UAV physic engine

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 DOUBLE_EPS . . . 10

5.1.2.2 FRICTION_EPS . . . 10

5.1.2.3 GENTLY_PUSH . . . 10

5.1.2.4 GRAVITY_CONST . . . 10

5.1.2.5 MIXING_FUNCTION . . . 10

5.1.2.6 STEP_TIME . . . 10

5.1.2.7 validityOfForce . . . 11

5.2 zmq_recv Namespace Reference . . . 11

5.2.1 Variable Documentation . . . 11

5.2.1.1 context . . . 11

5.2.1.2 s . . . 11

5.2.1.3 socket . . . 11

5.2.1.4 topicfilter . . . 11

5.3 zmq_recv_last Namespace Reference . . . 12

5.3.1 Variable Documentation . . . 12

5.3.1.1 context . . . 12

5.3.1.2 s . . . 12

5.3.1.3 socket . . . 12

5.3.1.4 topicfilter . . . 12

5.4 zmq_send Namespace Reference . . . 12

5.4.1 Variable Documentation . . . 13

5.4.1.1 context . . . 13

5.4.1.2 counter . . . 13

5.4.1.3 socket . . . 13

5.5 zmq_send_tcp Namespace Reference . . . 13

5.5.1 Variable Documentation . . . 13

5.5.1.1 angle . . . 13

5.5.1.2 context . . . 13

5.5.1.3 socket 13

6 Class Documentation **15**

6.1 AeroCoefficients Struct Reference 15

6.1.1 Detailed Description 15

6.1.2 Member Data Documentation 15

6.1.2.1 C0 15

6.1.2.2 Cab 16

6.1.2.3 Cpqr 16

6.1.2.4 d 16

6.1.2.5 eAR 16

6.1.2.6 S 16

6.1.2.7 stallLimit 16

6.2 AHRSParams Struct Reference 16

6.2.1 Detailed Description 17

6.2.2 Member Data Documentation 17

6.2.2.1 alpha 17

6.2.2.2 Q 17

6.2.2.3 R 17

6.2.2.4 type 17

6.3 Aircraft Class Reference 18

6.3.1 Detailed Description 19

6.3.2 Constructor & Destructor Documentation 19

6.3.2.1 Aircraft() 19

6.3.2.2 ∼Aircraft() 19

6.3.3 Member Function Documentation 19

6.3.3.1 calcImpulseForce() 19

6.3.3.2 calcMomentumConservanceConservation() 20

6.3.3.3 dropCargo() 20

6.3.3.4 reduceMass() 21

6.3.3.5 RHS() 21

6.3.3.6 sendState() 21

6.3.3.7 setHinge() 22

6.3.3.8 setSurface() 22

6.3.3.9 shootAmmo() 23

6.3.3.10 startJet() 23

6.3.3.11 trim() 23

6.3.3.12 update() 24

6.3.4 Member Data Documentation 24

6.3.4.1 aero 24

6.3.4.2 ammo 24

6.3.4.3 cargo 24

6.3.4.4 invMassMatrix . . . 24
6.3.4.5 jets . . . 24
6.3.4.6 massMatrix . . . 24
6.3.4.7 mtx . . . 25
6.3.4.8 noOfAmmo . . . 25
6.3.4.9 noOfCargo . . . 25
6.3.4.10 noOfJets . . . 25
6.3.4.11 noOfRotors . . . 25
6.3.4.12 ode . . . 25
6.3.4.13 rotors . . . 25
6.3.4.14 state . . . 25
6.3.4.15 surfaces . . . 26
6.4 Ammo Class Reference . . . 26
6.4.1 Constructor & Destructor Documentation . . . 26
6.4.1.1 Ammo() [1/2] . . . 26
6.4.1.2 Ammo() [2/2] . . . 27
6.4.2 Member Function Documentation . . . 27
6.4.2.1 getV0() . . . 27
6.4.2.2 operator=() . . . 27
6.4.3 Member Data Documentation . . . 27
6.4.3.1 _V0 . . . 27
6.5 Atmosphere Class Reference . . . 28
6.5.1 Detailed Description . . . 28
6.5.2 Constructor & Destructor Documentation . . . 28
6.5.2.1 Atmosphere() . . . 28
6.5.2.2 ~Atmosphere() . . . 29
6.5.3 Member Function Documentation . . . 29
6.5.3.1 getAirDensity() . . . 29
6.5.3.2 getAirPressure() . . . 29
6.5.3.3 getAirTemperature() . . . 29
6.5.3.4 getSingleton() . . . 30
6.5.3.5 getWind() . . . 30
6.5.3.6 update() . . . 30
6.6 AtmosphereInfo Struct Reference . . . 30
6.6.1 Detailed Description . . . 31
6.6.2 Member Data Documentation . . . 31
6.6.2.1 air_density . . . 31
6.6.2.2 air_pressure . . . 31
6.6.2.3 air_temperature . . . 31
6.6.2.4 wind . . . 31
6.7 Cargo Class Reference . . . 32
6.7.1 Constructor & Destructor Documentation . . . 32

6.7.1.1 Cargo() [1/2] . . . 32
6.7.1.2 Cargo() [2/2] . . . 32
6.8 ControlSurfaces Class Reference . . . 32
6.8.1 Detailed Description . . . 33
6.8.2 Constructor & Destructor Documentation . . . 33
6.8.2.1 ControlSurfaces() [1/2] . . . 33
6.8.2.2 ControlSurfaces() [2/2] . . . 33
6.8.3 Member Function Documentation . . . 34
6.8.3.1 getCoefficients() . . . 34
6.8.3.2 getNoOfSurface() . . . 34
6.8.3.3 getValues() . . . 34
6.8.3.4 restoreTrim() . . . 34
6.8.3.5 setValues() . . . 34
6.9 Drive Struct Reference . . . 34
6.9.1 Detailed Description . . . 35
6.9.2 Member Data Documentation . . . 35
6.9.2.1 axis . . . 35
6.9.2.2 hinges . . . 35
6.9.2.3 noOfHinges . . . 35
6.9.2.4 position . . . 35
6.10 EKFScalers Struct Reference . . . 35
6.10.1 Detailed Description . . . 36
6.10.2 Member Data Documentation . . . 36
6.10.2.1 baroScaler . . . 36
6.10.2.2 predictScaler . . . 36
6.10.2.3 updateScaler . . . 36
6.10.2.4 zScaler . . . 36
6.11 Forces Class Reference . . . 36
6.11.1 Member Function Documentation . . . 37
6.11.1.1 aerodynamic_loads() . . . 37
6.11.1.2 angularAcceleration() . . . 37
6.11.1.3 generateCharacteristics() . . . 39
6.11.1.4 gravity_loads() . . . 39
6.11.1.5 jet_lift_loads() . . . 39
6.11.1.6 rotor_lift_loads() . . . 40
6.12 Hinge Class Reference . . . 40
6.12.1 Detailed Description . . . 41
6.12.2 Constructor & Destructor Documentation . . . 41
6.12.2.1 Hinge() [1/3] . . . 41
6.12.2.2 Hinge() [2/3] . . . 41
6.12.2.3 Hinge() [3/3] . . . 41
6.12.3 Member Function Documentation . . . 41

6.12.3.1 getRot() . . . 42
6.12.3.2 operator=() . . . 42
6.12.3.3 updateValue() . . . 42
6.13 Jet Class Reference . . . 42
6.13.1 Detailed Description . . . 43
6.13.2 Member Function Documentation . . . 43
6.13.2.1 getLastThrust() . . . 43
6.13.2.2 getThrust() . . . 43
6.13.2.3 start() . . . 44
6.13.3 Member Data Documentation . . . 44
6.13.3.1 phases . . . 44
6.13.3.2 thrust . . . 44
6.13.3.3 time . . . 44
6.14 Load Class Reference . . . 45
6.14.1 Detailed Description . . . 45
6.14.2 Constructor & Destructor Documentation . . . 45
6.14.2.1 Load() [1/2] . . . 45
6.14.2.2 Load() [2/2] . . . 45
6.14.3 Member Function Documentation . . . 46
6.14.3.1 getMass() . . . 46
6.14.3.2 getOffset() . . . 46
6.14.3.3 operator=() . . . 46
6.14.3.4 release() . . . 46
6.15 Logger Class Reference . . . 47
6.15.1 Detailed Description . . . 47
6.15.2 Constructor & Destructor Documentation . . . 47
6.15.2.1 Logger() . . . 47
6.15.2.2 ~Logger() . . . 48
6.15.3 Member Function Documentation . . . 48
6.15.3.1 log() [1/2] . . . 48
6.15.3.2 log() [2/2] . . . 48
6.15.3.3 setFmt() . . . 49
6.15.3.4 setLogDirectory() . . . 49
6.16 Matrices Class Reference . . . 49
6.16.1 Member Function Documentation . . . 50
6.16.1.1 gyroMatrix() . . . 50
6.16.1.2 massMatrix() . . . 50
6.16.1.3 OM_conj() . . . 50
6.16.1.4 quaterionsToRPY() . . . 51
6.16.1.5 R_nb() [1/2] . . . 51
6.16.1.6 R_nb() [2/2] . . . 51
6.16.1.7 R_wind_b() . . . 53

6.16.1.8 RPYtoQuaterion() . . . 53
6.16.1.9 TMatrix() . . . 54
6.17 ODE Class Reference . . . 54
6.17.1 Detailed Description . . . 55
6.17.2 Member Enumeration Documentation . . . 55
6.17.2.1 ODEMethod . . . 55
6.17.3 Constructor & Destructor Documentation . . . 55
6.17.3.1 ODE() . . . 55
6.17.3.2 ∼ODE() . . . 56
6.17.4 Member Function Documentation . . . 56
6.17.4.1 factory() . . . 56
6.17.4.2 fromString() . . . 56
6.17.4.3 getMicrosteps() [1/2] . . . 56
6.17.4.4 getMicrosteps() [2/2] . . . 57
6.17.4.5 step() . . . 57
6.18 ODE_Euler Class Reference . . . 58
6.18.1 Detailed Description . . . 58
6.18.2 Constructor & Destructor Documentation . . . 58
6.18.2.1 ODE_Euler() . . . 58
6.18.3 Member Function Documentation . . . 58
6.18.3.1 step() . . . 58
6.19 ODE_Heun Class Reference . . . 59
6.19.1 Detailed Description . . . 59
6.19.2 Constructor & Destructor Documentation . . . 59
6.19.2.1 ODE_Heun() . . . 60
6.19.3 Member Function Documentation . . . 60
6.19.3.1 step() . . . 60
6.20 ODE_RK4 Class Reference . . . 60
6.20.1 Detailed Description . . . 61
6.20.2 Constructor & Destructor Documentation . . . 61
6.20.2.1 ODE_RK4() . . . 61
6.20.3 Member Function Documentation . . . 61
6.20.3.1 step() . . . 61
6.21 ODETest Class Reference . . . 62
6.21.1 Member Function Documentation . . . 62
6.21.1.1 SetUp() . . . 62
6.21.1.2 TearDown() . . . 62
6.22 PID Class Reference . . . 62
6.22.1 Detailed Description . . . 63
6.22.2 Constructor & Destructor Documentation . . . 63
6.22.2.1 PID() . . . 63
6.22.2.2 ∼PID() . . . 63

