

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	11
5.1 controllers Namespace Reference	11
5.2 def Namespace Reference	11
5.2.1 Detailed Description	11
5.2.2 Variable Documentation	12
5.2.2.1 DOUBLE_EPS	12
5.2.2.2 FRICTION_EPS	12
5.2.2.3 GENTLY_PUSH	12
5.2.2.4 GRAVITY_CONST	12
5.2.2.5 MIXING_FUNCTION	12
5.2.2.6 validityOfForce	12
5.3 zmq_recv Namespace Reference	13
5.3.1 Variable Documentation	13
5.3.1.1 context	13
5.3.1.2 s	13
5.3.1.3 socket	13
5.3.1.4 topicfilter	13
5.4 zmq_recv_last Namespace Reference	13
5.4.1 Variable Documentation	14
5.4.1.1 context	14
5.4.1.2 s	14
5.4.1.3 socket	14
5.4.1.4 topicfilter	14
5.5 zmq_send Namespace Reference	14
5.5.1 Variable Documentation	14
5.5.1.1 context	14
5.5.1.2 counter	15
5.5.1.3 socket	15
5.6 zmq_send_tcp Namespace Reference	15
5.6.1 Variable Documentation	15
5.6.1.1 angle	15
5.6.1.2 context	15

5.6.1.3 socket	15
6 Class Documentation	17
6.1 AeroCoefficients Struct Reference	17
6.1.1 Detailed Description	17
6.1.2 Member Data Documentation	17
6.1.2.1 C0	17
6.1.2.2 Cab	18
6.1.2.3 Cpqr	18
6.1.2.4 d	18
6.1.2.5 eAR	18
6.1.2.6 S	18
6.1.2.7 stallLimit	18
6.2 AHRSParams Struct Reference	18
6.2.1 Detailed Description	19
6.2.2 Member Data Documentation	19
6.2.2.1 alpha	19
6.2.2.2 Q	19
6.2.2.3 R	19
6.2.2.4 type	19
6.3 Aircraft Class Reference	20
6.3.1 Detailed Description	21
6.3.2 Constructor & Destructor Documentation	21
6.3.2.1 Aircraft()	21
6.3.2.2 ~Aircraft()	21
6.3.3 Member Function Documentation	21
6.3.3.1 calcImpulseForce()	21
6.3.3.2 calcMomentumConservanceConservation()	22
6.3.3.3 dropCargo()	22
6.3.3.4 reduceMass()	23
6.3.3.5 RHS()	23
6.3.3.6 sendState()	23
6.3.3.7 setHinge()	24
6.3.3.8 setSurface()	24
6.3.3.9 shootAmmo()	25
6.3.3.10 startJet()	25
6.3.3.11 trim()	25
6.3.3.12 update()	26
6.3.4 Member Data Documentation	26
6.3.4.1 aero	26
6.3.4.2 ammo	26
6.3.4.3 cargo	26

6.3.4.4 invMassMatrix	26
6.3.4.5 jets	26
6.3.4.6 massMatrix	26
6.3.4.7 mtx	27
6.3.4.8 noOfAmmo	27
6.3.4.9 noOfCargo	27
6.3.4.10 noOfJets	27
6.3.4.11 noOfRotors	27
6.3.4.12 ode	27
6.3.4.13 rotors	27
6.3.4.14 state	27
6.3.4.15 surfaces	28
6.4 Ammo Class Reference	28
6.4.1 Constructor & Destructor Documentation	28
6.4.1.1 Ammo() [1/2]	28
6.4.1.2 Ammo() [2/2]	29
6.4.2 Member Function Documentation	29
6.4.2.1 getV0()	29
6.4.2.2 operator=()	29
6.4.3 Member Data Documentation	29
6.4.3.1 _V0	29
6.5 Atmosphere Class Reference	30
6.5.1 Detailed Description	30
6.5.2 Constructor & Destructor Documentation	30
6.5.2.1 Atmosphere()	30
6.5.2.2 ~Atmosphere()	31
6.5.3 Member Function Documentation	31
6.5.3.1 getAirDensity()	31
6.5.3.2 getAirPressure()	31
6.5.3.3 getAirTemperature()	31
6.5.3.4 getSingleton()	32
6.5.3.5 getWind()	32
6.5.3.6 update()	32
6.6 AtmosphereInfo Struct Reference	32
6.6.1 Detailed Description	33
6.6.2 Member Data Documentation	33
6.6.2.1 air_density	33
6.6.2.2 air_pressure	33
6.6.2.3 air_temperature	33
6.6.2.4 wind	33
6.7 controllers::BangBang Class Reference	34
6.7.1 Constructor & Destructor Documentation	34

6.7.1.1 BangBang() [1/2]	34
6.7.1.2 BangBang() [2/2]	34
6.7.2 Member Function Documentation	35
6.7.2.1 calc()	35
6.7.2.2 clear()	35
6.7.2.3 clone()	35
6.8 Cargo Class Reference	36
6.8.1 Constructor & Destructor Documentation	36
6.8.1.1 Cargo() [1/2]	36
6.8.1.2 Cargo() [2/2]	36
6.9 Controller Class Reference	36
6.9.1 Constructor & Destructor Documentation	37
6.9.1.1 Controller()	37
6.9.1.2 ~Controller()	37
6.9.2 Member Function Documentation	37
6.9.2.1 calc() [1/2]	37
6.9.2.2 calc() [2/2]	38
6.9.2.3 clear()	38
6.9.2.4 clone()	38
6.9.2.5 ControllerFactory()	39
6.9.2.6 set_dt()	39
6.9.3 Member Data Documentation	39
6.9.3.1 _dt	39
6.10 ControllerTest Class Reference	39
6.10.1 Member Function Documentation	40
6.10.1.1 SetUp()	40
6.10.1.2 TearDown()	40
6.11 ControlSurfaces Class Reference	40
6.11.1 Detailed Description	40
6.11.2 Constructor & Destructor Documentation	41
6.11.2.1 ControlSurfaces() [1/2]	41
6.11.2.2 ControlSurfaces() [2/2]	41
6.11.3 Member Function Documentation	41
6.11.3.1 getCoefficients()	41
6.11.3.2 getNoOfSurface()	41
6.11.3.3 getValues()	42
6.11.3.4 restoreTrim()	42
6.11.3.5 setValues()	42
6.12 controllers::DoubleSetpoint Class Reference	42
6.12.1 Constructor & Destructor Documentation	42
6.12.1.1 DoubleSetpoint() [1/2]	43
6.12.1.2 DoubleSetpoint() [2/2]	43

