UAV drop physic

Generated by Doxygen 1.9.1

1 Namespace Index 1
- 1.1 Namespace List 1

2 Hierarchical Index 3
- 2.1 Class Hierarchy 3

3 Class Index 5
- 3.1 Class List 5

4 File Index 7
- 4.1 File List 7

5 Namespace Documentation 9
- 5.1 def Namespace Reference 9
  - 5.1.1 Detailed Description 9
  - 5.1.2 Variable Documentation 9
    - 5.1.2.1 DEFAULT_AIR_DENSITY 9
    - 5.1.2.2 FRICTION_EPS 10
    - 5.1.2.3 GENTLY_PUSH 10
    - 5.1.2.4 GRAVITY_CONST 10

6 Class Documentation 11
- 6.1 AeroCoefficients Struct Reference 11
  - 6.1.1 Detailed Description 11
  - 6.1.2 Member Data Documentation 11
    - 6.1.2.1 C0 11
    - 6.1.2.2 Cab 12
    - 6.1.2.3 Cpqr 12
    - 6.1.2.4 d 12
    - 6.1.2.5 eAR 12
    - 6.1.2.6 S 12
    - 6.1.2.7 stallLimit 12
- 6.2 AHRSParams Struct Reference 12
  - 6.2.1 Detailed Description 13
  - 6.2.2 Member Data Documentation 13
    - 6.2.2.1 alpha 13
    - 6.2.2.2 Q 13
    - 6.2.2.3 R 13
    - 6.2.2.4 type 13
- 6.3 Ammo Class Reference 14
  - 6.3.1 Constructor & Destructor Documentation 14
    - 6.3.1.1 Ammo() [1/2] 14
    - 6.3.1.2 Ammo() [2/2] 14
  - 6.3.2 Member Function Documentation 14

6.3.2.1 getV0() 15
6.3.2.2 operator=() 15
6.3.3 Member Data Documentation 15
6.3.3.1 _V0 15
6.4 Cargo Class Reference 15
6.4.1 Constructor & Destructor Documentation 16
6.4.1.1 Cargo() [1/2] 16
6.4.1.2 Cargo() [2/2] 16
6.5 ControlSurfaces Class Reference 16
6.5.1 Detailed Description 16
6.5.2 Constructor & Destructor Documentation 17
6.5.2.1 ControlSurfaces() [1/2] 17
6.5.2.2 ControlSurfaces() [2/2] 17
6.5.3 Member Function Documentation 17
6.5.3.1 getCoefficients() 17
6.5.3.2 getNoOfSurface() 17
6.5.3.3 getValues() 18
6.5.3.4 restoreTrim() 18
6.5.3.5 setValues() 18
6.6 Drive Struct Reference 18
6.6.1 Detailed Description 18
6.6.2 Member Data Documentation 18
6.6.2.1 axis 19
6.6.2.2 hinges 19
6.6.2.3 noOfHinges 19
6.6.2.4 position 19
6.7 EKFScalers Struct Reference 19
6.7.1 Detailed Description 19
6.7.2 Member Data Documentation 20
6.7.2.1 baroScaler 20
6.7.2.2 predictScaler 20
6.7.2.3 updateScaler 20
6.7.2.4 zScaler 20
6.8 Hinge Class Reference 20
6.8.1 Detailed Description 21
6.8.2 Constructor & Destructor Documentation 21
6.8.2.1 Hinge() [1/3] 21
6.8.2.2 Hinge() [2/3] 21
6.8.2.3 Hinge() [3/3] 21
6.8.3 Member Function Documentation 21
6.8.3.1 getRot() 21
6.8.3.2 operator=() 22

6.8.3.3 updateValue() 22

6.9 Jet Class Reference 23

6.9.1 Detailed Description 23

6.9.2 Member Function Documentation 23

6.9.2.1 getLastThrust() 24

6.9.2.2 getThrust() 24

6.9.2.3 start() 24

6.9.3 Member Data Documentation 24

6.9.3.1 phases 25

6.9.3.2 thrust 25

6.9.3.3 time 25

6.10 Load Class Reference 25

6.10.1 Detailed Description 26

6.10.2 Constructor & Destructor Documentation 26

6.10.2.1 Load() [1/2] 26

6.10.2.2 Load() [2/2] 26

6.10.3 Member Function Documentation 26

6.10.3.1 getMass() 26

6.10.3.2 getOffset() 26

6.10.3.3 operator=() 27

6.10.3.4 release() 27

6.11 Logger Class Reference 27

6.11.1 Detailed Description 28

6.11.2 Constructor & Destructor Documentation 28

6.11.2.1 Logger() 28

6.11.2.2 ∼Logger() 28

6.11.3 Member Function Documentation 28

6.11.3.1 log() [1/2] 28

6.11.3.2 log() [2/2] 29

6.11.3.3 setFmt() 29

6.11.3.4 setLogDirectory() 29

6.12 ObjParams Class Reference 30

6.12.1 Detailed Description 30

6.12.2 Constructor & Destructor Documentation 30

6.12.2.1 ObjParams() [1/2] 30

6.12.2.2 ObjParams() [2/2] 31

6.12.3 Member Function Documentation 31

6.12.3.1 getForce() 31

6.12.3.2 getWind() 31

6.12.3.3 setForce() 31

6.12.3.4 setWind() 32

6.12.4 Member Data Documentation 32

6.12.4.1 CS_coff 32
6.12.4.2 id 32
6.12.4.3 mass 32
6.13 ODE Class Reference 33
6.13.1 Detailed Description 33
6.13.2 Member Enumeration Documentation 33
6.13.2.1 ODEMethod 33
6.13.3 Constructor & Destructor Documentation 34
6.13.3.1 ODE() 34
6.13.3.2 ∼ODE() 34
6.13.4 Member Function Documentation 34
6.13.4.1 factory() 34
6.13.4.2 fromString() 35
6.13.4.3 getMicrosteps() [1/2] 35
6.13.4.4 getMicrosteps() [2/2] 35
6.13.4.5 step() 36
6.14 ODE_Euler Class Reference 36
6.14.1 Detailed Description 36
6.14.2 Constructor & Destructor Documentation 37
6.14.2.1 ODE_Euler() 37
6.14.3 Member Function Documentation 37
6.14.3.1 step() 37
6.15 ODE_Heun Class Reference 37
6.15.1 Detailed Description 38
6.15.2 Constructor & Destructor Documentation 38
6.15.2.1 ODE_Heun() 38
6.15.3 Member Function Documentation 38
6.15.3.1 step() 38
6.16 ODE_RK4 Class Reference 39
6.16.1 Detailed Description 39
6.16.2 Constructor & Destructor Documentation 39
6.16.2.1 ODE_RK4() 39
6.16.3 Member Function Documentation 39
6.16.3.1 step() 39
6.17 ODETest Class Reference 40
6.17.1 Member Function Documentation 40
6.17.1.1 SetUp() 40
6.17.1.2 TearDown() 40
6.18 Params Class Reference 41
6.18.1 Detailed Description 41
6.18.2 Constructor & Destructor Documentation 41
6.18.2.1 Params() 41

6.18.2.2 ∼Params() 41
6.18.3 Member Function Documentation 42
6.18.3.1 getSingleton() 42
6.18.4 Member Data Documentation 42
6.18.4.1 ODE_METHOD 42
6.18.4.2 STEP_TIME 42
6.19 PID Class Reference 42
6.19.1 Detailed Description 43
6.19.2 Constructor & Destructor Documentation 43
6.19.2.1 PID() 43
6.19.2.2 ∼PID() 43
6.19.3 Member Function Documentation 43
6.19.3.1 calc() [1/2] 43
6.19.3.2 calc() [2/2] 44
6.19.3.3 clear() 44
6.19.3.4 set_dt() 44
6.20 Rotor Struct Reference 45
6.20.1 Detailed Description 45
6.20.2 Member Data Documentation 45
6.20.2.1 direction 45
6.20.2.2 forceCoff 45
6.20.2.3 hoverSpeed 46
6.20.2.4 maxSpeed 46
6.20.2.5 timeConstant 46
6.20.2.6 torqueCoff 46
6.21 SensorParams Struct Reference 46
6.21.1 Detailed Description 46
6.21.2 Member Data Documentation 47
6.21.2.1 bias 47
6.21.2.2 name 47
6.21.2.3 refreshTime 47
6.21.2.4 sd 47
6.22 Simulation Class Reference 47
6.22.1 Constructor & Destructor Documentation 48
6.22.1.1 Simulation() 48
6.22.1.2 ∼Simulation() 48
6.22.2 Member Function Documentation 49
6.22.2.1 addCommand() 49
6.22.2.2 addObj() 49
6.22.2.3 calcImpulseForce() 49
6.22.2.4 removeObj() 51
6.22.2.5 run() 51

6.22.2.6 solidSurfColision() 51
6.22.2.7 updateForce() 52
6.22.2.8 updateWind() 52
6.23 State Class Reference 52
6.23.1 Constructor & Destructor Documentation 53
6.23.1.1 State() 53
6.23.2 Member Function Documentation 53
6.23.2.1 addObj() 53
6.23.2.2 findIndex() 54
6.23.2.3 getNoObj() 54
6.23.2.4 getParams() 54
6.23.2.5 getPos() 55
6.23.2.6 getState() 55
6.23.2.7 getVel() 55
6.23.2.8 removeObj() 56
6.23.2.9 setVel() 56
6.23.2.10 to_string() 56
6.23.2.11 updateForce() 57
6.23.2.12 updateState() 57
6.23.2.13 updateWind() 57
6.23.3 Member Data Documentation 57
6.23.3.1 real_time 58
6.23.3.2 stateMutex 58
6.23.3.3 status 58
6.24 TimedLoop Class Reference 58
6.24.1 Detailed Description 58
6.24.2 Constructor & Destructor Documentation 58
6.24.2.1 TimedLoop() 58
6.24.3 Member Function Documentation 59
6.24.3.1 go() [1/2] 59
6.24.3.2 go() [2/2] 59
6.25 UAVparams Struct Reference 59
6.25.1 Detailed Description 60
6.25.2 Constructor & Destructor Documentation 61
6.25.2.1 UAVparams() 61
6.25.2.2 ∼UAVparams() 61
6.25.3 Member Function Documentation 61
6.25.3.1 getRotorHoverSpeeds() 61
6.25.3.2 getRotorMaxSpeeds() 61
6.25.3.3 getRotorTimeContants() 61
6.25.3.4 getSingleton() 61
6.25.3.5 loadConfig() 62