6.22.3 Member Function Documentation . . . 63
6.22.3.1 calc() [1/2] . . . 63
6.22.3.2 calc() [2/2] . . . 64
6.22.3.3 clear() . . . 64
6.22.3.4 set_dt() . . . 64
6.23 Rotor Struct Reference . . . 65
6.23.1 Detailed Description . . . 65
6.23.2 Member Data Documentation . . . 65
6.23.2.1 direction . . . 65
6.23.2.2 forceCoff . . . 65
6.23.2.3 hoverSpeed . . . 66
6.23.2.4 maxSpeed . . . 66
6.23.2.5 timeConstant . . . 66
6.23.2.6 torqueCoff . . . 66
6.24 SensorParams Struct Reference . . . 66
6.24.1 Detailed Description . . . 66
6.24.2 Member Data Documentation . . . 67
6.24.2.1 bias . . . 67
6.24.2.2 name . . . 67
6.24.2.3 refreshTime . . . 67
6.24.2.4 sd . . . 67
6.25 Simulation Class Reference . . . 67
6.25.1 Constructor & Destructor Documentation . . . 68
6.25.1.1 Simulation() . . . 68
6.25.1.2 ~Simulation() . . . 68
6.25.2 Member Function Documentation . . . 68
6.25.2.1 run() . . . 68
6.26 TimedLoop Class Reference . . . 68
6.26.1 Detailed Description . . . 69
6.26.2 Constructor & Destructor Documentation . . . 69
6.26.2.1 TimedLoop() . . . 69
6.26.3 Member Function Documentation . . . 69
6.26.3.1 go() [1/2] . . . 69
6.26.3.2 go() [2/2] . . . 69
6.27 UAVparams Struct Reference . . . 70
6.27.1 Detailed Description . . . 71
6.27.2 Constructor & Destructor Documentation . . . 71
6.27.2.1 UAVparams() . . . 71
6.27.2.2 ~UAVparams() . . . 71
6.27.3 Member Function Documentation . . . 71
6.27.3.1 getRotorHoverSpeeds() . . . 71
6.27.3.2 getRotorMaxSpeeds() . . . 71

6.27.3.3 getRotorTimeContants() . . . 71
6.27.3.4 getSingleton() . . . 72
6.27.3.5 loadConfig() . . . 72
6.27.4 Member Data Documentation . . . 72
6.27.4.1 aero_coffs . . . 72
6.27.4.2 ahrs . . . 72
6.27.4.3 ammo . . . 72
6.27.4.4 cargo . . . 72
6.27.4.5 ekf . . . 72
6.27.4.6 initialMode . . . 73
6.27.4.7 initialOrientation . . . 73
6.27.4.8 initialPosition . . . 73
6.27.4.9 initialVelocity . . . 73
6.27.4.10 instantRun . . . 73
6.27.4.11 Ix . . . 73
6.27.4.12 Ixy . . . 73
6.27.4.13 Ixz . . . 73
6.27.4.14 Iy . . . 74
6.27.4.15 Iyz . . . 74
6.27.4.16 Iz . . . 74
6.27.4.17 jets . . . 74
6.27.4.18 m . . . 74
6.27.4.19 name . . . 74
6.27.4.20 noOfAmmo . . . 74
6.27.4.21 noOfCargo . . . 74
6.27.4.22 noOfJets . . . 75
6.27.4.23 noOfRotors . . . 75
6.27.4.24 pids . . . 75
6.27.4.25 rotorMixer . . . 75
6.27.4.26 rotors . . . 75
6.27.4.27 sensors . . . 75
6.27.4.28 surfaceMixer . . . 75
6.27.4.29 surfaces . . . 76
6.28 UAVstate Struct Reference . . . 76
6.28.1 Constructor & Destructor Documentation . . . 77
6.28.1.1 UAVstate() . . . 77
6.28.1.2 ~UAVstate() . . . 77
6.28.2 Member Function Documentation . . . 78
6.28.2.1 getAcceleration() . . . 78
6.28.2.2 getDemandedOm() . . . 78
6.28.2.3 getNoOfRotors() . . . 78
6.28.2.4 getOm() [1/2] . . . 78

6.28.2.5 getOm() [2/2] . . . 78
6.28.2.6 getOuterForce() . . . 79
6.28.2.7 getState() . . . 79
6.28.2.8 getX() [1/2] . . . 79
6.28.2.9 getX() [2/2] . . . 79
6.28.2.10 getY() [1/2] . . . 80
6.28.2.11 getY() [2/2] . . . 80
6.28.2.12 operator=() . . . 80
6.28.2.13 setAcceleration() . . . 81
6.28.2.14 setDemandedOm() . . . 81
6.28.2.15 setForce() . . . 81
6.28.2.16 setOm() . . . 81
6.28.2.17 setStatus() . . . 82
6.28.2.18 setX() [1/2] . . . 82
6.28.2.19 setX() [2/2] . . . 82
6.28.2.20 setY() . . . 83
6.28.3 Friends And Related Function Documentation . . . 83
6.28.3.1 operator<< . . . 83
6.28.4 Member Data Documentation . . . 83
6.28.4.1 real_time . . . 83
6.28.4.2 state_mtx . . . 83
6.28.4.3 status . . . 84
6.28.4.4 status_cv . . . 84

7 File Documentation **85**

7.1 lib/UAV_common/header/common.hpp File Reference . . . 85
7.2 lib/UAV_common/src/components/aero_coefficients.hpp File Reference . . . 85
7.3 lib/UAV_common/src/components/components.hpp File Reference . . . 85
7.4 lib/UAV_common/src/components/control_surfaces.cpp File Reference . . . 86
7.5 lib/UAV_common/src/components/control_surfaces.hpp File Reference . . . 86
7.6 lib/UAV_common/src/components/drive.cpp File Reference . . . 86
7.7 lib/UAV_common/src/components/drive.hpp File Reference . . . 86
7.8 lib/UAV_common/src/components/hinge.cpp File Reference . . . 86
7.8.1 Function Documentation . . . 87
7.8.1.1 asSkewMatrix() . . . 87
7.9 lib/UAV_common/src/components/hinge.hpp File Reference . . . 87
7.10 lib/UAV_common/src/components/loads.cpp File Reference . . . 87
7.11 lib/UAV_common/src/components/loads.hpp File Reference . . . 87
7.12 lib/UAV_common/src/components/navi.hpp File Reference . . . 88
7.13 lib/UAV_common/src/logger/logger.cpp File Reference . . . 88
7.13.1 Function Documentation . . . 88
7.13.1.1 shouldLog() . . . 88

7.14 lib/UAV_common/src/logger/logger.hpp File Reference . . . 88
7.14.1 Macro Definition Documentation . . . 89
7.14.1.1 LOGGER_MASK . . . 89
7.15 lib/UAV_common/src/ode/ode.cpp File Reference . . . 89
7.16 lib/UAV_common/src/ode/ode.hpp File Reference . . . 89
7.17 lib/UAV_common/src/ode/ode_impl.hpp File Reference . . . 89
7.18 lib/UAV_common/src/ode/ode_test.cpp File Reference . . . 90
7.18.1 Function Documentation . . . 90
7.18.1.1 getMethodsToTest() . . . 90
7.18.1.2 INSTANTIATE_TEST_SUITE_P() . . . 90
7.18.1.3 main() . . . 91
7.18.1.4 TEST_F() [1/2] . . . 91
7.18.1.5 TEST_F() [2/2] . . . 91
7.18.1.6 TEST_P() [1/3] . . . 91
7.18.1.7 TEST_P() [2/3] . . . 91
7.18.1.8 TEST_P() [3/3] . . . 91
7.19 lib/UAV_common/src/parser/parser.cpp File Reference . . . 92
7.19.1 Function Documentation . . . 92
7.19.1.1 parseMatrixXd() . . . 92
7.19.1.2 parseVectorXd() . . . 92
7.20 lib/UAV_common/src/parser/parser.hpp File Reference . . . 93
7.20.1 Function Documentation . . . 93
7.20.1.1 parseMatrixXd() . . . 93
7.20.1.2 parseVectorXd() . . . 94
7.21 lib/UAV_common/src/parser/uav_params.cpp File Reference . . . 94
7.21.1 Function Documentation . . . 94
7.21.1.1 parseHinge() . . . 94
7.21.1.2 parsePID() . . . 95
7.22 lib/UAV_common/src/parser/uav_params.hpp File Reference . . . 95
7.23 lib/UAV_common/src/PID/PID.cpp File Reference . . . 95
7.24 lib/UAV_common/src/PID/PID.hpp File Reference . . . 95
7.24.1 Enumeration Type Documentation . . . 95
7.24.1.1 AntiWindUpMode . . . 95
7.25 lib/UAV_common/src/timed_loop/status.hpp File Reference . . . 96
7.25.1 Enumeration Type Documentation . . . 96
7.25.1.1 Status . . . 96
7.26 lib/UAV_common/src/timed_loop/timed_loop.cpp File Reference . . . 96
7.27 lib/UAV_common/src/timed_loop/timed_loop.hpp File Reference . . . 97
7.28 src/aircraft/aircraft.cpp File Reference . . . 97
7.28.1 Function Documentation . . . 97
7.28.1.1 clampOrientationIfNessessery() . . . 97
7.29 src/aircraft/aircraft.hpp File Reference . . . 97

7.30 src/aircraft/aircraft_comm.cpp File Reference . . . 98
7.31 src/aircraft/aircraft_impulse.cpp File Reference . . . 98
7.32 src/defines.hpp File Reference . . . 98
7.32.1 Macro Definition Documentation . . . 99
7.32.1.1 USE_QUATERIONS . . . 99
7.33 src/dynamic/forces.cpp File Reference . . . 99
7.34 src/dynamic/forces.hpp File Reference . . . 99
7.35 src/dynamic/matrices.cpp File Reference . . . 99
7.36 src/dynamic/matrices.hpp File Reference . . . 99
7.37 src/main.cpp File Reference . . . 100
7.37.1 Function Documentation . . . 100
7.37.1.1 main() . . . 100
7.37.1.2 parseArgs() . . . 100
7.38 src/simulation/atmosphere.cpp File Reference . . . 101
7.39 src/simulation/atmosphere.hpp File Reference . . . 101
7.40 src/simulation/control.cpp File Reference . . . 101
7.40.1 Function Documentation . . . 102
7.40.1.1 control() . . . 102
7.40.1.2 controlListenerJob() . . . 102
7.40.1.3 dropCargo() . . . 102
7.40.1.4 isNormal() . . . 102
7.40.1.5 setAtmosphere() . . . 103
7.40.1.6 setControlSurface() . . . 103
7.40.1.7 setForce() . . . 103
7.40.1.8 setHinges() . . . 103
7.40.1.9 setSpeed() . . . 103
7.40.1.10 shoot() . . . 103
7.40.1.11 solidSurfColision() . . . 104
7.40.1.12 startJet() . . . 104
7.41 src/simulation/control.hpp File Reference . . . 104
7.41.1 Function Documentation . . . 104
7.41.1.1 controlListenerJob() . . . 104
7.42 src/simulation/simulation.cpp File Reference . . . 105
7.43 src/simulation/simulation.hpp File Reference . . . 105
7.44 src/simulation/uav_state.cpp File Reference . . . 105
7.44.1 Function Documentation . . . 105
7.44.1.1 operator<<() . . . 106
7.45 src/simulation/uav_state.hpp File Reference . . . 106
7.46 tests/test1.cpp File Reference . . . 106
7.46.1 Function Documentation . . . 106
7.46.1.1 fun() . . . 106
7.46.1.2 test1() . . . 107

7.47 tests/test2.cpp File Reference . . . 107
7.47.1 Function Documentation . . . 107
7.47.1.1 main() . . . 107
7.48 tests/test3.cpp File Reference . . . 107
7.48.1 Function Documentation . . . 107
7.48.1.1 fun() . . . 108
7.48.1.2 main() . . . 108
7.49 tests/test4.cpp File Reference . . . 108
7.49.1 Function Documentation . . . 108
7.49.1.1 main() . . . 108
7.50 tests/test5.cpp File Reference . . . 108
7.50.1 Function Documentation . . . 109
7.50.1.1 main() . . . 109
7.51 tests/zmq_recv.py File Reference . . . 109
7.52 tests/zmq_recv_last.py File Reference . . . 109
7.53 tests/zmq_send.py File Reference . . . 109
7.54 tests/zmq_send_tcp.py File Reference . . . 110

Index **111**

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

def
    Simulation constants 9
zmq_recv 11
zmq_recv_last 12
zmq_send 12
zmq_send_tcp 13

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

AeroCoefficients . . . 15
AHRSParams . . . 16
Aircraft . . . 18
Atmosphere . . . 28
AtmosphereInfo . . . 30
ControlSurfaces . . . 32
Drive . . . 34
  Jet . . . 42
  Rotor . . . 65
EKFScalers . . . 35
Forces . . . 36
Hinge . . . 40
Load . . . 45
  Ammo . . . 26
  Cargo . . . 32
Logger . . . 47
Matrices . . . 49
ODE . . . 54
  ODE_Euler . . . 58
  ODE_Heun . . . 59
  ODE_RK4 . . . 60
PID . . . 62
SensorParams . . . 66
Simulation . . . 67
testing::TestWithParam
  ODETest . . . 62
TimedLoop . . . 68
UAVparams . . . 70
UAVstate . . . 76

