table	specific data vector
Value	wanted value

Definition at line 57 of file LinearInterpolation.cpp.

6.1.2.2 linearInterpolation2D()

```
Float64 LinearInterpolation::linearInterpolation2D (
            VectorXd Vector1,
            VectorXd Vector2,
            MatrixXd Table,
            Float64 Value1,
            Float64 Value2 )
```

2D linear interpolation

Parameters

Vector1	vector that defines lines of a table
Vector2	vector that defines columns of a table
Table	specific table
Value1	wanted value of line vector
Value2	wanted value of column vector

Definition at line 30 of file LinearInterpolation.cpp.

6.1.2.3 searchIndex()

```
Float64 LinearInterpolation::searchIndex (
            VectorXd Vector,
            Float64 Value )
```

searches index of of a specific value in a vector/matrix

Parameters

Vector	specific vector to search for index
Value	wanted value

Definition at line 11 of file LinearInterpolation.cpp.

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

- LinearInterpolation.h
- LinearInterpolation.cpp

6.2 readInData Class Reference

Public Member Functions

- readInData ()
  constructor
- ~readInData ()
  destructor
- Float64 readInParameter (std::string CodeWord, std::string Filename)
  read in a specific value from a file
- MatrixXd readInTable (std::string FileName)
  read in tables from input file
- VectorXd readInVector (std::string FileName)
  read in vector from input file
- void setPath (std::string Pathname)
  defines a relative path directory of input file

6.2.1 Detailed Description

Definition at line 28 of file readInData.h.

6.2.2 Member Function Documentation

6.2.2.1 readInParameter()

```
Float64 readInData::readInParameter (
            std::string CodeWord,
            std::string Filename )
```

read in a specific value from a file

Parameters

CodeWord	specific name of a parameter
Filename	Name of specific file

Definition at line 11 of file readInData.cpp.

6.2.2.2 readInTable()

```
MatrixXd readInData::readInTable (
            std::string FileName )
```

read in tables from input file

Parameters

Filename	Name of specific file

Definition at line 59 of file readInData.cpp.

6.2.2.3 readInVector()

```
VectorXd readInData::readInVector (
            std::string FileName )
```

read in vector from input file

Parameters

Filename	Name of specific file

Definition at line 125 of file readInData.cpp.

6.2.2.4 setPath()

```
void readInData::setPath (
            std::string Pathname )
```

defines a relative path directory of input file

Parameters

Pathname	relative path of data file directory

Definition at line 173 of file readInData.cpp.

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

- readInData.h
- readInData.cpp