6.25.4 Member Data Documentation 62
6.25.4.1 aero_coffs 62
6.25.4.2 ahrs 62
6.25.4.3 ammo 62
6.25.4.4 cargo 62
6.25.4.5 ekf 62
6.25.4.6 initialMode 62
6.25.4.7 initialOrientation 63
6.25.4.8 initialPosition 63
6.25.4.9 initialVelocity 63
6.25.4.10 instantRun 63
6.25.4.11 Ix 63
6.25.4.12 Ixy 63
6.25.4.13 Ixz 63
6.25.4.14 Iy 63
6.25.4.15 Iyz 64
6.25.4.16 Iz 64
6.25.4.17 jets 64
6.25.4.18 m 64
6.25.4.19 name 64
6.25.4.20 noOfAmmo 64
6.25.4.21 noOfCargo 64
6.25.4.22 noOfJets 64
6.25.4.23 noOfRotors 65
6.25.4.24 pids 65
6.25.4.25 rotorMixer 65
6.25.4.26 rotors 65
6.25.4.27 sensors 65
6.25.4.28 surfaceMixer 65
6.25.4.29 surfaces 65

7 File Documentation **67**
7.1 lib/UAV_common/header/common.hpp File Reference 67
7.2 lib/UAV_common/src/components/aero_coefficients.hpp File Reference 67
7.3 lib/UAV_common/src/components/components.hpp File Reference 67
7.4 lib/UAV_common/src/components/control_surfaces.cpp File Reference 68
7.5 lib/UAV_common/src/components/control_surfaces.hpp File Reference 68
7.6 lib/UAV_common/src/components/drive.cpp File Reference 68
7.7 lib/UAV_common/src/components/drive.hpp File Reference 68
7.8 lib/UAV_common/src/components/hinge.cpp File Reference 68
7.8.1 Function Documentation 69
7.8.1.1 asSkewMatrix() 69

7.9 lib/UAV_common/src/components/hinge.hpp File Reference 69
7.10 lib/UAV_common/src/components/loads.cpp File Reference 69
7.11 lib/UAV_common/src/components/loads.hpp File Reference 69
7.12 lib/UAV_common/src/components/navi.hpp File Reference 70
7.13 lib/UAV_common/src/logger/logger.cpp File Reference 70
7.13.1 Function Documentation 70
7.13.1.1 shouldLog() 70
7.14 lib/UAV_common/src/logger/logger.hpp File Reference 70
7.14.1 Macro Definition Documentation 71
7.14.1.1 LOGGER_MASK 71
7.15 lib/UAV_common/src/ode/ode.cpp File Reference 71
7.16 lib/UAV_common/src/ode/ode.hpp File Reference 71
7.17 lib/UAV_common/src/ode/ode_impl.hpp File Reference 71
7.18 lib/UAV_common/src/ode/ode_test.cpp File Reference 72
7.18.1 Function Documentation 72
7.18.1.1 getMethodsToTest() 72
7.18.1.2 INSTANTIATE_TEST_SUITE_P() 72
7.18.1.3 main() 73
7.18.1.4 TEST_F() [1/2] 73
7.18.1.5 TEST_F() [2/2] 73
7.18.1.6 TEST_P() [1/3] 73
7.18.1.7 TEST_P() [2/3] 73
7.18.1.8 TEST_P() [3/3] 73
7.19 lib/UAV_common/src/parser/parser.cpp File Reference 74
7.19.1 Function Documentation 74
7.19.1.1 parseMatrixXd() 74
7.19.1.2 parseVectorXd() 74
7.20 lib/UAV_common/src/parser/parser.hpp File Reference 75
7.20.1 Function Documentation 75
7.20.1.1 parseMatrixXd() 75
7.20.1.2 parseVectorXd() 76
7.21 lib/UAV_common/src/parser/uav_params.cpp File Reference 76
7.21.1 Function Documentation 76
7.21.1.1 parseHinge() 76
7.21.1.2 parsePID() 77
7.22 lib/UAV_common/src/parser/uav_params.hpp File Reference 77
7.23 lib/UAV_common/src/PID/PID.cpp File Reference 77
7.24 lib/UAV_common/src/PID/PID.hpp File Reference 77
7.24.1 Enumeration Type Documentation 77
7.24.1.1 AntiWindUpMode 77
7.25 lib/UAV_common/src/timed_loop/status.hpp File Reference 78
7.25.1 Enumeration Type Documentation 78

7.25.1.1 Status 78
7.26 lib/UAV_common/src/timed_loop/timed_loop.cpp File Reference 78
7.27 lib/UAV_common/src/timed_loop/timed_loop.hpp File Reference 79
7.28 src/defines.hpp File Reference 79
7.29 src/main.cpp File Reference 79
7.29.1 Function Documentation 80
7.29.1.1 main() 80
7.29.1.2 parseArgs() 80
7.30 src/params.cpp File Reference 80
7.31 src/params.hpp File Reference 80
7.32 src/simulation.cpp File Reference 81
7.32.1 Function Documentation 81
7.32.1.1 isNormal() 81
7.33 src/simulation.hpp File Reference 81
7.34 src/state.cpp File Reference 82
7.35 src/state.hpp File Reference 82

Index **83**

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

def
Simulation constants 9

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

AeroCoefficients 11
AHRSParams 12
ControlSurfaces 16
Drive 18
    Jet 23
    Rotor 45
EKFScalers 19
Hinge 20
Load 25
    Ammo 14
    Cargo 15
Logger 27
ObjParams 30
ODE 33
    ODE_Euler 36
    ODE_Heun 37
    ODE_RK4 39
Params 41
PID 42
SensorParams 46
Simulation 47
State 52
testing::TestWithParam
    ODETest 40
TimedLoop 58
UAVparams 59

Chapter 3

Class Index

3.1 Class List

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

AeroCoefficients
Aerodynamic coefficient 11
AHRSParams
AHRS parameters 12
Ammo 14
Cargo 15
ControlSurfaces
Aircraft's control surfaces 16
Drive
Drive propelling aircraft 18
EKFScalers
Scalers for EKF 19
Hinge
Hinge connecting aircraft with drives 20
Jet
Jet rocket engine 23
Load
Load of aircraft that can be droped or launched 25
Logger
Log vector data with timestamp in file 27
ObjParams
Single obj parameters 30
ODE
Ordinal differencial equation solver 33
ODE_Euler
Explicit Euler algorithm 36
ODE_Heun
Second order explicit Heun algorithm 37
ODE_RK4
Fourth order Runge Kutta algorithm 39
ODETest 40
Params
Simulation parameters 41
PID
PID discrete controller 42

Rotor
Rotor engine with controlled speed 45
SensorParams
Base parameters of a sensor 46
Simulation 47
State 52
TimedLoop
Simulation of real-time synchronized loop 58
UAVparams
Parsed UAV configuration from XML 59

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

lib/UAV_common/header/common.hpp 67
lib/UAV_common/src/components/aero_coefficients.hpp 67
lib/UAV_common/src/components/components.hpp 67
lib/UAV_common/src/components/control_surfaces.cpp 68
lib/UAV_common/src/components/control_surfaces.hpp 68
lib/UAV_common/src/components/drive.cpp 68
lib/UAV_common/src/components/drive.hpp 68
lib/UAV_common/src/components/hinge.cpp 68
lib/UAV_common/src/components/hinge.hpp 69
lib/UAV_common/src/components/loads.cpp 69
lib/UAV_common/src/components/loads.hpp 69
lib/UAV_common/src/components/navi.hpp 70
lib/UAV_common/src/logger/logger.cpp 70
lib/UAV_common/src/logger/logger.hpp 70
lib/UAV_common/src/ode/ode.cpp 71
lib/UAV_common/src/ode/ode.hpp 71
lib/UAV_common/src/ode/ode_impl.hpp 71
lib/UAV_common/src/ode/ode_test.cpp 72
lib/UAV_common/src/parser/parser.cpp 74
lib/UAV_common/src/parser/parser.hpp 75
lib/UAV_common/src/parser/uav_params.cpp 76
lib/UAV_common/src/parser/uav_params.hpp 77
lib/UAV_common/src/PID/PID.cpp 77
lib/UAV_common/src/PID/PID.hpp 77
lib/UAV_common/src/timed_loop/status.hpp 78
lib/UAV_common/src/timed_loop/timed_loop.cpp 78
lib/UAV_common/src/timed_loop/timed_loop.hpp 79
src/defines.hpp 79
src/main.cpp 79
src/params.cpp 80
src/params.hpp 80
src/simulation.cpp 81
src/simulation.hpp 81
src/state.cpp 82
src/state.hpp 82

Chapter 5

Namespace Documentation

5.1 def Namespace Reference

Simulation constants.

Variables

- const double GRAVITY_CONST = 9.81

  Gravity constant on Earth in m/s2.
- const double FRICTION_EPS = 0.001

  minimal friction that is calculated (numerical float eps)
- const double GENTLY_PUSH = 0.15

  artificial force cofficient. Protect again diving objects in horizontal wall
- const double DEFAULT_AIR_DENSITY = 1.224

  Dry air density in normal conditions in kg/m3.

5.1.1 Detailed Description

Simulation constants.

5.1.2 Variable Documentation

5.1.2.1 DEFAULT_AIR_DENSITY

```
const double def::DEFAULT_AIR_DENSITY = 1.224
```

Dry air density in normal conditions in kg/m3.

5.1.2.2 FRICTION_EPS

```
const double def::FRICTION_EPS = 0.001
```

minimal friction that is calculated (numerical float eps)

5.1.2.3 GENTLY_PUSH

```
const double def::GENTLY_PUSH = 0.15
```

artificial force cofficient. Protect again diving objects in horizontal wall

5.1.2.4 GRAVITY_CONST

```
const double def::GRAVITY_CONST = 9.81
```

Gravity constant on Earth in m/s2.

Chapter 6

Class Documentation

6.1 AeroCoefficients Struct Reference

Aerodynamic coefficient.

```
#include <aero_coefficients.hpp>
```

Public Attributes

- double S
- double d
- double eAR
- Eigen::Vector< double, 6 > C0
- Eigen::Matrix< double, 6, 3 > Cpqr
- Eigen::Matrix< double, 6, 4 > Cab
- double stallLimit

6.1.1 Detailed Description

Aerodynamic coefficient.

6.1.2 Member Data Documentation

6.1.2.1 C0

```
Eigen::Vector<double,6> AeroCoefficients::C0
```

6.1.2.2 Cab

```
Eigen::Matrix<double,6,4> AeroCoefficients::Cab
```

6.1.2.3 Cpqr

```
Eigen::Matrix<double,6,3> AeroCoefficients::Cpqr
```

6.1.2.4 d

```
double AeroCoefficients::d
```

6.1.2.5 eAR

```
double AeroCoefficients::eAR
```

6.1.2.6 S

```
double AeroCoefficients::S
```

6.1.2.7 stallLimit

```
double AeroCoefficients::stallLimit
```

The documentation for this struct was generated from the following file:

- lib/UAV_common/src/components/aero_coefficients.hpp

6.2 AHRSParams Struct Reference

AHRS parameters.

```
#include <navi.hpp>
```

Public Attributes

- std::string type
- double alpha
- double Q
- double R

6.2.1 Detailed Description

AHRS parameters.