Chapter 3

Class Index

3.1 Class List

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

AeroCoefficients
: Aerodynamic coefficient . . . 15

AHRSParams
: AHRS parameters . . . 16

Aircraft
: Central class in simulation . . . 18

Ammo . . . 26

Atmosphere
: Representation of atmosphere where aircrafts fly . . . 28

AtmosphereInfo
: DTO containing atmosphere information . . . 30

Cargo . . . 32

ControlSurfaces
: Aircraft's control surfaces . . . 32

Drive
: Drive propelling aircraft . . . 34

EKFScalers
: Scalers for EKF . . . 35

Forces . . . 36

Hinge
: Hinge connecting aircraft with drives . . . 40

Jet
: Jet rocket engine . . . 42

Load
: Load of aircraft that can be droped or launched . . . 45

Logger
: Log vector data with timestamp in file . . . 47

Matrices . . . 49

ODE
: Ordinal differencial equation solver . . . 54

ODE_Euler
: Explicit Euler algorithm . . . 58

ODE_Heun
: Second order explicit Heun algorithm . . . 59

ODE_RK4
: Fourth order Runge Kutta algorithm . . . 60

ODETest 62
PID
PID discrete controller 62
Rotor
Rotor engine with controlled speed 65
SensorParams
Base parameters of a sensor 66
Simulation 67
TimedLoop
Simulation of real-time synchronized loop 68
UAVparams
Parsed UAV configuration from XML 70
UAVstate 76

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

lib/UAV_common/header/common.hpp . . . 85
lib/UAV_common/src/components/aero_coefficients.hpp . . . 85
lib/UAV_common/src/components/components.hpp . . . 85
lib/UAV_common/src/components/control_surfaces.cpp . . . 86
lib/UAV_common/src/components/control_surfaces.hpp . . . 86
lib/UAV_common/src/components/drive.cpp . . . 86
lib/UAV_common/src/components/drive.hpp . . . 86
lib/UAV_common/src/components/hinge.cpp . . . 86
lib/UAV_common/src/components/hinge.hpp . . . 87
lib/UAV_common/src/components/loads.cpp . . . 87
lib/UAV_common/src/components/loads.hpp . . . 87
lib/UAV_common/src/components/navi.hpp . . . 88
lib/UAV_common/src/logger/logger.cpp . . . 88
lib/UAV_common/src/logger/logger.hpp . . . 88
lib/UAV_common/src/ode/ode.cpp . . . 89
lib/UAV_common/src/ode/ode.hpp . . . 89
lib/UAV_common/src/ode/ode_impl.hpp . . . 89
lib/UAV_common/src/ode/ode_test.cpp . . . 90
lib/UAV_common/src/parser/parser.cpp . . . 92
lib/UAV_common/src/parser/parser.hpp . . . 93
lib/UAV_common/src/parser/uav_params.cpp . . . 94
lib/UAV_common/src/parser/uav_params.hpp . . . 95
lib/UAV_common/src/PID/PID.cpp . . . 95
lib/UAV_common/src/PID/PID.hpp . . . 95
lib/UAV_common/src/timed_loop/status.hpp . . . 96
lib/UAV_common/src/timed_loop/timed_loop.cpp . . . 96
lib/UAV_common/src/timed_loop/timed_loop.hpp . . . 97
src/defines.hpp . . . 98
src/main.cpp . . . 100
src/aircraft/aircraft.cpp . . . 97
src/aircraft/aircraft.hpp . . . 97
src/aircraft/aircraft_comm.cpp . . . 98
src/aircraft/aircraft_impulse.cpp . . . 98
src/dynamic/forces.cpp . . . 99
src/dynamic/forces.hpp . . . 99

src/dynamic/matrices.cpp . . . 99
src/dynamic/matrices.hpp . . . 99
src/simulation/atmosphere.cpp . . . 101
src/simulation/atmosphere.hpp . . . 101
src/simulation/control.cpp . . . 101
src/simulation/control.hpp . . . 104
src/simulation/simulation.cpp . . . 105
src/simulation/simulation.hpp . . . 105
src/simulation/uav_state.cpp . . . 105
src/simulation/uav_state.hpp . . . 106
tests/test1.cpp . . . 106
tests/test2.cpp . . . 107
tests/test3.cpp . . . 107
tests/test4.cpp . . . 108
tests/test5.cpp . . . 108
tests/zmq_recv.py . . . 109
tests/zmq_recv_last.py . . . 109
tests/zmq_send.py . . . 109
tests/zmq_send_tcp.py . . . 110

Chapter 5

Namespace Documentation

5.1 def Namespace Reference

Simulation constants.

Variables

- const double STEP_TIME = 0.001

  Step time of simulation. Step of ODE solving methods.
- 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 coefficient. Protect again diving objects in horizontal wall
- const double DOUBLE_EPS = 1e-5

  near zero floating point eps
- const double MIXING_FUNCTION = 0.1

  mixing window used in blending normal coefficients with standard ones, when stall angle was exceeded
- const int validityOfForce = 5

  how many times outer force should be used

5.1.1 Detailed Description

Simulation constants.

5.1.2 Variable Documentation

5.1.2.1 DOUBLE_EPS

```
const double def::DOUBLE_EPS = 1e-5
```

near zero floating point eps

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 coefficient. 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.

5.1.2.5 MIXING_FUNCTION

```
const double def::MIXING_FUNCTION = 0.1
```

mixing window used in blending normal coefficients with standard ones, when stall angle was exceeded

5.1.2.6 STEP_TIME

```
const double def::STEP_TIME = 0.001
```

Step time of simulation. Step of ODE solving methods.

5.1.2.7 validityOfForce

```
const int def::validityOfForce = 5
```

how many times outer force should be used

5.2 zmq_recv Namespace Reference

Variables

- context = zmq.Context()
- socket = context.socket(zmq.SUB)
- string topicfilter = "pos"
- s = socket.recv_string()

5.2.1 Variable Documentation

5.2.1.1 context

```
zmq_recv.context = zmq.Context()
```

5.2.1.2 s

```
zmq_recv.s = socket.recv_string()
```

5.2.1.3 socket

```
zmq_recv.socket = context.socket(zmq.SUB)
```

5.2.1.4 topicfilter

```
string zmq_recv.topicfilter = "pos"
```

5.3 zmq_recv_last Namespace Reference

Variables

- context = zmq.Context()
- socket = context.socket(zmq.SUB)
- string topicfilter = ""
- s = socket.recv_string()

5.3.1 Variable Documentation

5.3.1.1 context

```
zmq_recv_last.context = zmq.Context()
```

5.3.1.2 s

```
zmq_recv_last.s = socket.recv_string()
```

5.3.1.3 socket

```
zmq_recv_last.socket = context.socket(zmq.SUB)
```

5.3.1.4 topicfilter

```
string zmq_recv_last.topicfilter = ""
```

5.4 zmq_send Namespace Reference

Variables

- context = zmq.Context()
- socket = context.socket(zmq.PUB)
- int counter = 0

5.4.1 Variable Documentation

5.4.1.1 context

```
zmq_send.context = zmq.Context()
```

5.4.1.2 counter

```
int zmq_send.counter = 0
```

5.4.1.3 socket

```
zmq_send.socket = context.socket(zmq.PUB)
```

5.5 zmq_send_tcp Namespace Reference

Variables

- context = zmq.Context()
- socket = context.socket(zmq.PUB)
- float angle = 0.0

5.5.1 Variable Documentation

5.5.1.1 angle

```
float zmq_send_tcp.angle = 0.0
```

5.5.1.2 context

```
zmq_send_tcp.context = zmq.Context()
```

5.5.1.3 socket

```
zmq_send_tcp.socket = context.socket(zmq.PUB)
```

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 Aircraft Class Reference

central class in simulation

`#include <aircraft.hpp>`

Collaboration diagram for Aircraft:

Public Member Functions

- Aircraft ()

  Default constructor.
- virtual ~Aircraft ()

  Virtual deconstructor in case of future use in devired class.
- void update ()

  Simulation step.
- void sendState (zmq::socket_t *socket)

  Sends simulation state via publisher socket.
- bool startJet (int index)

  Starts jet engine.
- void trim ()

  Restore trim values of surface angles.
- bool setSurface (Eigen::VectorXd angles)

  Set surface deflation.
- bool setHinge (char type, int index, int hinge_index, double value)

  Set angle of specified hinge in specified drive.
- void calcImpulseForce (double COR, double mi_static, double mi_dynamic, Eigen::Vector3d collisionPoint, Eigen::Vector3d surfaceNormal)

  Calculate impact and result of collision with with solid surface. Results are applied to UAV state.
- std::tuple< int, Eigen::Vector3d > dropCargo (int index)

  Release cargo of specified index.
- std::tuple< int, Eigen::Vector3d > shootAmmo (int index)

  Shoot ammo of specified index.

Public Attributes

- UAVstate state

Protected Member Functions

- void reduceMass (double delta_m)

  Reduces mass of aircraft of given value. Mass matrix is reduced proportionally - moments of inertia is scaled as well.
- Eigen::Vector3d calcMomentumConservanceConservation (double m, Eigen::Vector3d speed, Eigen::↩ Vector3d r)

  Calculateds result of releasing/launching object from aircraft.
- virtual Eigen::VectorXd RHS (double, Eigen::VectorXd)

  Right hand side of main differential equation.

Protected Attributes

- Matrix< double, 6, 6 > massMatrix
- Matrix< double, 6, 6 > invMassMatrix
- std::mutex mtx
- int noOfRotors
- std::unique_ptr< Rotor[ ]> rotors
- int noOfJets
- std::unique_ptr< Jet[ ]> jets
- ControlSurfaces surfaces
- AeroCoefficients aero
- int noOfAmmo
- std::unique_ptr< Ammo[ ]> ammo
- int noOfCargo
- std::unique_ptr< Cargo[ ]> cargo
- std::unique_ptr< ODE > ode

6.3.1 Detailed Description

central class in simulation

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Aircraft()

```
Aircraft::Aircraft ( )
```

Default constructor.

6.3.2.2 ~Aircraft()

```
virtual Aircraft::~Aircraft ( )  [inline], [virtual]
```

Virtual deconstructor in case of future use in devired class.

6.3.3 Member Function Documentation

6.3.3.1 calcImpulseForce()

```
void Aircraft::calcImpulseForce (
            double COR,
            double mi_static,
            double mi_dynamic,
            Eigen::Vector3d collisionPoint,
            Eigen::Vector3d surfaceNormal )
```

Calculate impact and result of collision with with solid surface. Results are applied to UAV state.

Parameters

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
collisionPoint	point of collision
surfaceNormal	surface normal vector

6.3.3.2 calcMomentumConservanceConservation()

```
Eigen::Vector3d Aircraft::calcMomentumConservanceConservation (
            double m,
            Eigen::Vector3d speed,
            Eigen::Vector3d r )  [protected]
```

Calculateds result of releasing/launching object from aircraft.

Parameters

m	object mass
speed	object's initial velocity vector in body frame
r	offset of object

Returns

initial linear velocity of object in world frame

6.3.3.3 dropCargo()

```
std::tuple< int, Eigen::Vector3d > Aircraft::dropCargo (
            int index )
```

Release cargo of specified index.

Parameters

index	index of cargo

Returns

Returns pair of result and velocity of released cargo in world frame. Result is number of cargo of given type that left on board. Result also informs about fails:

- -1 - cooldown, next drop is not ready
- -2 - out of cargos
- -10 - index not found

6.3.3.4 reduceMass()

```
void Aircraft::reduceMass (
            double delta_m )  [protected]
```

Reduces mass of aircraft of given value. Mass matrix is reduced proportionally - moments of inertia is scaled as well.

Parameters

delta_m	mass reduction

6.3.3.5 RHS()

```
Eigen::VectorXd Aircraft::RHS (
            double time,
            Eigen::VectorXd local_state )  [protected], [virtual]
```

Right hand side of main differential equation.

Parameters

time	time of simulation
state	state vector

Returns

derivative of state

6.3.3.6 sendState()

```
void Aircraft::sendState (
            zmq::socket_t * socket )
```

Sends simulation state via publisher socket.

Parameters

socket	zmq socket to send simulation state

6.3.3.7 setHinge()

```
bool Aircraft::setHinge (
            char type,
            int index,
            int hinge_index,
            double value )
```

Set angle of specified hinge in specified drive.

Parameters

type	type of drive: 'r' - rotor, 'j' - jet
index	index of drive
hinge_index	index of hinde
value	new angle value

Returns

true, if angle was set. Returns false if specified drive or hinge wasn't found

6.3.3.8 setSurface()

```
bool Aircraft::setSurface (
            Eigen::VectorXd angles )
```

Set surface deflation.

Parameters

angles	vector of new surface angles

Returns

true if angles were set. Returns false if length of vector is not equal to numbers of surfaces

6.3.3.9 shootAmmo()

```
std::tuple< int, Eigen::Vector3d > Aircraft::shootAmmo (
            int index )
```

Shoot ammo of specified index.

Parameters

index	index of ammo

Returns

Returns pair of result and velocity of released cargo in world frame. Result is number of ammo of given type that left on board. Result also informs about fails:

- -1 - cooldown, next shoot is not ready
- -2 - out of ammo
- -10 - index not found

6.3.3.10 startJet()

```
bool Aircraft::startJet (
            int index )
```

Starts jet engine.

Parameters

index	index of engine to start

Returns

return true if jet was started. False if jet is running or burnt out

6.3.3.11 trim()

```
void Aircraft::trim ( )
```

Restore trim values of surface angles.

6.3.3.12 update()

```
void Aircraft::update ( )
```

Simulation step.

6.3.4 Member Data Documentation

6.3.4.1 aero

```
AeroCoefficients Aircraft::aero  [protected]
```

6.3.4.2 ammo

```
std::unique_ptr<Ammo[]> Aircraft::ammo  [protected]
```

6.3.4.3 cargo

```
std::unique_ptr<Cargo[]> Aircraft::cargo  [protected]
```

6.3.4.4 invMassMatrix

```
Matrix<double,6,6> Aircraft::invMassMatrix  [protected]
```

6.3.4.5 jets

```
std::unique_ptr<Jet[]> Aircraft::jets  [protected]
```

6.3.4.6 massMatrix

```
Matrix<double,6,6> Aircraft::massMatrix  [protected]
```

6.3.4.7 mtx

```
std::mutex Aircraft::mtx  [protected]
```

6.3.4.8 noOfAmmo

```
int Aircraft::noOfAmmo  [protected]
```

6.3.4.9 noOfCargo

```
int Aircraft::noOfCargo  [protected]
```

6.3.4.10 noOfJets

```
int Aircraft::noOfJets  [protected]
```

6.3.4.11 noOfRotors

```
int Aircraft::noOfRotors  [protected]
```

6.3.4.12 ode

```
std::unique_ptr<ODE> Aircraft::ode  [protected]
```

6.3.4.13 rotors

```
std::unique_ptr<Rotor[]> Aircraft::rotors  [protected]
```

6.3.4.14 state

```
UAVstate Aircraft::state
```

6.3.4.15 surfaces

```
ControlSurfaces Aircraft::surfaces  [protected]
```

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

- src/aircraft/aircraft.hpp
- src/aircraft/aircraft.cpp
- src/aircraft/aircraft_comm.cpp
- src/aircraft/aircraft_impulse.cpp

6.4 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

6.4.1 Constructor & Destructor Documentation

6.4.1.1 Ammo() [1/2]

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

6.4.1.2 Ammo() [2/2]

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

6.4.2 Member Function Documentation

6.4.2.1 getV0()

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

get start velocity of ammo when launched

Returns

start velocity vector

6.4.2.2 operator=()

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

6.4.3 Member Data Documentation

6.4.3.1 _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

6.5 Atmosphere Class Reference

Representation of atmosphere where aircrafts fly.

`#include <atmosphere.hpp>`

Public Member Functions

- Atmosphere ()
  Default constructor.
- ~Atmosphere ()
  Default deconstructor.
- Eigen::Vector3d getWind ()
  Returns wind speed vector in world frame.
- double getAirTemperature ()
  Returns air temperature.
- double getAirPressure ()
  Returns air pressure.
- double getAirDensity ()
  Returns air density.
- void update (AtmosphereInfo info)
  Update atmosphere status.

Static Public Member Functions

- static Atmosphere * getSingleton ()
  Returns pointer to singleton of atmosphere.

6.5.1 Detailed Description

Representation of atmosphere where aircrafts fly.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 Atmosphere()

`Atmosphere::Atmosphere ( )`

Default constructor.

6.5.2.2 ~Atmosphere()

```
Atmosphere::~Atmosphere ( )
```

Default deconstructor.

6.5.3 Member Function Documentation

6.5.3.1 getAirDensity()

```
double Atmosphere::getAirDensity ( )
```

Returns air density.

Returns

air density kg/m3

6.5.3.2 getAirPressure()

```
double Atmosphere::getAirPressure ( )
```

Returns air pressure.

Returns

air pressure Pa

6.5.3.3 getAirTemperature()

```
double Atmosphere::getAirTemperature ( )
```

Returns air temperature.

Returns

air temperature K

6.5.3.4 getSingleton()

```
Atmosphere * Atmosphere::getSingleton ( )  [static]
```

Returns pointer to singleton of atmosphere.

Returns

pointer to Atmosphere instance. Nullptr if singleton not exists

6.5.3.5 getWind()

```
Eigen::Vector3d Atmosphere::getWind ( )
```

Returns wind speed vector in world frame.

Returns

wind speed vector m/s

6.5.3.6 update()

```
void Atmosphere::update (
            AtmosphereInfo info )
```

Update atmosphere status.

Parameters

info	dto with new atmosphere info

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

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

6.6 AtmosphereInfo Struct Reference

DTO containing atmosphere information.

```
#include <atmosphere.hpp>
```

Public Attributes

- Eigen::Vector3d wind = Eigen::Vector3d(0.0,0.0,0.0)
- double air_temperature = 288.15
- double air_pressure = 101300.0
- double air_density = 1.224

6.6.1 Detailed Description

DTO containing atmosphere information.

6.6.2 Member Data Documentation

6.6.2.1 air_density

```
double AtmosphereInfo::air_density = 1.224
```

6.6.2.2 air_pressure

```
double AtmosphereInfo::air_pressure = 101300.0
```

6.6.2.3 air_temperature

```
double AtmosphereInfo::air_temperature = 288.15
```

6.6.2.4 wind

```
Eigen::Vector3d AtmosphereInfo::wind = Eigen::Vector3d(0.0,0.0,0.0)
```

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

- src/simulation/atmosphere.hpp

6.7 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

6.7.1 Constructor & Destructor Documentation

6.7.1.1 Cargo() [1/2]

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

6.7.1.2 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

6.8 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

6.8.1 Detailed Description

Aircraft's control surfaces.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 ControlSurfaces() [1/2]

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

6.8.2.2 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

6.8.3 Member Function Documentation

6.8.3.1 getCoefficients()

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

6.8.3.2 getNoOfSurface()

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

6.8.3.3 getValues()

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

6.8.3.4 restoreTrim()

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

6.8.3.5 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

6.9 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]

6.9.1 Detailed Description

Drive propelling aircraft.

6.9.2 Member Data Documentation

6.9.2.1 axis

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

6.9.2.2 hinges

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

6.9.2.3 noOfHinges

```
int Drive::noOfHinges
```

6.9.2.4 position

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

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

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

6.10 EKFScalers Struct Reference

Scalers for EKF.

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

Public Attributes

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

6.10.1 Detailed Description

Scalers for EKF.

6.10.2 Member Data Documentation

6.10.2.1 baroScaler

```
double EKFScalers::baroScaler
```

6.10.2.2 predictScaler

```
double EKFScalers::predictScaler
```

6.10.2.3 updateScaler

```
double EKFScalers::updateScaler
```

6.10.2.4 zScaler

```
double EKFScalers::zScaler
```

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

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

6.11 Forces Class Reference

```
#include <forces.hpp>
```

Static Public Member Functions

- static Vector< double, 6 > gravity_loads (const Matrix3d &r_nb)

  Calculates gravity loads acting on UAV.
- static Vector< double, 6 > rotor_lift_loads (int noOfRotors, Rotor *rotors, VectorXd rotorAngularVelocity)

  Calculates loads generated by rotors.
- static Vector< double, 6 > jet_lift_loads (int noOfJets, Jet *jets, double time)

  Calculates loads generated by jet.
- static Vector< double, 6 > aerodynamic_loads (const Vector< double, 6 > &x, Vector3d wind_body, const ControlSurfaces &surface, const AeroCoefficients &aero, double height)

  Calculates aerodynamic loads.
- static VectorXd angularAcceleration (VectorXd demandedAngularVelocity, VectorXd rotorAngularVelocity)

  Calculates acceleration of propellers.
- static void generateCharacteristics (const ControlSurfaces &surface, const AeroCoefficients &aero)

  Generates aerodynamics characteristics and save in csv files.

6.11.1 Member Function Documentation

6.11.1.1 aerodynamic_loads()

```
Vector< double, 6 > Forces::aerodynamic_loads (
            const Vector< double, 6 > & x,
            Vector3d wind_body,
            const ControlSurfaces & surface,
            const AeroCoefficients & aero,
            double height )  [static]
```

Calculates aerodynamic loads.

Parameters

x	vector of UAV velocities
wind_body	vector of wind acting on UAV
surface	reference to ControlSurfaces instance
aero	reference to AeroCoefficients instance
height	absolute height about sea (AMSL)

Returns

loads in body frame

6.11.1.2 angularAcceleration()

```
VectorXd Forces::angularAcceleration (
            VectorXd demandedAngularVelocity,
            VectorXd rotorAngularVelocity )  [static]
```

Calculates acceleration of propellers.

Parameters

demandedAngularVelocity	vector of demanded angular velocities
rotorAngularVelocity	vector of actual angular velocities

Returns

vector of angular accelerations

6.11.1.3 generateCharacteristics()

```
void Forces::generateCharacteristics (
            const ControlSurfaces & surface,
            const AeroCoefficients & aero )  [static]
```

Generates aerodynamics characteristics and save in csv files.

Parameters

surface	reference to ControlSurfaces instance
aero	reference to AeroCoefficients instance

6.11.1.4 gravity_loads()

```
Vector< double, 6 > Forces::gravity_loads (
            const Matrix3d & r_nb )  [static]
```

Calculates gravity loads acting on UAV.

Parameters

r_nb	rotation matrix from world to body frame

Returns

gravity load in body frame

6.11.1.5 jet_lift_loads()

```
Vector< double, 6 > Forces::jet_lift_loads (
            int noOfJets,
```

```
            Jet * jets,
            double time )  [static]
```

Calculates loads generated by jet.

Parameters

noOfJets	numbers of jets
jets	pointer to jet instance
time	simulation time

Returns

loads in body frame

6.11.1.6 rotor_lift_loads()

```
Vector< double, 6 > Forces::rotor_lift_loads (
            int noOfRotors,
            Rotor * rotors,
            VectorXd rotorAngularVelocity )  [static]
```

Calculates loads generated by rotors.

Parameters

noOfRotors	numbers of rotors
rotors	pointer to rotor instance
rotorAngularVelocity	vector of angular velocities of rotors

Returns

loads in body frame

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

- src/dynamic/forces.hpp
- src/dynamic/forces.cpp

6.12 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.

6.12.1 Detailed Description

Hinge connecting aircraft with drives.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 Hinge() [1/3]

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

6.12.2.2 Hinge() [2/3]

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

6.12.2.3 Hinge() [3/3]

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

6.12.3 Member Function Documentation

6.12.3.1 getRot()

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

Get rotattion matrix of orientation change due to hinge.

Returns

rotation matrix

6.12.3.2 operator=()

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

6.12.3.3 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

6.13 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

6.13.1 Detailed Description

Jet rocket engine.

6.13.2 Member Function Documentation

6.13.2.1 getLastThrust()

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

get last calculated thrust

Returns

last calculated thrust

6.13.2.2 getThrust()

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

get thrust in specific time

Parameters

time	timestamp

Returns

thrust value in Newtons

6.13.2.3 start()

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

start jet engine

Parameters

time	timestamp of start

Returns

true if start succesful, false if already started

6.13.3 Member Data Documentation

6.13.3.1 phases

```
int Jet::phases
```

6.13.3.2 thrust

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

6.13.3.3 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

6.14 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)

6.14.1 Detailed Description

Load of aircraft that can be droped or launched.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 Load() [1/2]

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

6.14.2.2 Load() [2/2]

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

6.14.3 Member Function Documentation

6.14.3.1 getMass()

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

get mass of load

Returns

mass

6.14.3.2 getOffset()

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

get offset of load

Returns

offset vector

6.14.3.3 operator=()

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

6.14.3.4 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

6.15 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.