6.2.2 Member Data Documentation

6.2.2.1 alpha

```
double AHRSParams::alpha
```

6.2.2.2 Q

```
double AHRSParams::Q
```

6.2.2.3 R

```
double AHRSParams::R
```

6.2.2.4 type

```
std::string AHRSParams::type
```

The documentation for this struct was generated from the following file:

- lib/UAV_common/src/components/navi.hpp

6.3 Ammo Class Reference

```
#include <loads.hpp>
```

Inheritance diagram for Ammo:

Collaboration diagram for Ammo:

Public Member Functions

- Ammo ()=default
- Ammo (int ammount, double reload, Eigen::Vector3d offset, double mass, Eigen::Vector3d V0)
- Ammo & operator= (const Ammo &other)
- Eigen::Vector3d getV0 ()

  get start velocity of ammo when launched

Protected Attributes

- Eigen::Vector3d _V0

Additional Inherited Members

6.3.1 Constructor & Destructor Documentation

6.3.1.1 Ammo() [1/2]

```
Ammo::Ammo ( )  [default]
```

6.3.1.2 Ammo() [2/2]

```
Ammo::Ammo (
             int ammount,
             double reload,
             Eigen::Vector3d offset,
             double mass,
             Eigen::Vector3d V0 )
```

6.3.2 Member Function Documentation

6.3.2.1 getV0()

```
Eigen::Vector3d Ammo::getV0 ( )  [inline]
```

get start velocity of ammo when launched

Returns

start velocity vector

6.3.2.2 operator=()

```
Ammo & Ammo::operator= (
             const Ammo & other )
```

6.3.3 Member Data Documentation

6.3.3.1 _V0

```
Eigen::Vector3d Ammo::_V0  [protected]
```

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

- lib/UAV_common/src/components/loads.hpp
- lib/UAV_common/src/components/loads.cpp

6.4 Cargo Class Reference

```
#include <loads.hpp>
```

Inheritance diagram for Cargo:

Collaboration diagram for Cargo:

Public Member Functions

- Cargo ()=default
- Cargo (int ammount, double reload, Eigen::Vector3d offset, double mass)

Additional Inherited Members

6.4.1 Constructor & Destructor Documentation

6.4.1.1 Cargo() [1/2]

```
Cargo::Cargo ( )  [default]
```

6.4.1.2 Cargo() [2/2]

```
Cargo::Cargo (
              int ammount,
              double reload,
              Eigen::Vector3d offset,
              double mass )
```

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

- lib/UAV_common/src/components/loads.hpp
- lib/UAV_common/src/components/loads.cpp

6.5 ControlSurfaces Class Reference

Aircraft's control surfaces.

```
#include <control_surfaces.hpp>
```

Public Member Functions

- ControlSurfaces ()
- ControlSurfaces (int noOfSurfaces, Eigen::Matrix< double, 6,-1 > matrix, Eigen::VectorXd min, Eigen::VectorXd max, Eigen::VectorXd trim)

  Constructor.
- Eigen::Vector< double, 6 > getCoefficients () const
- bool setValues (Eigen::VectorXd new_values)
- void restoreTrim ()
- int getNoOfSurface () const
- Eigen::VectorXd getValues () const

6.5.1 Detailed Description

Aircraft's control surfaces.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 ControlSurfaces() [1/2]

```
ControlSurfaces::ControlSurfaces ( )
```

6.5.2.2 ControlSurfaces() [2/2]

```
ControlSurfaces::ControlSurfaces (
            int noOfSurfaces,
            Eigen::Matrix< double, 6,-1 > matrix,
            Eigen::VectorXd min,
            Eigen::VectorXd max,
            Eigen::VectorXd trim )
```

Constructor.

Parameters

noOfSurfaces	number of independent surfaces
matrix	coefficients matrix
min	vector of min angles
max	vector of max angles
trim	vector of trim angles

6.5.3 Member Function Documentation

6.5.3.1 getCoefficients()

```
Eigen::Vector< double, 6 > ControlSurfaces::getCoefficients ( ) const
```

6.5.3.2 getNoOfSurface()

```
int ControlSurfaces::getNoOfSurface ( ) const [inline]
```

6.5.3.3 getValues()

```
Eigen::VectorXd ControlSurfaces::getValues ( ) const  [inline]
```

6.5.3.4 restoreTrim()

```
void ControlSurfaces::restoreTrim ( )
```

6.5.3.5 setValues()

```
bool ControlSurfaces::setValues (
             Eigen::VectorXd new_values )
```

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

- lib/UAV_common/src/components/control_surfaces.hpp
- lib/UAV_common/src/components/control_surfaces.cpp

6.6 Drive Struct Reference

Drive propelling aircraft.

```
#include <drive.hpp>
```

Inheritance diagram for Drive:

Collaboration diagram for Drive:

Public Attributes

- Eigen::Vector3d position
- Eigen::Vector3d axis
- int noOfHinges
- Hinge hinges [2]

6.6.1 Detailed Description

Drive propelling aircraft.

6.6.2 Member Data Documentation

6.6.2.1 axis

```
Eigen::Vector3d Drive::axis
```

6.6.2.2 hinges

```
Hinge Drive::hinges[2]
```

6.6.2.3 noOfHinges

```
int Drive::noOfHinges
```

6.6.2.4 position

```
Eigen::Vector3d Drive::position
```

The documentation for this struct was generated from the following file:

- lib/UAV_common/src/components/drive.hpp

6.7 EKFScalers Struct Reference

Scalers for EKF.

```
#include <navi.hpp>
```

Public Attributes

- double predictScaler
- double updateScaler
- double baroScaler
- double zScaler

6.7.1 Detailed Description

Scalers for EKF.

6.7.2 Member Data Documentation

6.7.2.1 baroScaler

```
double EKFScalers::baroScaler
```

6.7.2.2 predictScaler

```
double EKFScalers::predictScaler
```

6.7.2.3 updateScaler

```
double EKFScalers::updateScaler
```

6.7.2.4 zScaler

```
double EKFScalers::zScaler
```

The documentation for this struct was generated from the following file:

- lib/UAV_common/src/components/navi.hpp

6.8 Hinge Class Reference

Hinge connecting aircraft with drives.

```
#include <hinge.hpp>
```

Public Member Functions

- Hinge ()=default
- Hinge (Eigen::Vector3d axis, double max, double min, double trim)
- Hinge (const Hinge &old)
- Hinge & operator= (const Hinge &old)
- void updateValue (double newValue)

  set new angle on hinge
- const Eigen::Matrix3d getRot ()

  Get rotattion matrix of orientation change due to hinge.

6.8.1 Detailed Description

Hinge connecting aircraft with drives.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Hinge() [1/3]

```
Hinge::Hinge ( )  [default]
```

6.8.2.2 Hinge() [2/3]

```
Hinge::Hinge (
             Eigen::Vector3d axis,
             double max,
             double min,
             double trim )
```

6.8.2.3 Hinge() [3/3]

```
Hinge::Hinge (
             const Hinge & old )
```

6.8.3 Member Function Documentation

6.8.3.1 getRot()

```
const Eigen::Matrix3d Hinge::getRot ( )
```

Get rotattion matrix of orientation change due to hinge.

Returns

rotation matrix

6.8.3.2 operator=()

```
Hinge & Hinge::operator= (
             const Hinge & old )
```

6.8.3.3 updateValue()

```
void Hinge::updateValue (
             double newValue )
```

set new angle on hinge

Parameters

newValue	new angle of hinge

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

- lib/UAV_common/src/components/hinge.hpp
- lib/UAV_common/src/components/hinge.cpp

6.9 Jet Class Reference

Jet rocket engine.

```
#include <drive.hpp>
```

Inheritance diagram for Jet:

Collaboration diagram for Jet:

Public Member Functions

- bool start (double time)

  start jet engine
- double getThrust (double time)

  get thrust in specific time
- double getLastThrust ()

  get last calculated thrust

Public Attributes

- int phases
- Eigen::VectorXd thrust
- Eigen::VectorXd time

6.9.1 Detailed Description

Jet rocket engine.

6.9.2 Member Function Documentation

6.9.2.1 getLastThrust()

```
double Jet::getLastThrust ( )  [inline]
```

get last calculated thrust

Returns

last calculated thrust

6.9.2.2 getThrust()

```
double Jet::getThrust (
            double time )
```

get thrust in specific time

Parameters

time	timestamp

Returns

thrust value in Newtons

6.9.2.3 start()

```
bool Jet::start (
            double time )
```

start jet engine

Parameters

time	timestamp of start

Returns

true if start succesful, false if already started

6.9.3 Member Data Documentation

6.9.3.1 phases

```
int Jet::phases
```

6.9.3.2 thrust

```
Eigen::VectorXd Jet::thrust
```

6.9.3.3 time

```
Eigen::VectorXd Jet::time
```

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

- lib/UAV_common/src/components/drive.hpp
- lib/UAV_common/src/components/drive.cpp

6.10 Load Class Reference

Load of aircraft that can be droped or launched.

```
#include <loads.hpp>
```

Inheritance diagram for Load:

Public Member Functions

- double getMass ()

  get mass of load
- Eigen::Vector3d getOffset ()

  get offset of load
- int release (double time)

  Try to release load.

Protected Member Functions

- Load ()=default
- Load (int ammount, double reload, Eigen::Vector3d offset, double mass)
- Load & operator= (const Load &other)

6.10.1 Detailed Description

Load of aircraft that can be droped or launched.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 Load() [1/2]

```
Load::Load ( )  [protected], [default]
```

6.10.2.2 Load() [2/2]

```
Load::Load (
             int ammount,
             double reload,
             Eigen::Vector3d offset,
             double mass )  [protected]
```

6.10.3 Member Function Documentation

6.10.3.1 getMass()

```
double Load::getMass ( )  [inline]
```

get mass of load

Returns

mass

6.10.3.2 getOffset()

```
Eigen::Vector3d Load::getOffset ( )  [inline]
```

get offset of load

Returns

offset vector

6.10.3.3 operator=()

```
Load & Load::operator= (
             const Load & other )  [protected]
```

6.10.3.4 release()

```
int Load::release (
             double time )
```

Try to release load.

Parameters

time	

Returns

leftover ammount of loads. Return -1 if load is not ready and -2 if out of load

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

- lib/UAV_common/src/components/loads.hpp
- lib/UAV_common/src/components/loads.cpp

6.11 Logger Class Reference

Log vector data with timestamp in file.

```
#include <logger.hpp>
```

Public Member Functions

- Logger (std::string path, std::string fmt="", uint8_t group=0)

  Constructor.
- ∼Logger ()

  Deconstructor.
- void setFmt (std::string fmt)

  Set new format if was not known in constructor.
- void log (double time, std::initializer_list< Eigen::VectorXd > args)

  Log one row.
- void log (double time, std::initializer_list< double > args)

  Log one row.

Static Public Member Functions

- static void setLogDirectory (std::string subdirectory)

  Set global path that log should be created at. Path will be added to relative path of specific log instance.

6.11.1 Detailed Description

Log vector data with timestamp in file.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 Logger()

```
Logger::Logger (
            std::string path,
            std::string fmt = "",
            uint8_t group = 0 )
```

Constructor.

Parameters

path	relative path with log file name.
fmt	format - information about log structure. First line in log file
group	log group - log will be created only if group is in actual LOGGER_MASK

6.11.2.2 ∼Logger()

```
Logger::∼Logger ( )
```

Deconstructor.