6.15.1 Detailed Description

Log vector data with timestamp in file.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 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

6.15.2.2 ~Logger()

```
Logger::~Logger ( )
```

Deconstructor.

6.15.3 Member Function Documentation

6.15.3.1 log() [1/2]

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

Log one row.

Parameters

time	timestamp
args	list of doubles

6.15.3.2 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

6.15.3.3 setFmt()

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

Set new format if was not known in constructor.

Parameters

fmt	new format

6.15.3.4 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

6.16 Matrices Class Reference

`#include <matrices.hpp>`

Static Public Member Functions

- static Matrix< double, 6, 6 > massMatrix ()

  Constucts initial mass matrix, based on parameters.
- static Matrix< double, 6, 6 > gyroMatrix (Vector< double, 6 > x)

  Calculates gyroscopic matrix.
- static Matrix< double, 6, 6 > TMatrix (Vector< double, 6 > y)

  Calculates transformation matrix from body to world frame for velocities.
- static Matrix< double, 3, 3 > R_nb (const Vector< double, 6 > &y)

  Calculates rotation matrix from world to body frame.
- static Matrix< double, 3, 3 > R_nb (const Vector< double, 7 > &y)

Calculates rotation matrix from world to body frame.

- static Matrix< double, 3, 3 > R_wind_b (double alpha, double beta)

  Calculates rotation matrix from wind to body frame.
- static Vector< double, 6 > quaterionsToRPY (Vector< double, 7 > y)

  Convert position and orientation vector from quaterions to RPY Euler angles.
- static Vector< double, 7 > RPYtoQuaterion (Vector< double, 6 > y)

  Convert position and orientation vector from RPY Euler angles to quaterions.
- static Matrix4d OM_conj (Vector< double, 6 > x)

  Calculates conjugation matrix for angular velocity.

6.16.1 Member Function Documentation

6.16.1.1 gyroMatrix()

```
Matrix< double, 6, 6 > Matrices::gyroMatrix (
            Vector< double, 6 > x )  [static]
```

Calculates gyroscopic matrix.

Parameters

x	velocity vector

Returns

gyroscopic matrix

6.16.1.2 massMatrix()

```
Matrix< double, 6, 6 > Matrices::massMatrix ( )  [static]
```

Constucts initial mass matrix, based on parameters.

Returns

mass matrix

6.16.1.3 OM_conj()

```
Matrix4d Matrices::OM_conj (
            Vector< double, 6 > x )  [static]
```

Calculates conjugation matrix for angular velocity.

Parameters

x	vector of velocites

Returns

conjugation matrix

6.16.1.4 quaterionsToRPY()

```
Vector< double, 6 > Matrices::quaterionsToRPY (
            Vector< double, 7 > y )  [static]
```

Convert position and orientation vector from quaterions to RPY Euler angles.

Parameters

y	position & orientation vector

Returns

position & orientation vector. Orientation is given in RPY

6.16.1.5 R_nb() [1/2]

```
Matrix< double, 3, 3 > Matrices::R_nb (
            const Vector< double, 6 > & y )  [static]
```

Calculates rotation matrix from world to body frame.

Parameters

y	actual position & orietation vector (RPY)

Returns

rotation matrix

6.16.1.6 R_nb() [2/2]

```
Matrix< double, 3, 3 > Matrices::R_nb (
            const Vector< double, 7 > & y )  [static]
```

Calculates rotation matrix from world to body frame.

Parameters

y	actual position & orietation vector (quaterions)

Returns

rotation matrix

6.16.1.7 R_wind_b()

```
Matrix< double, 3, 3 > Matrices::R_wind_b (
            double alpha,
            double beta )  [static]
```

Calculates rotation matrix from wind to body frame.

Parameters

alpha	angle of attack
beta	angle of slide

Returns

rotation matrix

6.16.1.8 RPYtoQuaterion()

```
Vector< double, 7 > Matrices::RPYtoQuaterion (
            Vector< double, 6 > y )  [static]
```

Convert position and orientation vector from RPY Euler angles to quaterions.

Parameters

y	position & orientation vector

Returns

position & orientation vector. Orientation is given in quaterions

6.16.1.9 TMatrix()

```
Matrix< double, 6, 6 > Matrices::TMatrix (
            Vector< double, 6 > y )  [static]
```

Calculates transformation matrix from body to world frame for velocities.

Parameters

y	actual position vector

Returns

transformation matrix

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

- src/dynamic/matrices.hpp
- src/dynamic/matrices.cpp

6.17 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.

6.17.1 Detailed Description

Ordinal differencial equation solver.

6.17.2 Member Enumeration Documentation

6.17.2.1 ODEMethod

```
enum ODE::ODEMethod
```

Supported solving method.

Enumerator

Euler	
Heun	
RK4	
NONE	

6.17.3 Constructor & Destructor Documentation

6.17.3.1 ODE()

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

Constructor.

6.17.3.2 ~ODE()

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

Virtual deconstructor.

6.17.4 Member Function Documentation

6.17.4.1 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

6.17.4.2 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

6.17.4.3 getMicrosteps() [1/2]

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

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

Returns

microsteps

6.17.4.4 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

6.17.4.5 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

6.18 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::Vector↩ Xd)> rhs_fun, double h) override

  One step of explicit solving algorithm.

Additional Inherited Members

6.18.1 Detailed Description

Explicit Euler algorithm.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 ODE_Euler()

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

6.18.3 Member Function Documentation

6.18.3.1 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

6.19 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::Vector↩ Xd)> rhs_fun, double h) override

  One step of explicit solving algorithm.

Additional Inherited Members

6.19.1 Detailed Description

Second order explicit Heun algorithm.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 ODE_Heun()

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

6.19.3 Member Function Documentation

6.19.3.1 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

6.20 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::Vector↩ Xd)> rhs_fun, double h) override

  One step of explicit solving algorithm.

Additional Inherited Members

6.20.1 Detailed Description

Fourth order Runge Kutta algorithm.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 ODE_RK4()

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

6.20.3 Member Function Documentation

6.20.3.1 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

6.21 ODETest Class Reference

Inheritance diagram for ODETest:

Collaboration diagram for ODETest:

Protected Member Functions

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

6.21.1 Member Function Documentation

6.21.1.1 SetUp()

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

6.21.1.2 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

6.22 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

6.22.1 Detailed Description

PID discrete controller.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 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 )
```

6.22.2.2 ~PID()

```
PID::~PID ( )
```

6.22.3 Member Function Documentation

6.22.3.1 calc() [1/2]

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

calc output of controller

Parameters

error	input of controller

Returns

output of controller

6.22.3.2 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

6.22.3.3 clear()

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

clear internal state

6.22.3.4 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

6.23 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

6.23.1 Detailed Description

Rotor engine with controlled speed.

6.23.2 Member Data Documentation

6.23.2.1 direction

```
int Rotor::direction
```

6.23.2.2 forceCoff

```
double Rotor::forceCoff
```

6.23.2.3 hoverSpeed

```
double Rotor::hoverSpeed
```

6.23.2.4 maxSpeed

```
double Rotor::maxSpeed
```

6.23.2.5 timeConstant

```
double Rotor::timeConstant
```

6.23.2.6 torqueCoff

```
double Rotor::torqueCoff
```

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

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

6.24 SensorParams Struct Reference

Base parameters of a sensor.

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

Public Attributes

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

6.24.1 Detailed Description

Base parameters of a sensor.

6.24.2 Member Data Documentation

6.24.2.1 bias

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

6.24.2.2 name

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

6.24.2.3 refreshTime

```
double SensorParams::refreshTime
```

6.24.2.4 sd

```
double SensorParams::sd
```

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

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

6.25 Simulation Class Reference

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

Public Member Functions

- Simulation ()
  Default constructor.
- ~Simulation ()
  Deconstructor.
- void run ()
  Run simulation.

6.25.1 Constructor & Destructor Documentation

6.25.1.1 Simulation()

```
Simulation::Simulation ( )
```

Default constructor.

6.25.1.2 ~Simulation()

```
Simulation::~Simulation ( )
```

Deconstructor.

6.25.2 Member Function Documentation

6.25.2.1 run()

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

Run simulation.

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

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

6.26 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

6.26.1 Detailed Description

Simulation of real-time synchronized loop.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 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

6.26.3 Member Function Documentation

6.26.3.1 go() [1/2]

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

start infinite loop

6.26.3.2 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

6.27 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

6.27.1 Detailed Description

Parsed UAV configuration from XML.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 UAVparams()

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

Initialize default data.

6.27.2.2 ~UAVparams()

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

6.27.3 Member Function Documentation

6.27.3.1 getRotorHoverSpeeds()

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

6.27.3.2 getRotorMaxSpeeds()

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

6.27.3.3 getRotorTimeContants()

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

6.27.3.4 getSingleton()

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

6.27.3.5 loadConfig()

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

6.27.4 Member Data Documentation

6.27.4.1 aero_coffs

```
AeroCoefficients UAVparams::aero_coffs
```

6.27.4.2 ahrs

```
AHRSParams UAVparams::ahrs
```

6.27.4.3 ammo

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

6.27.4.4 cargo

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

6.27.4.5 ekf

```
EKFScalers UAVparams::ekf
```

6.27.4.6 initialMode

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

6.27.4.7 initialOrientation

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

6.27.4.8 initialPosition

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

6.27.4.9 initialVelocity

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

6.27.4.10 instantRun

```
bool UAVparams::instantRun
```

6.27.4.11 Ix

```
double UAVparams::Ix
```

6.27.4.12 Ixy

```
double UAVparams::Ixy
```

6.27.4.13 Ixz

```
double UAVparams::Ixz
```

6.27.4.14 Iy

```
double UAVparams::Iy
```

6.27.4.15 Iyz

```
double UAVparams::Iyz
```

6.27.4.16 Iz

```
double UAVparams::Iz
```

6.27.4.17 jets

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

6.27.4.18 m

```
double UAVparams::m
```

6.27.4.19 name

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

6.27.4.20 noOfAmmo

```
int UAVparams::noOfAmmo
```

6.27.4.21 noOfCargo

```
int UAVparams::noOfCargo
```

6.27.4.22 noOfJets

```
int UAVparams::noOfJets
```

6.27.4.23 noOfRotors

```
int UAVparams::noOfRotors
```

6.27.4.24 pids

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

6.27.4.25 rotorMixer

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

6.27.4.26 rotors

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

6.27.4.27 sensors

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

6.27.4.28 surfaceMixer

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

6.27.4.29 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

6.28 UAVstate Struct Reference

```
#include <uav_state.hpp>
```

Public Member Functions

- UAVstate ()
  Default constructor.
- ~UAVstate ()
  Deconstructor.
- Eigen::Vector< double, 7 > getY ()
  Returns position vector Y from state (quaterions)
- Eigen::Vector< double, 6 > getX ()
  Returns velocity vector X.
- Eigen::VectorXd getOm ()
  Returns rotor's angular velocities vector.
- Eigen::VectorXd getDemandedOm ()
  Returns rotor's demanded angular velocities vector.
- Eigen::Vector< double, 6 > getOuterForce ()
  Returns outer force applied to aircraft.
- Eigen::VectorXd getState ()
  Returns raw state vector.
- int getNoOfRotors ()
  Returns number of rotors.
- Eigen::Vector< double, 6 > getAcceleration ()
  Returns aircraft acceleration.
- void setX (Eigen::Vector< double, 6 > newX)
  Set velocity vector X.
- void setDemandedOm (Eigen::VectorXd newOm)
  Set demanded angular velocity vector.
- void setForce (Eigen::Vector3d force, Eigen::Vector3d torque=Eigen::Vector3d(0.0, 0.0, 0.0))
  Set outer load.
- void setAcceleration (Eigen::Vector< double, 6 > newAccel)
  Set acceleration vector.
- UAVstate & operator= (Eigen::VectorXd &other)
  Assigns given raw state vector to state.
- void setStatus (Status newStatus)
  Sets new timed loop status.

Static Public Member Functions

- static void setY (Eigen::VectorXd &state, Eigen::Vector< double, 7 > Y)
  Sets position vector Y in given state (quaterions)
- static Eigen::Vector< double, 7 > getY (const Eigen::VectorXd &state)
  Returns position vector Y from given state (quaterions)
- static void setX (Eigen::VectorXd &state, Eigen::Vector< double, 6 > X)
  Set velocity vector X in given state.
- static void setOm (Eigen::VectorXd &state, Eigen::VectorXd Om)
  Set angular velocity vector in given state.
- static Eigen::Vector< double, 6 > getX (const Eigen::VectorXd &state)
  Returns velocity vector X from given state.
- static Eigen::VectorXd getOm (const Eigen::VectorXd &state)
  Return angular velocity vector from given state.

Public Attributes

- std::atomic< double > real_time
  simulation time
- std::mutex state_mtx
  state mutex
- Status status
  Timed loop status.
- std::condition_variable status_cv

Friends

- std::ostream & operator<< (std::ostream &outs, const UAVstate &state)
  Serializes state to stream.

6.28.1 Constructor & Destructor Documentation

6.28.1.1 UAVstate()

```
UAVstate::UAVstate ( )
```

Default constructor.

6.28.1.2 ~UAVstate()

```
UAVstate::~UAVstate ( )
```

Deconstructor.

6.28.2 Member Function Documentation

6.28.2.1 getAcceleration()

```
Eigen::Vector<double,6> UAVstate::getAcceleration ( )  [inline]
```

Returns aircraft acceleration.

Returns

acceleraton vector

6.28.2.2 getDemandedOm()

```
Eigen::VectorXd UAVstate::getDemandedOm ( )
```

Returns rotor's demanded angular velocities vector.

Returns

rotor's demanded angular velocities vector rad/s

6.28.2.3 getNoOfRotors()

```
int UAVstate::getNoOfRotors ( )  [inline]
```

Returns number of rotors.

Returns

number of rotors

6.28.2.4 getOm() [1/2]

```
Eigen::VectorXd UAVstate::getOm ( )
```

Returns rotor's angular velocities vector.

Returns

rotor's angular velocities vector

6.28.2.5 getOm() [2/2]

```
Eigen::VectorXd UAVstate::getOm (
            const Eigen::VectorXd & state )  [static]
```

Return angular velocity vector from given state.

Parameters

state	source state

Returns

angular velocity vector

6.28.2.6 getOuterForce()

```
Eigen::Vector< double, 6 > UAVstate::getOuterForce ( )
```

Returns outer force applied to aircraft.

Returns

outer force N

6.28.2.7 getState()

```
Eigen::VectorXd UAVstate::getState ( )
```

Returns raw state vector.

Returns

state vector

6.28.2.8 getX() [1/2]

```
Eigen::Vector< double, 6 > UAVstate::getX ( )
```

Returns velocity vector X.

Returns

velocity vector X

6.28.2.9 getX() [2/2]

```
Eigen::Vector< double, 6 > UAVstate::getX (
            const Eigen::VectorXd & state )  [static]
```

Returns velocity vector X from given state.

Parameters

state	source state

Returns

velocity vector X

6.28.2.10 getY() [1/2]

```
Eigen::Vector< double, 7 > UAVstate::getY ( )
```

Returns position vector Y from state (quaterions)

Returns

position vector Y

6.28.2.11 getY() [2/2]

```
Eigen::Vector< double, 7 > UAVstate::getY (
            const Eigen::VectorXd & state )  [static]
```

Returns position vector Y from given state (quaterions)

Parameters

state	source state

Returns

position vector Y

6.28.2.12 operator=()

```
UAVstate & UAVstate::operator= (
            Eigen::VectorXd & other )
```

Assigns given raw state vector to state.

6.28.2.13 setAcceleration()

```
void UAVstate::setAcceleration (
            Eigen::Vector< double, 6 > newAccel )
```

Set acceleration vector.

Parameters

newAccel	new acceleration vector

6.28.2.14 setDemandedOm()

```
void UAVstate::setDemandedOm (
            Eigen::VectorXd newOm )
```

Set demanded angular velocity vector.

Parameters

newOm	new demanded angular velocity vector

6.28.2.15 setForce()

```
void UAVstate::setForce (
            Eigen::Vector3d force,
            Eigen::Vector3d torque = Eigen::Vector3d(0.0,0.0,0.0) )
```

Set outer load.

Parameters

force	applied force
torque	applied torque

6.28.2.16 setOm()

```
void UAVstate::setOm (
            Eigen::VectorXd & state,
            Eigen::VectorXd Om )  [static]
```

Set angular velocity vector in given state.

Parameters

state	state that should be updated
Om	new angular velocity vector

6.28.2.17 setStatus()

```
void UAVstate::setStatus (
            Status newStatus )  [inline]
```

Sets new timed loop status.

Parameters

newStatus	new status

6.28.2.18 setX() [1/2]

```
void UAVstate::setX (
            Eigen::Vector< double, 6 > newX )
```

Set velocity vector X.

Parameters

newX	new velocity vector X

6.28.2.19 setX() [2/2]

```
void UAVstate::setX (
            Eigen::VectorXd & state,
            Eigen::Vector< double, 6 > X )  [static]
```

Set velocity vector X in given state.

Parameters

state	state that should be updated
X	new velocity vector X

6.28.2.20 setY()

```
void UAVstate::setY (
            Eigen::VectorXd & state,
            Eigen::Vector< double, 7 > Y )  [static]
```

Sets position vector Y in given state (quaterions)

Parameters

state	state that should be updated
Y	new position vector

6.28.3 Friends And Related Function Documentation

6.28.3.1 operator<<

```
std::ostream& operator<< (
            std::ostream & outs,
            const UAVstate & state )  [friend]
```

Serializes state to stream.

6.28.4 Member Data Documentation

6.28.4.1 real_time

```
std::atomic<double> UAVstate::real_time
```

simulation time

6.28.4.2 state_mtx

```
std::mutex UAVstate::state_mtx
```

state mutex

6.28.4.3 status

```
Status UAVstate::status
```

Timed loop status.

6.28.4.4 status_cv

```
std::condition_variable UAVstate::status_cv
```

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

- src/simulation/uav_state.hpp
- src/simulation/uav_state.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/aircraft/aircraft.cpp File Reference

```
#include "aircraft.hpp"
```

Include dependency graph for aircraft.cpp:

Functions

- void clampOrientationIfNessessery ([[maybe_unused]] Eigen::VectorXd &state)

7.28.1 Function Documentation

7.28.1.1 clampOrientationIfNessessery()

```
void clampOrientationIfNessessery (
            [[maybe_unused] ] Eigen::VectorXd & state )
```

7.29 src/aircraft/aircraft.hpp File Reference

```
#include <Eigen/Dense>
#include <memory>
#include <mutex>
#include <zmq.hpp>
#include <iostream>
#include "common.hpp"
#include "../defines.hpp"
#include "../dynamic/forces.hpp"
#include "../dynamic/matrices.hpp"
#include "../simulation/atmosphere.hpp"
#include "../simulation/uav_state.hpp"
```

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

Classes

- class Aircraft

  central class in simulation

7.30 src/aircraft/aircraft_comm.cpp File Reference

`#include "aircraft.hpp"`
Include dependency graph for aircraft_comm.cpp:

7.31 src/aircraft/aircraft_impulse.cpp File Reference

`#include "aircraft.hpp"`
Include dependency graph for aircraft_impulse.cpp:

7.32 src/defines.hpp File Reference

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

Namespaces

- def

  Simulation constants.

Macros

- #define USE_QUATERIONS 1

  define to use quaterion instead of RPY angles

Variables

- const double def::STEP_TIME = 0.001

  Step time of simulation. Step of ODE solving methods.
- 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 coefficient. Protect again diving objects in horizontal wall
- const double def::DOUBLE_EPS = 1e-5

  near zero floating point eps
- const double def::MIXING_FUNCTION = 0.1

  mixing window used in blending normal coefficients with standard ones, when stall angle was exceeded
- const int def::validityOfForce = 5

  how many times outer force should be used

7.32.1 Macro Definition Documentation

7.32.1.1 USE_QUATERIONS

```
#define USE_QUATERIONS 1
```

define to use quaterion instead of RPY angles

7.33 src/dynamic/forces.cpp File Reference

```
#include <Eigen/Dense>
#include <cmath>
#include <iostream>
#include <numbers>
#include "forces.hpp"
#include "matrices.hpp"
#include "../defines.hpp"
#include "../simulation/atmosphere.hpp"
#include "common.hpp"
```

Include dependency graph for forces.cpp:

7.34 src/dynamic/forces.hpp File Reference

```
#include <Eigen/Dense>
#include "common.hpp"
#include "matrices.hpp"
```

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

Classes

- class Forces

7.35 src/dynamic/matrices.cpp File Reference

```
#include <Eigen/Dense>
#include <cmath>
#include "matrices.hpp"
#include "common.hpp"
```

Include dependency graph for matrices.cpp:

7.36 src/dynamic/matrices.hpp File Reference

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

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

Classes

- class Matrices

7.37 src/main.cpp File Reference

```
#include <iostream>
#include <cxxopts.hpp>
#include "simulation/simulation.hpp"
#include "simulation/uav_state.hpp"
#include "dynamic/forces.hpp"
#include "common.hpp"
```

Include dependency graph for main.cpp:

Functions

- void parseArgs (int argc, char **argv, UAVparams *params)

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

7.37.1 Function Documentation

7.37.1.1 main()

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

7.37.1.2 parseArgs()

```
void parseArgs (
            int argc,
            char ** argv,
            UAVparams * params )
```

Parse CL arguments.

Parameters

argc	number of argument
argv	argument array
params	pointer to UAVparams instant that should be filled

7.38 src/simulation/atmosphere.cpp File Reference

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

Include dependency graph for atmosphere.cpp:

7.39 src/simulation/atmosphere.hpp File Reference

```
#include <Eigen/Dense>
#include <atomic>
#include "common.hpp"
```

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

Classes

- struct AtmosphereInfo

  DTO containing atmosphere information.
- class Atmosphere

  Representation of atmosphere where aircrafts fly.

7.40 src/simulation/control.cpp File Reference

```
#include <Eigen/Dense>
#include <zmq.hpp>
#include <sstream>
#include <iostream>
#include "control.hpp"
#include "uav_state.hpp"
#include "common.hpp"
#include "../dynamic/matrices.hpp"
#include "../aircraft/aircraft.hpp"
#include "../defines.hpp"
#include "atmosphere.hpp"
```

Include dependency graph for control.cpp:

Functions

- void setAtmosphere (std::string &msg_str, zmq::socket_t &sock)
- void setForce (Aircraft *aircraft, std::string &msg_str, zmq::socket_t &sock)
- void setSpeed (Aircraft *aircraft, std::string &msg_str, zmq::socket_t &sock)
- bool control (Aircraft *aircraft, std::string &msg_str, zmq::socket_t &sock)
- void shoot (Aircraft *aircraft, std::string &msg_str, zmq::socket_t &sock)
- void dropCargo (Aircraft *aircraft, std::string &msg_str, zmq::socket_t &sock)
- bool isNormal (double factor)
- void solidSurfColision (Aircraft *aircraft, std::string &msg_str, zmq::socket_t &sock)
- void setControlSurface (Aircraft *aircraft, std::string &msg_str, zmq::socket_t &sock)
- void startJet (Aircraft *aircraft, std::string &msg_str, zmq::socket_t &sock)
- void setHinges (Aircraft *aircraft, std::string &msg_str, zmq::socket_t &sock)
- void controlListenerJob (zmq::context_t *ctx, std::string address, Aircraft *aircraft)

  Job of control listener thread. Listen for new control command and handle them.

7.40.1 Function Documentation

7.40.1.1 control()

```
bool control (
            Aircraft * aircraft,
            std::string & msg_str,
            zmq::socket_t & sock )
```

7.40.1.2 controlListenerJob()

```
void controlListenerJob (
            zmq::context_t * ctx,
            std::string address,
            Aircraft * aircraft )
```

Job of control listener thread. Listen for new control command and handle them.

Parameters

ctx	zero mq context
address	address of listener socket
aircraft	pointer to aircraft

7.40.1.3 dropCargo()