6.11.3 Member Function Documentation

6.11.3.1 log() [1/2]

```
void Logger::log (
            double time,
            std::initializer_list< double > args )
```

Log one row.

Parameters

time	timestamp
args	list of doubles

6.11.3.2 log() [2/2]

```
void Logger::log (
             double time,
             std::initializer_list< Eigen::VectorXd > args )
```

Log one row.

Parameters

time	timestamp
args	list of double vectors

6.11.3.3 setFmt()

```
void Logger::setFmt (
             std::string fmt )
```

Set new format if was not known in constructor.

Parameters

fmt	new format

6.11.3.4 setLogDirectory()

```
void Logger::setLogDirectory (
             std::string subdirectory )  [static]
```

Set global path that log should be created at. Path will be added to relative path of specific log instance.

Parameters

subdirectory	new global log path

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

- lib/UAV_common/src/logger/logger.hpp
- lib/UAV_common/src/logger/logger.cpp

6.12 ObjParams Class Reference

Single obj parameters.

```
#include <state.hpp>
```

Public Member Functions

- ObjParams (double mass, double CS_coff)

  Constructor.
- ObjParams (ObjParams &&rhs)

  Moving constructor.
- void setWind (Eigen::Vector3d newWind)

  Set wind vector affecting on object.
- Eigen::Vector3d getWind ()

  Get wind vector.
- void setForce (Eigen::Vector3d newForce)

  Set outer force applied to object.
- Eigen::Vector3d getForce ()

  Get outer force.

Public Attributes

- const int id

  object id
- const double mass

  object mass
- const double CS_coff

  aerodynamic drag force cofficent multipled by aerodynamic field

6.12.1 Detailed Description

Single obj parameters.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 ObjParams() [1/2]

```
ObjParams::ObjParams (
            double mass,
            double CS_coff )  [inline]
```

Constructor.

Parameters

mass	object mass
CS_coff	aerodynamic drag force cofficent multipled by aerodynamic field

6.12.2.2 ObjParams() [2/2]

```
ObjParams::ObjParams (
             ObjParams && rhs )  [inline]
```

Moving constructor.

Parameters

rhs	other instant that should be consumed

6.12.3 Member Function Documentation

6.12.3.1 getForce()

```
Eigen::Vector3d ObjParams::getForce ( )
```

Get outer force.

Returns

outer force vector in N

6.12.3.2 getWind()

```
Eigen::Vector3d ObjParams::getWind ( )
```

Get wind vector.

Returns

wind speed vector in m/s

6.12.3.3 setForce()

```
void ObjParams::setForce (
             Eigen::Vector3d newForce )
```

Set outer force applied to object.

Parameters

newForce	new force vector in N

6.12.3.4 setWind()

```
void ObjParams::setWind (
             Eigen::Vector3d newWind )
```

Set wind vector affecting on object.

Parameters

newWind	new wind speed vector in m/s

6.12.4 Member Data Documentation

6.12.4.1 CS_coff

```
const double ObjParams::CS_coff
```

aerodynamic drag force cofficent multipled by aerodynamic field

6.12.4.2 id

```
const int ObjParams::id
```

object id

6.12.4.3 mass

```
const double ObjParams::mass
```

object mass

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

- src/state.hpp
- src/state.cpp

6.13 ODE Class Reference

Ordinal differencial equation solver.

```
#include <ode.hpp>
```

Inheritance diagram for ODE:

Public Types

- enum ODEMethod { Euler , Heun , RK4 , NONE }

    Supported solving method.

Public Member Functions

- ODE (int micro_steps)

    Constructor.
- virtual ∼ODE ()

    Virtual deconstructor.
- virtual Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::↩ VectorXd)> rhs_fun, double h)=0

    One step of explicit solving algorithm.
- int getMicrosteps () const

    Return microsteps - number of rhs function calls to calculate on step.

Static Public Member Functions

- static ODEMethod fromString (std::string str)

    Parse solving method from string.
- static std::unique_ptr< ODE > factory (ODEMethod method)

    Factory constructing ODE solvers.
- static int getMicrosteps (ODEMethod method)

    Get microsteps of given method.

6.13.1 Detailed Description

Ordinal differencial equation solver.

6.13.2 Member Enumeration Documentation

6.13.2.1 ODEMethod

```
enum ODE::ODEMethod
```

Supported solving method.

Enumerator

Euler	
Heun	
RK4	
NONE	

6.13.3 Constructor & Destructor Documentation

6.13.3.1 ODE()

```
ODE::ODE (
             int micro_steps )
```

Constructor.

6.13.3.2 ∼ODE()

```
virtual ODE::∼ODE ( )  [inline], [virtual]
```

Virtual deconstructor.

6.13.4 Member Function Documentation

6.13.4.1 factory()

```
std::unique_ptr< ODE > ODE::factory (
             ODEMethod method )  [static]
```

Factory constructing ODE solvers.

Parameters

method	type of desired method

Returns

instance of ODE solver

6.13.4.2 fromString()

```
ODE::ODEMethod ODE::fromString (
             std::string str )  [static]
```

Parse solving method from string.

Parameters

str	input string

Returns

solving method if parsed, NONE if unknown

6.13.4.3 getMicrosteps() [1/2]

```
int ODE::getMicrosteps ( ) const
```

Return microsteps - number of rhs function calls to calculate on step.

Returns

microsteps

6.13.4.4 getMicrosteps() [2/2]

```
int ODE::getMicrosteps (
             ODEMethod method )  [static]
```

Get microsteps of given method.

Parameters

method	method type

Returns

number of microstep in one algoritm step

6.13.4.5 step()

```
virtual Eigen::VectorXd ODE::step (
            double t,
            Eigen::VectorXd y0,
            std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun,
            double h ) [pure virtual]
```

One step of explicit solving algorithm.

Parameters

t	start time
y0	start variable
rhs_fun	right-hand-side function, calculation of derivative
h	time step

Returns

Implemented in ODE_RK4, ODE_Heun, and ODE_Euler.

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

- lib/UAV_common/src/ode/ode.hpp
- lib/UAV_common/src/ode/ode.cpp

6.14 ODE_Euler Class Reference

Explicit Euler algorithm.

`#include <ode_impl.hpp>`

Inheritance diagram for ODE_Euler:

Collaboration diagram for ODE_Euler:

Public Member Functions

- ODE_Euler ()
- Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun, double h) override

  One step of explicit solving algorithm.

Additional Inherited Members

6.14.1 Detailed Description

Explicit Euler algorithm.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 ODE_Euler()

```
ODE_Euler::ODE_Euler ( )  [inline]
```

6.14.3 Member Function Documentation

6.14.3.1 step()

```
Eigen::VectorXd ODE_Euler::step (
             double t,
             Eigen::VectorXd y0,
             std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun,
             double h )  [inline], [override], [virtual]
```

One step of explicit solving algorithm.

Parameters

t	start time
y0	start variable
rhs_fun	right-hand-side function, calculation of derivative
h	time step

Returns

Implements ODE.

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

- lib/UAV_common/src/ode/ode_impl.hpp

6.15 ODE_Heun Class Reference

Second order explicit Heun algorithm.

`#include <ode_impl.hpp>`

Inheritance diagram for ODE_Heun:

Collaboration diagram for ODE_Heun:

Public Member Functions

- ODE_Heun ()
- Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun, double h) override

    One step of explicit solving algorithm.

Additional Inherited Members

6.15.1 Detailed Description

Second order explicit Heun algorithm.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 ODE_Heun()

```
ODE_Heun::ODE_Heun ( )  [inline]
```

6.15.3 Member Function Documentation

6.15.3.1 step()

```
Eigen::VectorXd ODE_Heun::step (
             double t,
             Eigen::VectorXd y0,
             std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun,
             double h )  [inline], [override], [virtual]
```

One step of explicit solving algorithm.

Parameters

t	start time
y0	start variable
rhs_fun	right-hand-side function, calculation of derivative
h	time step

Returns

Implements ODE.

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

- lib/UAV_common/src/ode/ode_impl.hpp

6.16 ODE_RK4 Class Reference

Fourth order Runge Kutta algorithm.

```
#include <ode_impl.hpp>
```

Inheritance diagram for ODE_RK4:

Collaboration diagram for ODE_RK4:

Public Member Functions

- ODE_RK4 ()
- Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun, double h) override

    One step of explicit solving algorithm.

Additional Inherited Members

6.16.1 Detailed Description

Fourth order Runge Kutta algorithm.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 ODE_RK4()

```
ODE_RK4::ODE_RK4 ( ) [inline]
```

6.16.3 Member Function Documentation

6.16.3.1 step()

```
Eigen::VectorXd ODE_RK4::step (
            double t,
            Eigen::VectorXd y0,
            std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun,
            double h ) [inline], [override], [virtual]
```

One step of explicit solving algorithm.

Parameters

t	start time
y0	start variable
rhs_fun	right-hand-side function, calculation of derivative
h	time step

Returns

Implements ODE.

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

- lib/UAV_common/src/ode/ode_impl.hpp

6.17 ODETest Class Reference

Inheritance diagram for ODETest:

Collaboration diagram for ODETest:

Protected Member Functions

- void SetUp () override
- void TearDown () override

6.17.1 Member Function Documentation

6.17.1.1 SetUp()

```
void ODETest::SetUp ( )  [inline], [override], [protected]
```

6.17.1.2 TearDown()

```
void ODETest::TearDown ( )  [inline], [override], [protected]
```

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

- lib/UAV_common/src/ode/ode_test.cpp

6.18 Params Class Reference

Simulation parameters.

```
#include <params.hpp>
```

Public Member Functions

- Params ()

  Constructor.
- ∼Params ()

  Deconstructor.

Static Public Member Functions

- static const Params ∗ getSingleton ()

  Get singleton of Params.

Public Attributes

- double STEP_TIME

  Step time of simulation. Step of ODE solving methods.
- std::string ODE_METHOD

  ODE solving method used in simulation.

6.18.1 Detailed Description

Simulation parameters.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 Params()

```
Params::Params ( )
```

Constructor.

6.18.2.2 ∼Params()

```
Params::∼Params ( )
```

Deconstructor.

6.18.3 Member Function Documentation

6.18.3.1 getSingleton()

```
const Params * Params::getSingleton ( )  [static]
```

Get singleton of Params.

Returns

const pointer to Params instance. Return nullptr if not initialized

6.18.4 Member Data Documentation

6.18.4.1 ODE_METHOD

```
std::string Params::ODE_METHOD
```

ODE solving method used in simulation.

6.18.4.2 STEP_TIME

```
double Params::STEP_TIME
```

Step time of simulation. Step of ODE solving methods.