```
void dropCargo (
            Aircraft * aircraft,
            std::string & msg_str,
            zmq::socket_t & sock )
```

7.40.1.4 isNormal()

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

7.40.1.5 setAtmosphere()

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

7.40.1.6 setControlSurface()

```
void setControlSurface (
            Aircraft * aircraft,
            std::string & msg_str,
            zmq::socket_t & sock )
```

7.40.1.7 setForce()

```
void setForce (
            Aircraft * aircraft,
            std::string & msg_str,
            zmq::socket_t & sock )
```

7.40.1.8 setHinges()

```
void setHinges (
            Aircraft * aircraft,
            std::string & msg_str,
            zmq::socket_t & sock )
```

7.40.1.9 setSpeed()

```
void setSpeed (
            Aircraft * aircraft,
            std::string & msg_str,
            zmq::socket_t & sock )
```

7.40.1.10 shoot()

```
void shoot (
            Aircraft * aircraft,
            std::string & msg_str,
            zmq::socket_t & sock )
```

7.40.1.11 solidSurfColision()

```
void solidSurfColision (
            Aircraft * aircraft,
            std::string & msg_str,
            zmq::socket_t & sock )
```

7.40.1.12 startJet()

```
void startJet (
            Aircraft * aircraft,
            std::string & msg_str,
            zmq::socket_t & sock )
```

7.41 src/simulation/control.hpp File Reference

```
#include <zmq.hpp>
#include "uav_state.hpp"
#include "../dynamic/matrices.hpp"
#include "../defines.hpp"
#include "../aircraft/aircraft.hpp"
```

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

Functions

- void controlListenerJob (zmq::context_t *ctx, std::string address, Aircraft *aircraft)

  Job of control listener thread. Listen for new control command and handle them.

7.41.1 Function Documentation

7.41.1.1 controlListenerJob()

```
void controlListenerJob (
            zmq::context_t * ctx,
            std::string address,
            Aircraft * aircraft )
```

Job of control listener thread. Listen for new control command and handle them.

Parameters

ctx	zero mq context
address	address of listener socket
aircraft	pointer to aircraft

7.42 src/simulation/simulation.cpp File Reference

```
#include <Eigen/Dense>
#include <zmq.hpp>
#include <iostream>
#include <cstdio>
#include <thread>
#include <mutex>
#include <filesystem>
#include <chrono>
#include "simulation.hpp"
#include "uav_state.hpp"
#include "common.hpp"
#include "control.hpp"
#include "../defines.hpp"
```

Include dependency graph for simulation.cpp:

7.43 src/simulation/simulation.hpp File Reference

```
#include <zmq.hpp>
#include <memory>
#include "common.hpp"
#include "uav_state.hpp"
#include "../dynamic/forces.hpp"
#include "../dynamic/matrices.hpp"
#include "../aircraft/aircraft.hpp"
#include "atmosphere.hpp"
```

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

Classes

- class Simulation

7.44 src/simulation/uav_state.cpp File Reference

```
#include "uav_state.hpp"
#include "../dynamic/matrices.hpp"
```

Include dependency graph for uav_state.cpp:

Functions

- std::ostream & operator<< (std::ostream &outs, const UAVstate &state)

7.44.1 Function Documentation

7.44.1.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & outs,
            const UAVstate & state )
```

7.45 src/simulation/uav_state.hpp File Reference

```
#include <Eigen/Dense>
#include <atomic>
#include <condition_variable>
#include <mutex>
#include "common.hpp"
#include "../defines.hpp"
```

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

Classes

- struct UAVstate

7.46 tests/test1.cpp File Reference

```
#include <iostream>
#include <Eigen/Dense>
#include "../src/RK4.hpp"
```

Include dependency graph for test1.cpp:

Functions

- Eigen::Vector2d fun (double t, Eigen::Vector2d y)
- int test1 ()

7.46.1 Function Documentation

7.46.1.1 fun()

```
Eigen::Vector2d fun (
            double t,
            Eigen::Vector2d y )
```

7.46.1.2 test1()

```
int test1 ( )
```

7.47 tests/test2.cpp File Reference

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

Include dependency graph for test2.cpp:

Functions

- int main ()

7.47.1 Function Documentation

7.47.1.1 main()

```
int main ( )
```

7.48 tests/test3.cpp File Reference

```
#include <iostream>
#include <Eigen/Dense>
#include "constants.hpp"
#include "forces.hpp"
#include "RK4.hpp"
#include "common.hpp"
```

Include dependency graph for test3.cpp:

Functions

- Eigen::Vector4d fun (double t, Eigen::Vector4d om)
- int main ()

7.48.1 Function Documentation

7.48.1.1 fun()

```
Eigen::Vector4d fun (
            double t,
            Eigen::Vector4d om )
```

7.48.1.2 main()

```
int main ( )
```

7.49 tests/test4.cpp File Reference

```
#include <iostream>
#include <Eigen/Dense>
#include "common.hpp"
#include "uav_state.hpp"
```

Include dependency graph for test4.cpp:

Functions

- int main ()

7.49.1 Function Documentation

7.49.1.1 main()

```
int main ( )
```

7.50 tests/test5.cpp File Reference

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

Include dependency graph for test5.cpp:

Functions

- int main ()

7.50.1 Function Documentation

7.50.1.1 main()

```
int main ( )
```

7.51 tests/zmq_recv.py File Reference

Namespaces

- zmq_recv

Variables

- zmq_recv.context = zmq.Context()
- zmq_recv.socket = context.socket(zmq.SUB)
- string zmq_recv.topicfilter = "pos"
- zmq_recv.s = socket.recv_string()

7.52 tests/zmq_recv_last.py File Reference

Namespaces

- zmq_recv_last

Variables

- zmq_recv_last.context = zmq.Context()
- zmq_recv_last.socket = context.socket(zmq.SUB)
- string zmq_recv_last.topicfilter = ""
- zmq_recv_last.s = socket.recv_string()

7.53 tests/zmq_send.py File Reference

Namespaces

- zmq_send

Variables

- zmq_send.context = zmq.Context()
- zmq_send.socket = context.socket(zmq.PUB)
- int zmq_send.counter = 0

7.54 tests/zmq_send_tcp.py File Reference

Namespaces

- zmq_send_tcp

Variables

- zmq_send_tcp.context = zmq.Context()
- zmq_send_tcp.socket = context.socket(zmq.PUB)
- float zmq_send_tcp.angle = 0.0

Index

_V0
    Ammo, 27
~Aircraft
    Aircraft, 19
~Atmosphere
    Atmosphere, 28
~Logger
    Logger, 48
~ODE
    ODE, 55
~PID
    PID, 63
~Simulation
    Simulation, 68
~UAVparams
    UAVparams, 71
~UAVstate
    UAVstate, 77

aero
    Aircraft, 24
aero_coffs
    UAVparams, 72
AeroCoefficients, 15
    C0, 15
    Cab, 15
    Cpqr, 16
    d, 16
    eAR, 16
    S, 16
    stallLimit, 16
aerodynamic_loads
    Forces, 37
ahrs
    UAVparams, 72
AHRSParams, 16
    alpha, 17
    Q, 17
    R, 17
    type, 17
air_density
    AtmosphereInfo, 31
air_pressure
    AtmosphereInfo, 31
air_temperature
    AtmosphereInfo, 31
Aircraft, 18
    ~Aircraft, 19
    aero, 24
    Aircraft, 19
    ammo, 24
    calcImpulseForce, 19
    calcMomentumConservanceConservation, 20
    cargo, 24
    dropCargo, 20
    invMassMatrix, 24
    jets, 24
    massMatrix, 24
    mtx, 24
    noOfAmmo, 25
    noOfCargo, 25
    noOfJets, 25
    noOfRotors, 25
    ode, 25
    reduceMass, 21
    RHS, 21
    rotors, 25
    sendState, 21
    setHinge, 22
    setSurface, 22
    shootAmmo, 22
    startJet, 23
    state, 25
    surfaces, 25
    trim, 23
    update, 23
aircraft.cpp
    clampOrientationIfNessessery, 97
alpha
    AHRSParams, 17
Ammo, 26
    _V0, 27
    Ammo, 26
    getV0, 27
    operator=, 27
ammo
    Aircraft, 24
    UAVparams, 72
angle
    zmq_send_tcp, 13
angularAcceleration
    Forces, 37
AntiWindUpMode
    PID.hpp, 95
asSkewMatrix
    hinge.cpp, 87
Atmosphere, 28
    ~Atmosphere, 28
    Atmosphere, 28

getAirDensity, 29
getAirPressure, 29
getAirTemperature, 29
getSingleton, 29
getWind, 30
update, 30
AtmosphereInfo, 30
air_density, 31
air_pressure, 31
air_temperature, 31
wind, 31
axis
Drive, 35

baroScaler
EKFScalers, 36
bias
SensorParams, 67

C0
AeroCoefficients, 15
Cab
AeroCoefficients, 15
calc
PID, 63, 64
calcImpulseForce
Aircraft, 19
calcMomentumConservanceConservation
Aircraft, 20
Cargo, 32
Cargo, 32
cargo
Aircraft, 24
UAVparams, 72
Clamping
PID.hpp, 96
clampOrientationIfNessessery
aircraft.cpp, 97
clear
PID, 64
context
zmq_recv, 11
zmq_recv_last, 12
zmq_send, 13
zmq_send_tcp, 13
control
control.cpp, 102
control.cpp
control, 102
controlListenerJob, 102
dropCargo, 102
isNormal, 102
setAtmosphere, 102
setControlSurface, 103
setForce, 103
setHinges, 103
setSpeed, 103
shoot, 103
solidSurfColision, 103
startJet, 104
control.hpp
controlListenerJob, 104
controlListenerJob
control.cpp, 102
control.hpp, 104
ControlSurfaces, 32
ControlSurfaces, 33
getCoefficients, 34
getNoOfSurface, 34
getValues, 34
restoreTrim, 34
setValues, 34
counter
zmq_send, 13
Cpqr
AeroCoefficients, 16

d
AeroCoefficients, 16
def, 9
DOUBLE_EPS, 9
FRICTION_EPS, 10
GENTLY_PUSH, 10
GRAVITY_CONST, 10
MIXING_FUNCTION, 10
STEP_TIME, 10
validityOfForce, 10
defines.hpp
USE_QUATERIONS, 99
direction
Rotor, 65
DOUBLE_EPS
def, 9
Drive, 34
axis, 35
hinges, 35
noOfHinges, 35
position, 35
dropCargo
Aircraft, 20
control.cpp, 102

eAR
AeroCoefficients, 16
ekf
UAVparams, 72
EKFScalers, 35
baroScaler, 36
predictScaler, 36
updateScaler, 36
zScaler, 36
Euler
ODE, 55
exiting
status.hpp, 96

factory
ODE, 56

forceCoff
Rotor, 65
Forces, 36
aerodynamic_loads, 37
angularAcceleration, 37
generateCharacteristics, 39
gravity_loads, 39
jet_lift_loads, 39
rotor_lift_loads, 40
FRICTION_EPS
def, 10
fromString
ODE, 56
fun
test1.cpp, 106
test3.cpp, 107

generateCharacteristics
Forces, 39
GENTLY_PUSH
def, 10
getAcceleration
UAVstate, 78
getAirDensity
Atmosphere, 29
getAirPressure
Atmosphere, 29
getAirTemperature
Atmosphere, 29
getCoefficients
ControlSurfaces, 34
getDemandedOm
UAVstate, 78
getLastThrust
Jet, 43
getMass
Load, 46
getMethodsToTest
ode_test.cpp, 90
getMicrosteps
ODE, 56, 57
getNoOfRotors
UAVstate, 78
getNoOfSurface
ControlSurfaces, 34
getOffset
Load, 46
getOm
UAVstate, 78
getOuterForce
UAVstate, 79
getRot
Hinge, 41
getRotorHoverSpeeds
UAVparams, 71
getRotorMaxSpeeds
UAVparams, 71
getRotorTimeContants
UAVparams, 71
getSingleton
Atmosphere, 29
UAVparams, 71
getState
UAVstate, 79
getThrust
Jet, 43
getV0
Ammo, 27
getValues
ControlSurfaces, 34
getWind
Atmosphere, 30
getX
UAVstate, 79
getY
UAVstate, 80
go
TimedLoop, 69
GRAVITY_CONST
def, 10
gravity_loads
Forces, 39
gyroMatrix
Matrices, 50

Heun
ODE, 55
Hinge, 40
getRot, 41
Hinge, 41
operator=, 42
updateValue, 42
hinge.cpp
asSkewMatrix, 87
hinges
Drive, 35
hoverSpeed
Rotor, 65

idle
status.hpp, 96
initialMode
UAVparams, 72
initialOrientation
UAVparams, 73
initialPosition
UAVparams, 73
initialVelocity
UAVparams, 73
INSTANTIATE_TEST_SUITE_P
ode_test.cpp, 90
instantRun
UAVparams, 73
invMassMatrix
Aircraft, 24
isNormal
control.cpp, 102
Ix

UAVparams, 73
Ixy
UAVparams, 73
Ixz
UAVparams, 73
Iy
UAVparams, 73
Iyz
UAVparams, 74
Iz
UAVparams, 74

Jet, 42
getLastThrust, 43
getThrust, 43
phases, 44
start, 44
thrust, 44
time, 44
jet_lift_loads
Forces, 39
jets
Aircraft, 24
UAVparams, 74

lib/UAV_common/header/common.hpp, 85
lib/UAV_common/src/components/aero_coefficients.hpp, 85
lib/UAV_common/src/components/components.hpp, 85
lib/UAV_common/src/components/control_surfaces.cpp, 86
lib/UAV_common/src/components/control_surfaces.hpp, 86
lib/UAV_common/src/components/drive.cpp, 86
lib/UAV_common/src/components/drive.hpp, 86
lib/UAV_common/src/components/hinge.cpp, 86
lib/UAV_common/src/components/hinge.hpp, 87
lib/UAV_common/src/components/loads.cpp, 87
lib/UAV_common/src/components/loads.hpp, 87
lib/UAV_common/src/components/navi.hpp, 88
lib/UAV_common/src/logger/logger.cpp, 88
lib/UAV_common/src/logger/logger.hpp, 88
lib/UAV_common/src/ode/ode.cpp, 89
lib/UAV_common/src/ode/ode.hpp, 89
lib/UAV_common/src/ode/ode_impl.hpp, 89
lib/UAV_common/src/ode/ode_test.cpp, 90
lib/UAV_common/src/parser/parser.cpp, 92
lib/UAV_common/src/parser/parser.hpp, 93
lib/UAV_common/src/parser/uav_params.cpp, 94
lib/UAV_common/src/parser/uav_params.hpp, 95
lib/UAV_common/src/PID/PID.cpp, 95
lib/UAV_common/src/PID/PID.hpp, 95
lib/UAV_common/src/timed_loop/status.hpp, 96
lib/UAV_common/src/timed_loop/timed_loop.cpp, 96
lib/UAV_common/src/timed_loop/timed_loop.hpp, 97
Load, 45
getMass, 46
getOffset, 46
Load, 45
operator=, 46
release, 46
loadConfig
UAVparams, 72
log
Logger, 48
Logger, 47
∼Logger, 48
log, 48
Logger, 47
setFmt, 49
setLogDirectory, 49
logger.cpp
shouldLog, 88
logger.hpp
LOGGER_MASK, 89
LOGGER_MASK
logger.hpp, 89

m
UAVparams, 74
main
main.cpp, 100
ode_test.cpp, 90
test2.cpp, 107
test3.cpp, 108
test4.cpp, 108
test5.cpp, 109
main.cpp
main, 100
parseArgs, 100
massMatrix
Aircraft, 24
Matrices, 50
Matrices, 49
gyroMatrix, 50
massMatrix, 50
OM_conj, 50
quaterionsToRPY, 51
R_nb, 51
R_wind_b, 53
RPYtoQuaterion, 53
TMatrix, 53
maxSpeed
Rotor, 66
MIXING_FUNCTION
def, 10
mtx
Aircraft, 24

name
SensorParams, 67
UAVparams, 74
NONE
ODE, 55
None
PID.hpp, 96
noOfAmmo
Aircraft, 25

UAVparams, 74
noOfCargo
Aircraft, 25
UAVparams, 74
noOfHinges
Drive, 35
noOfJets
Aircraft, 25
UAVparams, 74
noOfRotors
Aircraft, 25
UAVparams, 75

ODE, 54
∼ODE, 55
Euler, 55
factory, 56
fromString, 56
getMicrosteps, 56, 57
Heun, 55
NONE, 55
ODE, 55
ODEMethod, 55
RK4, 55
step, 57
ode
Aircraft, 25
ODE_Euler, 58
ODE_Euler, 58
step, 58
ODE_Heun, 59
ODE_Heun, 59
step, 60
ODE_RK4, 60
ODE_RK4, 61
step, 61
ode_test.cpp
getMethodsToTest, 90
INSTANTIATE_TEST_SUITE_P, 90
main, 90
TEST_F, 91
TEST_P, 91
ODEMethod
ODE, 55
ODETest, 62
SetUp, 62
TearDown, 62
OM_conj
Matrices, 50
operator<<
uav_state.cpp, 105
UAVstate, 83
operator=
Ammo, 27
Hinge, 42
Load, 46
UAVstate, 80

parseArgs
main.cpp, 100
parseHinge
uav_params.cpp, 94
parseMatrixXd
parser.cpp, 92
parser.hpp, 93
parsePID
uav_params.cpp, 94
parser.cpp
parseMatrixXd, 92
parseVectorXd, 92
parser.hpp
parseMatrixXd, 93
parseVectorXd, 94
parseVectorXd
parser.cpp, 92
parser.hpp, 94
phases
Jet, 44
PID, 62
∼PID, 63
calc, 63, 64
clear, 64
PID, 63
set_dt, 64
PID.hpp
AntiWindUpMode, 95
Clamping, 96
None, 96
pids
UAVparams, 75
position
Drive, 35
predictScaler
EKFScalers, 36

Q
AHRSParams, 17
quaterionsToRPY
Matrices, 51

R
AHRSParams, 17
R_nb
Matrices, 51
R_wind_b
Matrices, 53
real_time
UAVstate, 83
reduceMass
Aircraft, 21
refreshTime
SensorParams, 67
release
Load, 46
reload
status.hpp, 96
restoreTrim
ControlSurfaces, 34

RHS
Aircraft, 21
RK4
ODE, 55
Rotor, 65
direction, 65
forceCoff, 65
hoverSpeed, 65
maxSpeed, 66
timeConstant, 66
torqueCoff, 66
rotor_lift_loads
Forces, 40
rotorMixer
UAVparams, 75
rotors
Aircraft, 25
UAVparams, 75
RPYtoQuaterion
Matrices, 53
run
Simulation, 68
running
status.hpp, 96

S
AeroCoefficients, 16
s
zmq_recv, 11
zmq_recv_last, 12
sd
SensorParams, 67
sendState
Aircraft, 21
SensorParams, 66
bias, 67
name, 67
refreshTime, 67
sd, 67
sensors
UAVparams, 75
set_dt
PID, 64
setAcceleration
UAVstate, 80
setAtmosphere
control.cpp, 102
setControlSurface
control.cpp, 103
setDemandedOm
UAVstate, 81
setFmt
Logger, 49
setForce
control.cpp, 103
UAVstate, 81
setHinge
Aircraft, 22
setHinges
control.cpp, 103
setLogDirectory
Logger, 49
setOm
UAVstate, 81
setSpeed
control.cpp, 103
setStatus
UAVstate, 82
setSurface
Aircraft, 22
SetUp
ODETest, 62
setValues
ControlSurfaces, 34
setX
UAVstate, 82
setY
UAVstate, 82
shoot
control.cpp, 103
shootAmmo
Aircraft, 22
shouldLog
logger.cpp, 88
Simulation, 67
∼Simulation, 68
run, 68
Simulation, 68
socket
zmq_recv, 11
zmq_recv_last, 12
zmq_send, 13
zmq_send_tcp, 13
solidSurfColision
control.cpp, 103
src/aircraft/aircraft.cpp, 97
src/aircraft/aircraft.hpp, 97
src/aircraft/aircraft_comm.cpp, 98
src/aircraft/aircraft_impulse.cpp, 98
src/defines.hpp, 98
src/dynamic/forces.cpp, 99
src/dynamic/forces.hpp, 99
src/dynamic/matrices.cpp, 99
src/dynamic/matrices.hpp, 99
src/main.cpp, 100
src/simulation/atmosphere.cpp, 101
src/simulation/atmosphere.hpp, 101
src/simulation/control.cpp, 101
src/simulation/control.hpp, 104
src/simulation/simulation.cpp, 105
src/simulation/simulation.hpp, 105
src/simulation/uav_state.cpp, 105
src/simulation/uav_state.hpp, 106
stallLimit
AeroCoefficients, 16
start
Jet, 44

startJet
Aircraft, 23
control.cpp, 104
state
Aircraft, 25
state_mtx
UAVstate, 83
Status
status.hpp, 96
status
UAVstate, 83
status.hpp
exiting, 96
idle, 96
reload, 96
running, 96
Status, 96
status_cv
UAVstate, 84
step
ODE, 57
ODE_Euler, 58
ODE_Heun, 60
ODE_RK4, 61
STEP_TIME
def, 10
surfaceMixer
UAVparams, 75
surfaces
Aircraft, 25
UAVparams, 75

TearDown
ODETest, 62
test1
test1.cpp, 106
test1.cpp
fun, 106
test1, 106
test2.cpp
main, 107
test3.cpp
fun, 107
main, 108
test4.cpp
main, 108
test5.cpp
main, 109
TEST_F
ode_test.cpp, 91
TEST_P
ode_test.cpp, 91
tests/test1.cpp, 106
tests/test2.cpp, 107
tests/test3.cpp, 107
tests/test4.cpp, 108
tests/test5.cpp, 108
tests/zmq_recv.py, 109
tests/zmq_recv_last.py, 109
tests/zmq_send.py, 109
tests/zmq_send_tcp.py, 110
thrust
Jet, 44
time
Jet, 44
timeConstant
Rotor, 66
TimedLoop, 68
go, 69
TimedLoop, 69
TMatrix
Matrices, 53
topicfilter
zmq_recv, 11
zmq_recv_last, 12
torqueCoff
Rotor, 66
trim
Aircraft, 23
type
AHRSParams, 17

uav_params.cpp
parseHinge, 94
parsePID, 94
uav_state.cpp
operator<<, 105
UAVparams, 70
~UAVparams, 71
aero_coffs, 72
ahrs, 72
ammo, 72
cargo, 72
ekf, 72
getRotorHoverSpeeds, 71
getRotorMaxSpeeds, 71
getRotorTimeContants, 71
getSingleton, 71
initialMode, 72
initialOrientation, 73
initialPosition, 73
initialVelocity, 73
instantRun, 73
Ix, 73
Ixy, 73
Ixz, 73
Iy, 73
Iyz, 74
Iz, 74
jets, 74
loadConfig, 72
m, 74
name, 74
noOfAmmo, 74
noOfCargo, 74
noOfJets, 74
noOfRotors, 75
pids, 75

rotorMixer, 75
rotors, 75
sensors, 75
surfaceMixer, 75
surfaces, 75
UAVparams, 71
UAVstate, 76
~UAVstate, 77
getAcceleration, 78
getDemandedOm, 78
getNoOfRotors, 78
getOm, 78
getOuterForce, 79
getState, 79
getX, 79
getY, 80
operator<<, 83
operator=, 80
real_time, 83
setAcceleration, 80
setDemandedOm, 81
setForce, 81
setOm, 81
setStatus, 82
setX, 82
setY, 82
state_mtx, 83
status, 83
status_cv, 84
UAVstate, 77
update
Aircraft, 23
Atmosphere, 30
updateScaler
EKFScalers, 36
updateValue
Hinge, 42
USE_QUATERIONS
defines.hpp, 99

validityOfForce
def, 10

wind
AtmosphereInfo, 31

zmq_recv, 11
context, 11
s, 11
socket, 11
topicfilter, 11
zmq_recv_last, 12
context, 12
s, 12
socket, 12
topicfilter, 12
zmq_send, 12
context, 13
counter, 13
socket, 13
zmq_send_tcp, 13
angle, 13
context, 13
socket, 13
zScaler
EKFScalers, 36