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

- src/params.hpp
- src/params.cpp

6.19 PID Class Reference

PID discrete controller.

```
#include <PID.hpp>
```

Public Member Functions

- PID (double Kp, double Ki, double Kd, double min=std::numeric_limits< double >::min(), double max=std::numeric_limits< double >::max(), AntiWindUpMode antiWindUp=AntiWindUpMode::Clamping)
- ∼PID ()
- void set_dt (double dt)

  Set new time step.
- double calc (double error)

  calc output of controller
- double calc (double error, double dt)

  calc output of controller with specific time step
- void clear ()

  clear internal state

6.19.1 Detailed Description

PID discrete controller.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 PID()

```
PID::PID (
            double Kp,
            double Ki,
            double Kd,
            double min = std::numeric_limits<double>::min(),
            double max = std::numeric_limits<double>::max(),
            AntiWindUpMode antiWindUp = AntiWindUpMode::Clamping )
```

6.19.2.2 ∼PID()

```
PID::∼PID ( )
```

6.19.3 Member Function Documentation

6.19.3.1 calc() [1/2]

```
double PID::calc (
            double error )
```

calc output of controller

Parameters

error	input of controller

Returns

output of controller

6.19.3.2 calc() [2/2]

```
double PID::calc (
             double error,
             double dt )
```

calc output of controller with specific time step

Parameters

error	input of controller
dt	time step

Returns

output of controller

6.19.3.3 clear()

```
void PID::clear ( )
```

clear internal state

6.19.3.4 set_dt()

```
void PID::set_dt (
             double dt )
```

Set new time step.

Parameters

dt	new time step

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

- lib/UAV_common/src/PID/PID.hpp
- lib/UAV_common/src/PID/PID.cpp

6.20 Rotor Struct Reference

Rotor engine with controlled speed.

```
#include <drive.hpp>
```

Inheritance diagram for Rotor:

Collaboration diagram for Rotor:

Public Attributes

- double forceCoff
- double torqueCoff
- int direction
- double timeConstant
- double maxSpeed
- double hoverSpeed

6.20.1 Detailed Description

Rotor engine with controlled speed.

6.20.2 Member Data Documentation

6.20.2.1 direction

```
int Rotor::direction
```

6.20.2.2 forceCoff

```
double Rotor::forceCoff
```

6.20.2.3 hoverSpeed

```
double Rotor::hoverSpeed
```

6.20.2.4 maxSpeed

```
double Rotor::maxSpeed
```

6.20.2.5 timeConstant

```
double Rotor::timeConstant
```

6.20.2.6 torqueCoff

```
double Rotor::torqueCoff
```

The documentation for this struct was generated from the following file:

- lib/UAV_common/src/components/drive.hpp

6.21 SensorParams Struct Reference

Base parameters of a sensor.

```
#include <navi.hpp>
```

Public Attributes

- std::string name
- double sd
- Eigen::Vector3d bias
- double refreshTime

6.21.1 Detailed Description

Base parameters of a sensor.

6.21.2 Member Data Documentation

6.21.2.1 bias

```
Eigen::Vector3d SensorParams::bias
```

6.21.2.2 name

```
std::string SensorParams::name
```

6.21.2.3 refreshTime

```
double SensorParams::refreshTime
```

6.21.2.4 sd

```
double SensorParams::sd
```

The documentation for this struct was generated from the following file:

- lib/UAV_common/src/components/navi.hpp

6.22 Simulation Class Reference

```
#include <simulation.hpp>
```

Public Member Functions

- Simulation (const Params &params)

  Constructor.
- ∼Simulation ()

  Deconstructor.
- void run ()

  Run simulation.
- int addObj (double mass, double CS, Eigen::Vector3d pos, Eigen::Vector3d vel=Eigen::Vector3d())

  Add new object to simulation.
- void removeObj (int id)

  Remove object from simulation.
- void addCommand (std::string msg, zmq::socket_t &sock)

  Handle add new object command.
- void updateWind (std::string msg, zmq::socket_t &sock)

  Handle update wind command.
- void updateForce (std::string msg, zmq::socket_t &sock)

  Handle update force command.
- void solidSurfColision (std::string &msg_str, zmq::socket_t &sock)

  Handle solid surface collision command.
- void calcImpulseForce (int id, double COR, double mi_static, double mi_dynamic, Eigen::Vector3d surface↩ Normal)

  Calculates object state after collision with given surface.

6.22.1 Constructor & Destructor Documentation

6.22.1.1 Simulation()

```
Simulation::Simulation (
            const Params & params )
```

Constructor.

Parameters

params	simulation params

6.22.1.2 ∼Simulation()

```
Simulation::∼Simulation ( )
```

Deconstructor.

6.22.2 Member Function Documentation

6.22.2.1 addCommand()

```
void Simulation::addCommand (
            std::string msg,
            zmq::socket_t & sock )
```

Handle add new object command.

Parameters

msg	message content
sock	zmq socket reply is send by

6.22.2.2 addObj()

```
int Simulation::addObj (
            double mass,
            double CS,
            Eigen::Vector3d pos,
            Eigen::Vector3d vel = Eigen::Vector3d() )
```

Add new object to simulation.

Parameters

mass	obj mass
CS	aerodynamic drag force cofficent multipled by aerodynamic field
pos	start position of object
vel	start velocity of object

Returns

id of added object

6.22.2.3 calcImpulseForce()

```
void Simulation::calcImpulseForce (
            int id,
            double COR,
            double mi_static,
            double mi_dynamic,
            Eigen::Vector3d surfaceNormal )
```

Calculates object state after collision with given surface.

Parameters

id	object id
COR	coefficient of restitution. e = 0 is perfect inelastic collision, e = 1 is perfect elastic collision. 0 < e < 1 is a real-world inelastic collision, in which some kinetic energy is dissipated.
mi_static	static friction coefficient
mi_dynamic	dynamic friction coefficient
surfaceNormal	surface normal vector

6.22.2.4 removeObj()

```
void Simulation::removeObj (
             int id )
```

Remove object from simulation.

Parameters

id	object id

6.22.2.5 run()

```
void Simulation::run ( )
```

Run simulation.

6.22.2.6 solidSurfColision()

```
void Simulation::solidSurfColision (
             std::string & msg_str,
             zmq::socket_t & sock )
```

Handle solid surface collision command.

Parameters

msg	message content
sock	zmq socket reply is send by

6.22.2.7 updateForce()

```
void Simulation::updateForce (
            std::string msg,
            zmq::socket_t & sock )
```

Handle update force command.

Parameters

msg	message content
sock	zmq socket reply is send by

6.22.2.8 updateWind()

```
void Simulation::updateWind (
            std::string msg,
            zmq::socket_t & sock )
```

Handle update wind command.

Parameters

msg	message content
sock	zmq socket reply is send by

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

- src/simulation.hpp
- src/simulation.cpp

6.23 State Class Reference

```
#include <state.hpp>
```

Public Member Functions

- State ()

  Constructor.
- Eigen::VectorXd getState ()

  Get full state as vector.
- void updateState (Eigen::VectorXd newState)

  Update state.
- void updateWind (int id, Eigen::Vector3d newWind)

  update wind speed for obj specified by id

- void updateForce (int id, Eigen::Vector3d newForce)

  update outer force applied to object specified by id
- int addObj (double mass, double CS_coff, Eigen::Vector3d pos, Eigen::Vector3d vel=Eigen::Vector3d())

  Add new object to simulation.
- void removeObj (int id)

  remove object specified by id
- std::string to_string ()

  Serialize state to string.
- int findIndex (int id)

  Find index of object specified by id.
- int getNoObj ()

  Get number of active object in simulation.
- ObjParams ∗ getParams (int index)

  get params of object specified by index
- Eigen::Vector3d getPos (int index)

  Get position of object specified by index.
- Eigen::Vector3d getVel (int index)

  Get velocity of object specified by index.
- void setVel (int index, Eigen::Vector3d newVel)

  Override velocity of object, for example after collision.

Public Attributes

- std::mutex stateMutex

  mutex to manipule on state responses
- double real_time

  time of simulation
- Status status

  status for timed loop

6.23.1 Constructor & Destructor Documentation

6.23.1.1 State()

```
State::State ( )
```

Constructor.

6.23.2 Member Function Documentation

6.23.2.1 addObj()

```
int State::addObj (
             double mass,
             double CS_coff,
             Eigen::Vector3d pos,
             Eigen::Vector3d vel = Eigen::Vector3d() )
```

Add new object to simulation.

Parameters

mass	mass of object
CS_coff	aerodynamic drag force cofficent multipled by aerodynamic field
pos	start position
vel	start velocity

Returns

id of added object

6.23.2.2 findIndex()

```
int State::findIndex (
            int id )
```

Find index of object specified by id.

Parameters

id	object id

Returns

object index

6.23.2.3 getNoObj()

```
int State::getNoObj ( ) [inline]
```

Get number of active object in simulation.

Returns

number of object

6.23.2.4 getParams()

```
ObjParams* State::getParams (
            int index ) [inline]
```

get params of object specified by index

Parameters

index	index of object

Returns

pointer to object params

6.23.2.5 getPos()

```
Eigen::Vector3d State::getPos (
            int index )  [inline]
```

Get position of object specified by index.

Parameters

index	index of object

Returns

position vector

6.23.2.6 getState()

```
Eigen::VectorXd State::getState ( )
```

Get full state as vector.

Returns

state vector

6.23.2.7 getVel()

```
Eigen::Vector3d State::getVel (
            int index )  [inline]
```

Get velocity of object specified by index.

Parameters

index	index of object

Returns

velocity of object

6.23.2.8 removeObj()

```
void State::removeObj (
             int id )
```

remove object specified by id

Parameters

id	id of removing object

6.23.2.9 setVel()

```
void State::setVel (
             int index,
             Eigen::Vector3d newVel )  [inline]
```

Override velocity of object, for example after collision.

Parameters

index	index of object
newVel	new velocity vector

6.23.2.10 to_string()

```
std::string State::to_string ( )
```

Serialize state to string.

Returns

serialized state

6.23.2.11 updateForce()

```
void State::updateForce (
             int id,
             Eigen::Vector3d newForce )
```

update outer force applied to object specified by id

Parameters

id	id of updated obj
newForce	new force value

6.23.2.12 updateState()

```
void State::updateState (
             Eigen::VectorXd newState )
```

Update state.

Parameters

newState	new state vector

6.23.2.13 updateWind()

```
void State::updateWind (
             int id,
             Eigen::Vector3d newWind )
```

update wind speed for obj specified by id

Parameters

id	id of updated obj
newWind	new wind speed vector

6.23.3 Member Data Documentation

6.23.3.1 real_time

```
double State::real_time
```

time of simulation

6.23.3.2 stateMutex

```
std::mutex State::stateMutex
```

mutex to manipule on state responses

6.23.3.3 status

```
Status State::status
```

status for timed loop

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

- src/state.hpp
- src/state.cpp

6.24 TimedLoop Class Reference

Simulation of real-time synchronized loop.

```
#include <timed_loop.hpp>
```

Public Member Functions

- TimedLoop (int periodInMs, std::function< void(void)> func, Status &status)

  Constructor.
- void go ()

  start infinite loop
- void go (uint32_t loops)

  start loop for specific cycle numbers

6.24.1 Detailed Description

Simulation of real-time synchronized loop.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 TimedLoop()

```
TimedLoop::TimedLoop (
            int periodInMs,
            std::function< void(void)> func,
            Status & status )
```

Constructor.

Parameters

periodInMs	loop period in milliseconds
func	function that should be called in loop
status	reference to controlling status

6.24.3 Member Function Documentation

6.24.3.1 go() [1/2]

```
void TimedLoop::go ( )
```

start infinite loop

6.24.3.2 go() [2/2]

```
void TimedLoop::go (
             uint32_t loops )
```

start loop for specific cycle numbers

Parameters

loops	how many cycles should be done

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

- lib/UAV_common/src/timed_loop/timed_loop.hpp
- lib/UAV_common/src/timed_loop/timed_loop.cpp

6.25 UAVparams Struct Reference

Parsed UAV configuration from XML.

```
#include <uav_params.hpp>
```

Collaboration diagram for UAVparams:

Public Member Functions

- UAVparams ()

  Initialize default data.
- ∼UAVparams ()
- void loadConfig (std::string configFile)
- Eigen::VectorXd getRotorTimeContants () const
- Eigen::VectorXd getRotorMaxSpeeds () const
- Eigen::VectorXd getRotorHoverSpeeds () const

Static Public Member Functions

- static const UAVparams ∗ getSingleton ()

Public Attributes

- std::string name
- bool instantRun
- std::string initialMode
- Eigen::Vector3d initialPosition
- Eigen::Vector3d initialOrientation
- Eigen::Vector3d initialVelocity
- double m
- double Ix
- double Iy
- double Iz
- double Ixy
- double Ixz
- double Iyz
- int noOfRotors
- std::unique_ptr< Rotor[ ]> rotors
- int noOfJets
- std::unique_ptr< Jet[ ]> jets
- ControlSurfaces surfaces
- AeroCoefficients aero_coffs
- std::map< std::string, PID > pids
- std::vector< SensorParams > sensors
- AHRSParams ahrs
- EKFScalers ekf
- Eigen::MatrixX4d rotorMixer
- Eigen::MatrixX4d surfaceMixer
- int noOfAmmo
- std::unique_ptr< Ammo[ ]> ammo
- int noOfCargo
- std::unique_ptr< Cargo[ ]> cargo

6.25.1 Detailed Description

Parsed UAV configuration from XML.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 UAVparams()

```
UAVparams::UAVparams ( )
```

Initialize default data.

6.25.2.2 ∼UAVparams()

```
UAVparams::∼UAVparams ( )
```

6.25.3 Member Function Documentation

6.25.3.1 getRotorHoverSpeeds()

```
Eigen::VectorXd UAVparams::getRotorHoverSpeeds ( ) const
```

6.25.3.2 getRotorMaxSpeeds()

```
Eigen::VectorXd UAVparams::getRotorMaxSpeeds ( ) const
```

6.25.3.3 getRotorTimeContants()

```
Eigen::VectorXd UAVparams::getRotorTimeContants ( ) const
```

6.25.3.4 getSingleton()

```
const UAVparams * UAVparams::getSingleton ( ) [static]
```

6.25.3.5 loadConfig()

```
void UAVparams::loadConfig (
             std::string configFile )
```

6.25.4 Member Data Documentation

6.25.4.1 aero_coffs

```
AeroCoefficients UAVparams::aero_coffs
```

6.25.4.2 ahrs

```
AHRSParams UAVparams::ahrs
```

6.25.4.3 ammo

```
std::unique_ptr<Ammo[ ]> UAVparams::ammo
```

6.25.4.4 cargo

```
std::unique_ptr<Cargo[ ]> UAVparams::cargo
```

6.25.4.5 ekf

```
EKFScalers UAVparams::ekf
```

6.25.4.6 initialMode

```
std::string UAVparams::initialMode
```

6.25.4.7 initialOrientation

```
Eigen::Vector3d UAVparams::initialOrientation
```

6.25.4.8 initialPosition

```
Eigen::Vector3d UAVparams::initialPosition
```

6.25.4.9 initialVelocity

```
Eigen::Vector3d UAVparams::initialVelocity
```

6.25.4.10 instantRun

```
bool UAVparams::instantRun
```

6.25.4.11 Ix

```
double UAVparams::Ix
```

6.25.4.12 Ixy

```
double UAVparams::Ixy
```

6.25.4.13 Ixz

```
double UAVparams::Ixz
```

6.25.4.14 Iy

```
double UAVparams::Iy
```

6.25.4.15 Iyz

```
double UAVparams::Iyz
```

6.25.4.16 Iz

```
double UAVparams::Iz
```

6.25.4.17 jets

```
std::unique_ptr<Jet[]> UAVparams::jets
```

6.25.4.18 m

```
double UAVparams::m
```

6.25.4.19 name

```
std::string UAVparams::name
```

6.25.4.20 noOfAmmo

```
int UAVparams::noOfAmmo
```

6.25.4.21 noOfCargo

```
int UAVparams::noOfCargo
```

6.25.4.22 noOfJets

```
int UAVparams::noOfJets
```

6.25.4.23 noOfRotors

```
int UAVparams::noOfRotors
```

6.25.4.24 pids

```
std::map<std::string,PID> UAVparams::pids
```

6.25.4.25 rotorMixer

```
Eigen::MatrixX4d UAVparams::rotorMixer
```

6.25.4.26 rotors

```
std::unique_ptr<Rotor[ ]> UAVparams::rotors
```

6.25.4.27 sensors

```
std::vector<SensorParams> UAVparams::sensors
```

6.25.4.28 surfaceMixer

```
Eigen::MatrixX4d UAVparams::surfaceMixer
```

6.25.4.29 surfaces

```
ControlSurfaces UAVparams::surfaces
```

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

- lib/UAV_common/src/parser/uav_params.hpp
- lib/UAV_common/src/parser/uav_params.cpp

Chapter 7

File Documentation

7.1 lib/UAV_common/header/common.hpp File Reference

```
#include "../src/logger/logger.hpp"
#include "../src/ode/ode.hpp"
#include "../src/PID/PID.hpp"
#include "../src/timed_loop/timed_loop.hpp"
#include "../src/timed_loop/status.hpp"
#include "../src/parser/parser.hpp"
#include "../src/parser/uav_params.hpp"
#include "../src/components/components.hpp"
```

Include dependency graph for common.hpp: This graph shows which files directly or indirectly include this file:

7.2 lib/UAV_common/src/components/aero_coefficients.hpp File Reference

```
#include <Eigen/Dense>
```

Include dependency graph for aero_coefficients.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct AeroCoefficients

  Aerodynamic coefficient.

7.3 lib/UAV_common/src/components/components.hpp File Reference

```
#include "drive.hpp"
#include "control_surfaces.hpp"
#include "aero_coefficients.hpp"
#include "loads.hpp"
#include "navi.hpp"
```

Include dependency graph for components.hpp: This graph shows which files directly or indirectly include this file:

7.4 lib/UAV_common/src/components/control_surfaces.cpp File Reference

```
#include "control_surfaces.hpp"
```

Include dependency graph for control_surfaces.cpp:

7.5 lib/UAV_common/src/components/control_surfaces.hpp File Reference

```
#include <Eigen/Dense>
```

Include dependency graph for control_surfaces.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ControlSurfaces

  Aircraft's control surfaces.

7.6 lib/UAV_common/src/components/drive.cpp File Reference

```
#include "drive.hpp"
```

Include dependency graph for drive.cpp:

7.7 lib/UAV_common/src/components/drive.hpp File Reference

```
#include <Eigen/Dense>
#include "hinge.hpp"
```

Include dependency graph for drive.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct Drive

  Drive propelling aircraft.
- struct Rotor

  Rotor engine with controlled speed.
- class Jet

  Jet rocket engine.

7.8 lib/UAV_common/src/components/hinge.cpp File Reference

```
#include "hinge.hpp"
```

Include dependency graph for hinge.cpp:

Functions

- Eigen::Matrix3d asSkewMatrix (Eigen::Vector3d v)

7.8.1 Function Documentation

7.8.1.1 asSkewMatrix()

```
Eigen::Matrix3d asSkewMatrix (
            Eigen::Vector3d v )
```

7.9 lib/UAV_common/src/components/hinge.hpp File Reference

```
#include <Eigen/Dense>
#include <mutex>
#include <memory>
```

Include dependency graph for hinge.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Hinge

  Hinge connecting aircraft with drives.

7.10 lib/UAV_common/src/components/loads.cpp File Reference

```
#include "loads.hpp"
#include <limits>
```

Include dependency graph for loads.cpp:

7.11 lib/UAV_common/src/components/loads.hpp File Reference

```
#include <Eigen/Dense>
#include <atomic>
```

Include dependency graph for loads.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Load

  Load of aircraft that can be droped or launched.
- class Ammo
- class Cargo

7.12 lib/UAV_common/src/components/navi.hpp File Reference

```
#include <Eigen/Dense>
```

Include dependency graph for navi.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct SensorParams

  Base parameters of a sensor.
- struct AHRSParams

  AHRS parameters.
- struct EKFScalers

  Scalers for EKF.

7.13 lib/UAV_common/src/logger/logger.cpp File Reference

```
#include "logger.hpp"
#include <Eigen/Dense>
#include <iostream>
#include <fstream>
#include <initializer_list>
#include <string>
#include <filesystem>
```

Include dependency graph for logger.cpp:

Functions

- bool shouldLog (uint8_t group)

7.13.1 Function Documentation

7.13.1.1 shouldLog()

```
bool shouldLog (
          uint8_t group )
```

7.14 lib/UAV_common/src/logger/logger.hpp File Reference

```
#include <Eigen/Dense>
#include <iostream>
#include <fstream>
#include <initializer_list>
#include <string>
#include <filesystem>
```

Include dependency graph for logger.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Logger

  Log vector data with timestamp in file.

Macros

- #define LOGGER_MASK -1

7.14.1 Macro Definition Documentation

7.14.1.1 LOGGER_MASK

```
#define LOGGER_MASK -1
```

7.15 lib/UAV_common/src/ode/ode.cpp File Reference

```
#include "ode.hpp"
#include "ode_impl.hpp"
```

Include dependency graph for ode.cpp:

7.16 lib/UAV_common/src/ode/ode.hpp File Reference

```
#include <functional>
#include <memory>
#include <Eigen/Dense>
```

Include dependency graph for ode.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ODE

  Ordinal differencial equation solver.

7.17 lib/UAV_common/src/ode/ode_impl.hpp File Reference

```
#include "ode.hpp"
```

Include dependency graph for ode_impl.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ODE_Euler

  Explicit Euler algorithm.
- class ODE_Heun

  Second order explicit Heun algorithm.
- class ODE_RK4

  Fourth order Runge Kutta algorithm.

7.18 lib/UAV_common/src/ode/ode_test.cpp File Reference

```
#include "ode.hpp"
#include <gtest/gtest.h>
#include <numbers>
```

Include dependency graph for ode_test.cpp:

Classes

- class ODETest

Functions

- std::vector< ODE::ODEMethod > getMethodsToTest ()
- TEST_F (ODETest, FromStringTest)
- TEST_F (ODETest, FactoryTest)
- TEST_P (ODETest, TestConstFunction)
- TEST_P (ODETest, TestFirstOrder)
- TEST_P (ODETest, TestRHSCalls)
- INSTANTIATE_TEST_SUITE_P (TestDerivedClasses, ODETest, testing::ValuesIn(getMethodsToTest()))
- int main (int argc, char **argv)

7.18.1 Function Documentation

7.18.1.1 getMethodsToTest()

```
std::vector<ODE::ODEMethod> getMethodsToTest ( )
```

7.18.1.2 INSTANTIATE_TEST_SUITE_P()

```
INSTANTIATE_TEST_SUITE_P (
            TestDerivedClasses ,
            ODETest ,
            testing::ValuesIn(getMethodsToTest())  )
```

7.18.1.3 main()

```
int main (
            int argc,
            char ** argv )
```

7.18.1.4 TEST_F() [1/2]

```
TEST_F (
            ODETest ,
            FactoryTest  )
```

7.18.1.5 TEST_F() [2/2]

```
TEST_F (
            ODETest ,
            FromStringTest  )
```

7.18.1.6 TEST_P() [1/3]

```
TEST_P (
            ODETest ,
            TestConstFunction  )
```

7.18.1.7 TEST_P() [2/3]

```
TEST_P (
            ODETest ,
            TestFirstOrder  )
```

7.18.1.8 TEST_P() [3/3]

```
TEST_P (
            ODETest ,
            TestRHSCalls  )
```

7.19 lib/UAV_common/src/parser/parser.cpp File Reference

```
#include "parser.hpp"
#include <Eigen/Dense>
#include <iostream>
#include <sstream>
```

Include dependency graph for parser.cpp:

Functions

- Eigen::MatrixXd parseMatrixXd (const std::string &input, int R, int C, char delimiter)

  Parse input string to double matrix of specific shape and delimiter.
- Eigen::VectorXd parseVectorXd (std::string str, int noOfElem, char delimiter)

  Parse input string to double vector of specific length and delimiter.

7.19.1 Function Documentation

7.19.1.1 parseMatrixXd()

```
Eigen::MatrixXd parseMatrixXd (
             const std::string & input,
             int R,
             int C,
             char delimiter = ' ' )
```

Parse input string to double matrix of specific shape and delimiter.

Parameters

input	input string
R	number of rows
C	number of columns
delimiter	delimiter

Returns

parsed matrix

7.19.1.2 parseVectorXd()

```
Eigen::VectorXd parseVectorXd (
             std::string str,
             int noOfElem,
             char delimiter = ' ' )
```

Parse input string to double vector of specific length and delimiter.

Parameters

str	input string
noOfElem	length of vector
delimiter	delimiter

Returns

parsed vector

7.20 lib/UAV_common/src/parser/parser.hpp File Reference

```
#include <Eigen/Dense>
```

Include dependency graph for parser.hpp: This graph shows which files directly or indirectly include this file:

Functions

- Eigen::MatrixXd parseMatrixXd (const std::string &input, int R, int C, char delimiter=' ')

  Parse input string to double matrix of specific shape and delimiter.
- Eigen::VectorXd parseVectorXd (std::string str, int noOfElem, char delimiter=' ')

  Parse input string to double vector of specific length and delimiter.

7.20.1 Function Documentation

7.20.1.1 parseMatrixXd()

```
Eigen::MatrixXd parseMatrixXd (
             const std::string & input,
             int R,
             int C,
             char delimiter = ' ' )
```

Parse input string to double matrix of specific shape and delimiter.

Parameters

input	input string
R	number of rows
C	number of columns
delimiter	delimiter

Returns

parsed matrix

7.20.1.2 parseVectorXd()

```
Eigen::VectorXd parseVectorXd (
            std::string str,
            int noOfElem,
            char delimiter = ' ' )
```

Parse input string to double vector of specific length and delimiter.

Parameters

str	input string
noOfElem	length of vector
delimiter	delimiter

Returns

parsed vector

7.21 lib/UAV_common/src/parser/uav_params.cpp File Reference

```
#include <Eigen/Dense>
#include "uav_params.hpp"
#include <iostream>
#include <fstream>
#include <filesystem>
#include <mutex>
#include "rapidxml/rapidxml.hpp"
#include "parser.hpp"
```

Include dependency graph for uav_params.cpp:

Functions

- void parseHinge (rapidxml::xml_node<> *hingeNode, Hinge *hinge)
- PID parsePID (rapidxml::xml_node<> *PIDNode)

7.21.1 Function Documentation

7.21.1.1 parseHinge()

```
void parseHinge (
            rapidxml::xml_node<> * hingeNode,
            Hinge * hinge )
```

7.21.1.2 parsePID()

```
PID parsePID (
            rapidxml::xml_node<> * PIDNode )
```

7.22 lib/UAV_common/src/parser/uav_params.hpp File Reference

```
#include <Eigen/Dense>
#include <mutex>
#include <memory>
#include <map>
#include "rapidxml/rapidxml.hpp"
#include "../components/components.hpp"
#include "../PID/PID.hpp"
```

Include dependency graph for uav_params.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct UAVparams

  Parsed UAV configuration from XML.

7.23 lib/UAV_common/src/PID/PID.cpp File Reference

```
#include "PID.hpp"
#include <limits>
#include <algorithm>
```

Include dependency graph for PID.cpp:

7.24 lib/UAV_common/src/PID/PID.hpp File Reference

```
#include <limits>
```

Include dependency graph for PID.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class PID

  PID discrete controller.

Enumerations

- enum AntiWindUpMode { None , Clamping }

  Methods of handling windup in controller.

7.24.1 Enumeration Type Documentation

7.24.1.1 AntiWindUpMode

```
enum AntiWindUpMode
```

Methods of handling windup in controller.

Enumerator

None	
Clamping	

7.25 lib/UAV_common/src/timed_loop/status.hpp File Reference

This graph shows which files directly or indirectly include this file:

Enumerations

- enum Status { idle = 1 , running = 2 , exiting = 3 , reload = 4 }

  status of timed loop. Control it's job

7.25.1 Enumeration Type Documentation

7.25.1.1 Status

```
enum Status
```

status of timed loop. Control it's job

Enumerator

idle	loop is ready to run
running	loop is running
exiting	loop will be break in next occasion.
reload	loop job should be reloaded

7.26 lib/UAV_common/src/timed_loop/timed_loop.cpp File Reference

```
#include "timed_loop.hpp"
#include <stdint.h>
#include <chrono>
#include <thread>
#include "status.hpp"
#include <iostream>
```

Include dependency graph for timed_loop.cpp:

7.27 lib/UAV_common/src/timed_loop/timed_loop.hpp File Reference

```
#include <stdint.h>
#include <functional>
#include "status.hpp"
```

Include dependency graph for timed_loop.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class TimedLoop

  Simulation of real-time synchronized loop.

7.28 src/defines.hpp File Reference

This graph shows which files directly or indirectly include this file:

Namespaces

- def

  Simulation constants.

Variables

- const double def::GRAVITY_CONST = 9.81

  Gravity constant on Earth in m/s2.
- const double def::FRICTION_EPS = 0.001

  minimal friction that is calculated (numerical float eps)
- const double def::GENTLY_PUSH = 0.15

  artificial force cofficient. Protect again diving objects in horizontal wall
- const double def::DEFAULT_AIR_DENSITY = 1.224

  Dry air density in normal conditions in kg/m3.

7.29 src/main.cpp File Reference

```
#include <iostream>
#include <Eigen/Dense>
#include <cxxopts.hpp>
#include "simulation.hpp"
#include "common.hpp"
#include "params.hpp"
```

Include dependency graph for main.cpp:

Functions

- Params parseArgs (int argc, char **argv, Params &p)

  Parse CL arguments.
- int main (int argc, char **argv)

7.29.1 Function Documentation

7.29.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

7.29.1.2 parseArgs()

```
Params parseArgs (
            int argc,
            char ** argv,
            Params & p )
```

Parse CL arguments.

Parameters

argc	number of argument
argv	argument array
p	reference to params instant that should be filled

7.30 src/params.cpp File Reference

```
#include "params.hpp"
#include <iostream>
```

Include dependency graph for params.cpp:

7.31 src/params.hpp File Reference

```
#include <string>
```

Include dependency graph for params.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Params

  Simulation parameters.

7.32 src/simulation.cpp File Reference

```
#include <Eigen/Dense>
#include <zmq.hpp>
#include <iostream>
#include <cstdio>
#include <thread>
#include <mutex>
#include <functional>
#include <map>
#include <filesystem>
#include "simulation.hpp"
#include "common.hpp"
#include "state.hpp"
```

Include dependency graph for simulation.cpp:

Functions

- bool isNormal (double factor)

7.32.1 Function Documentation

7.32.1.1 isNormal()

```
bool isNormal (
            double factor )
```

7.33 src/simulation.hpp File Reference

```
#include <zmq.hpp>
#include <thread>
#include "state.hpp"
#include <Eigen/Dense>
#include <functional>
#include "common.hpp"
#include "defines.hpp"
#include "params.hpp"
```

Include dependency graph for simulation.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Simulation

7.34 src/state.cpp File Reference

```
#include <Eigen/Dense>
#include <mutex>
#include <iostream>
#include "state.hpp"
#include "common.hpp"
#include "params.hpp"
#include "defines.hpp"
```

Include dependency graph for state.cpp:

7.35 src/state.hpp File Reference

```
#include <Eigen/Dense>
#include <zmq.hpp>
#include <thread>
#include <vector>
#include <mutex>
#include <atomic>
#include "common.hpp"
```

Include dependency graph for state.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ObjParams

  Single obj parameters.
- class State

Index

_V0
    Ammo, 15
~Logger
    Logger, 28
~ODE
    ODE, 34
~PID
    PID, 43
~Params
    Params, 41
~Simulation
    Simulation, 48
~UAVparams
    UAVparams, 61

addCommand
    Simulation, 49
addObj
    Simulation, 49
    State, 53
aero_coffs
    UAVparams, 62
AeroCoefficients, 11
    C0, 11
    Cab, 11
    Cpqr, 12
    d, 12
    eAR, 12
    S, 12
    stallLimit, 12
ahrs
    UAVparams, 62
AHRSParams, 12
    alpha, 13
    Q, 13
    R, 13
    type, 13
alpha
    AHRSParams, 13
Ammo, 14
    _V0, 15
    Ammo, 14
    getV0, 14
    operator=, 15
ammo
    UAVparams, 62
AntiWindUpMode
    PID.hpp, 77
asSkewMatrix
    hinge.cpp, 69
axis
    Drive, 18

baroScaler
    EKFScalers, 20
bias
    SensorParams, 47

C0
    AeroCoefficients, 11
Cab
    AeroCoefficients, 11
calc
    PID, 43, 44
calcImpulseForce
    Simulation, 49
Cargo, 15
    Cargo, 16
cargo
    UAVparams, 62
Clamping
    PID.hpp, 78
clear
    PID, 44
ControlSurfaces, 16
    ControlSurfaces, 17
    getCoefficients, 17
    getNoOfSurface, 17
    getValues, 17
    restoreTrim, 18
    setValues, 18
Cpqr
    AeroCoefficients, 12
CS_coff
    ObjParams, 32

d
    AeroCoefficients, 12
def, 9
    DEFAULT_AIR_DENSITY, 9
    FRICTION_EPS, 9
    GENTLY_PUSH, 10
    GRAVITY_CONST, 10
DEFAULT_AIR_DENSITY
    def, 9
direction
    Rotor, 45
Drive, 18
    axis, 18
    hinges, 19

noOfHinges, 19
position, 19

eAR
AeroCoefficients, 12
ekf
UAVparams, 62
EKFScalers, 19
baroScaler, 20
predictScaler, 20
updateScaler, 20
zScaler, 20
Euler
ODE, 34
exiting
status.hpp, 78

factory
ODE, 34
findIndex
State, 54
forceCoff
Rotor, 45
FRICTION_EPS
def, 9
fromString
ODE, 34

GENTLY_PUSH
def, 10
getCoefficients
ControlSurfaces, 17
getForce
ObjParams, 31
getLastThrust
Jet, 23
getMass
Load, 26
getMethodsToTest
ode_test.cpp, 72
getMicrosteps
ODE, 35
getNoObj
State, 54
getNoOfSurface
ControlSurfaces, 17
getOffset
Load, 26
getParams
State, 54
getPos
State, 55
getRot
Hinge, 21
getRotorHoverSpeeds
UAVparams, 61
getRotorMaxSpeeds
UAVparams, 61
getRotorTimeContants
UAVparams, 61
getSingleton
Params, 42
UAVparams, 61
getState
State, 55
getThrust
Jet, 24
getV0
Ammo, 14
getValues
ControlSurfaces, 17
getVel
State, 55
getWind
ObjParams, 31
go
TimedLoop, 59
GRAVITY_CONST
def, 10

Heun
ODE, 34
Hinge, 20
getRot, 21
Hinge, 21
operator=, 21
updateValue, 22
hinge.cpp
asSkewMatrix, 69
hinges
Drive, 19
hoverSpeed
Rotor, 45

id
ObjParams, 32
idle
status.hpp, 78
initialMode
UAVparams, 62
initialOrientation
UAVparams, 62
initialPosition
UAVparams, 63
initialVelocity
UAVparams, 63
INSTANTIATE_TEST_SUITE_P
ode_test.cpp, 72
instantRun
UAVparams, 63
isNormal
simulation.cpp, 81
Ix
UAVparams, 63
Ixy
UAVparams, 63
Ixz
UAVparams, 63

Iy
UAVparams, 63
Iyz
UAVparams, 63
Iz
UAVparams, 64

Jet, 23
getLastThrust, 23
getThrust, 24
phases, 24
start, 24
thrust, 25
time, 25
jets
UAVparams, 64

lib/UAV_common/header/common.hpp, 67
lib/UAV_common/src/components/aero_coefficients.hpp, 67
lib/UAV_common/src/components/components.hpp, 67
lib/UAV_common/src/components/control_surfaces.cpp, 68
lib/UAV_common/src/components/control_surfaces.hpp, 68
lib/UAV_common/src/components/drive.cpp, 68
lib/UAV_common/src/components/drive.hpp, 68
lib/UAV_common/src/components/hinge.cpp, 68
lib/UAV_common/src/components/hinge.hpp, 69
lib/UAV_common/src/components/loads.cpp, 69
lib/UAV_common/src/components/loads.hpp, 69
lib/UAV_common/src/components/navi.hpp, 70
lib/UAV_common/src/logger/logger.cpp, 70
lib/UAV_common/src/logger/logger.hpp, 70
lib/UAV_common/src/ode/ode.cpp, 71
lib/UAV_common/src/ode/ode.hpp, 71
lib/UAV_common/src/ode/ode_impl.hpp, 71
lib/UAV_common/src/ode/ode_test.cpp, 72
lib/UAV_common/src/parser/parser.cpp, 74
lib/UAV_common/src/parser/parser.hpp, 75
lib/UAV_common/src/parser/uav_params.cpp, 76
lib/UAV_common/src/parser/uav_params.hpp, 77
lib/UAV_common/src/PID/PID.cpp, 77
lib/UAV_common/src/PID/PID.hpp, 77
lib/UAV_common/src/timed_loop/status.hpp, 78
lib/UAV_common/src/timed_loop/timed_loop.cpp, 78
lib/UAV_common/src/timed_loop/timed_loop.hpp, 79
Load, 25
getMass, 26
getOffset, 26
Load, 26
operator=, 26
release, 27
loadConfig
UAVparams, 61
log
Logger, 28, 29
Logger, 27
∼Logger, 28
log, 28, 29
Logger, 28
setFmt, 29
setLogDirectory, 29
logger.cpp
shouldLog, 70
logger.hpp
LOGGER_MASK, 71
LOGGER_MASK
logger.hpp, 71

m
UAVparams, 64
main
main.cpp, 80
ode_test.cpp, 72
main.cpp
main, 80
parseArgs, 80
mass
ObjParams, 32
maxSpeed
Rotor, 46

name
SensorParams, 47
UAVparams, 64
NONE
ODE, 34
None
PID.hpp, 78
noOfAmmo
UAVparams, 64
noOfCargo
UAVparams, 64
noOfHinges
Drive, 19
noOfJets
UAVparams, 64
noOfRotors
UAVparams, 64

ObjParams, 30
CS_coff, 32
getForce, 31
getWind, 31
id, 32
mass, 32
ObjParams, 30, 31
setForce, 31
setWind, 32
ODE, 33
∼ODE, 34
Euler, 34
factory, 34
fromString, 34
getMicrosteps, 35
Heun, 34
NONE, 34

ODE, 34
ODEMethod, 33
RK4, 34
step, 35
ODE_Euler, 36
ODE_Euler, 37
step, 37
ODE_Heun, 37
ODE_Heun, 38
step, 38
ODE_METHOD
Params, 42
ODE_RK4, 39
ODE_RK4, 39
step, 39
ode_test.cpp
getMethodsToTest, 72
INSTANTIATE_TEST_SUITE_P, 72
main, 72
TEST_F, 73
TEST_P, 73
ODEMethod
ODE, 33
ODETest, 40
SetUp, 40
TearDown, 40
operator=
Ammo, 15
Hinge, 21
Load, 26

Params, 41
~Params, 41
getSingleton, 42
ODE_METHOD, 42
Params, 41
STEP_TIME, 42
parseArgs
main.cpp, 80
parseHinge
uav_params.cpp, 76
parseMatrixXd
parser.cpp, 74
parser.hpp, 75
parsePID
uav_params.cpp, 76
parser.cpp
parseMatrixXd, 74
parseVectorXd, 74
parser.hpp
parseMatrixXd, 75
parseVectorXd, 76
parseVectorXd
parser.cpp, 74
parser.hpp, 76
phases
Jet, 24
PID, 42
~PID, 43
calc, 43, 44
clear, 44
PID, 43
set_dt, 44
PID.hpp
AntiWindUpMode, 77
Clamping, 78
None, 78
pids
UAVparams, 65
position
Drive, 19
predictScaler
EKFScalers, 20

Q
AHRSParams, 13

R
AHRSParams, 13
real_time
State, 57
refreshTime
SensorParams, 47
release
Load, 27
reload
status.hpp, 78
removeObj
Simulation, 51
State, 56
restoreTrim
ControlSurfaces, 18
RK4
ODE, 34
Rotor, 45
direction, 45
forceCoff, 45
hoverSpeed, 45
maxSpeed, 46
timeConstant, 46
torqueCoff, 46
rotorMixer
UAVparams, 65
rotors
UAVparams, 65
run
Simulation, 51
running
status.hpp, 78

S
AeroCoefficients, 12
sd
SensorParams, 47
SensorParams, 46
bias, 47
name, 47
refreshTime, 47

sd, 47
sensors
UAVparams, 65
set_dt
PID, 44
setFmt
Logger, 29
setForce
ObjParams, 31
setLogDirectory
Logger, 29
SetUp
ODETest, 40
setValues
ControlSurfaces, 18
setVel
State, 56
setWind
ObjParams, 32
shouldLog
logger.cpp, 70
Simulation, 47
∼Simulation, 48
addCommand, 49
addObj, 49
calcImpulseForce, 49
removeObj, 51
run, 51
Simulation, 48
solidSurfColision, 51
updateForce, 51
updateWind, 52
simulation.cpp
isNormal, 81
solidSurfColision
Simulation, 51
src/defines.hpp, 79
src/main.cpp, 79
src/params.cpp, 80
src/params.hpp, 80
src/simulation.cpp, 81
src/simulation.hpp, 81
src/state.cpp, 82
src/state.hpp, 82
stallLimit
AeroCoefficients, 12
start
Jet, 24
State, 52
addObj, 53
findIndex, 54
getNoObj, 54
getParams, 54
getPos, 55
getState, 55
getVel, 55
real_time, 57
removeObj, 56
setVel, 56
State, 53
stateMutex, 58
status, 58
to_string, 56
updateForce, 56
updateState, 57
updateWind, 57
stateMutex
State, 58
Status
status.hpp, 78
status
State, 58
status.hpp
exiting, 78
idle, 78
reload, 78
running, 78
Status, 78
step
ODE, 35
ODE_Euler, 37
ODE_Heun, 38
ODE_RK4, 39
STEP_TIME
Params, 42
surfaceMixer
UAVparams, 65
surfaces
UAVparams, 65

TearDown
ODETest, 40
TEST_F
ode_test.cpp, 73
TEST_P
ode_test.cpp, 73
thrust
Jet, 25
time
Jet, 25
timeConstant
Rotor, 46
TimedLoop, 58
go, 59
TimedLoop, 58
to_string
State, 56
torqueCoff
Rotor, 46
type
AHRSParams, 13

uav_params.cpp
parseHinge, 76
parsePID, 76
UAVparams, 59
∼UAVparams, 61

aero_coffs, 62
ahrs, 62
ammo, 62
cargo, 62
ekf, 62
getRotorHoverSpeeds, 61
getRotorMaxSpeeds, 61
getRotorTimeContants, 61
getSingleton, 61
initialMode, 62
initialOrientation, 62
initialPosition, 63
initialVelocity, 63
instantRun, 63
Ix, 63
Ixy, 63
Ixz, 63
Iy, 63
Iyz, 63
Iz, 64
jets, 64
loadConfig, 61
m, 64
name, 64
noOfAmmo, 64
noOfCargo, 64
noOfJets, 64
noOfRotors, 64
pids, 65
rotorMixer, 65
rotors, 65
sensors, 65
surfaceMixer, 65
surfaces, 65
UAVparams, 61

updateForce
Simulation, 51
State, 56

updateScaler
EKFScalers, 20

updateState
State, 57

updateValue
Hinge, 22

updateWind
Simulation, 52
State, 57

zScaler
EKFScalers, 20