6.12.2 Member Function Documentation	43
6.12.2.1 calc()	43
6.12.2.2 clear()	44
6.12.2.3 clone()	44
6.13 Drive Struct Reference	44
6.13.1 Detailed Description	45
6.13.2 Member Data Documentation	45
6.13.2.1 axis	45
6.13.2.2 hinges	45
6.13.2.3 noOfHinges	45
6.13.2.4 position	45
6.14 EKFScalers Struct Reference	45
6.14.1 Detailed Description	46
6.14.2 Member Data Documentation	46
6.14.2.1 baroScaler	46
6.14.2.2 predictScaler	46
6.14.2.3 updateScaler	46
6.14.2.4 zScaler	46
6.15 Forces Class Reference	46
6.15.1 Member Function Documentation	47
6.15.1.1 aerodynamic_loads()	47
6.15.1.2 angularAcceleration()	47
6.15.1.3 generateCharacteristics()	49
6.15.1.4 gravity_loads()	49
6.15.1.5 jet_lift_loads()	49
6.15.1.6 rotor_lift_loads()	50
6.16 Hinge Class Reference	50
6.16.1 Detailed Description	51
6.16.2 Constructor & Destructor Documentation	51
6.16.2.1 Hinge() [1/3]	51
6.16.2.2 Hinge() [2/3]	51
6.16.2.3 Hinge() [3/3]	51
6.16.3 Member Function Documentation	51
6.16.3.1 getRot()	52
6.16.3.2 operator=()	52
6.16.3.3 updateValue()	52
6.17 Jet Class Reference	52
6.17.1 Detailed Description	53
6.17.2 Member Function Documentation	53
6.17.2.1 getLastThrust()	53
6.17.2.2 getThrust()	53
6.17.2.3 start()	54

6.17.3 Member Data Documentation	54
6.17.3.1 phases	54
6.17.3.2 thrust	54
6.17.3.3 time	54
6.18 Load Class Reference	55
6.18.1 Detailed Description	55
6.18.2 Constructor & Destructor Documentation	55
6.18.2.1 Load() [1/2]	55
6.18.2.2 Load() [2/2]	55
6.18.3 Member Function Documentation	56
6.18.3.1 getAmmount()	56
6.18.3.2 getMass()	56
6.18.3.3 getOffset()	56
6.18.3.4 operator=()	56
6.18.3.5 release()	56
6.19 Logger Class Reference	57
6.19.1 Detailed Description	57
6.19.2 Constructor & Destructor Documentation	57
6.19.2.1 Logger()	57
6.19.2.2 ~Logger()	58
6.19.3 Member Function Documentation	58
6.19.3.1 log() [1/2]	58
6.19.3.2 log() [2/2]	58
6.19.3.3 setFmt()	59
6.19.3.4 setLogDirectory()	59
6.20 Matrices Class Reference	59
6.20.1 Member Function Documentation	60
6.20.1.1 asSkewSymmeticMatrix()	60
6.20.1.2 gyroMatrix()	60
6.20.1.3 massMatrix()	61
6.20.1.4 OM_conj()	61
6.20.1.5 quaterionsToRPY()	61
6.20.1.6 R_nb() [1/2]	62
6.20.1.7 R_nb() [2/2]	62
6.20.1.8 R_wind_b()	62
6.20.1.9 RPYtoQuaterion()	63
6.20.1.10 TMatrix() [1/2]	63
6.20.1.11 TMatrix() [2/2]	64
6.21 ODE Class Reference	64
6.21.1 Detailed Description	65
6.21.2 Member Enumeration Documentation	65
6.21.2.1 ODEMethod	65

6.21.3 Constructor & Destructor Documentation	65
6.21.3.1 ODE()	65
6.21.3.2 ~ODE()	66
6.21.4 Member Function Documentation	66
6.21.4.1 factory()	66
6.21.4.2 fromString()	66
6.21.4.3 getMicrosteps() [1/2]	66
6.21.4.4 getMicrosteps() [2/2]	67
6.21.4.5 step()	67
6.22 ODE_Euler Class Reference	68
6.22.1 Detailed Description	68
6.22.2 Constructor & Destructor Documentation	68
6.22.2.1 ODE_Euler()	68
6.22.3 Member Function Documentation	68
6.22.3.1 step()	68
6.23 ODE_Heun Class Reference	69
6.23.1 Detailed Description	69
6.23.2 Constructor & Destructor Documentation	69
6.23.2.1 ODE_Heun()	70
6.23.3 Member Function Documentation	70
6.23.3.1 step()	70
6.24 ODE_PC2 Class Reference	70
6.24.1 Detailed Description	71
6.24.2 Constructor & Destructor Documentation	71
6.24.2.1 ODE_PC2()	71
6.24.3 Member Function Documentation	71
6.24.3.1 step()	71
6.25 ODE_PC4 Class Reference	72
6.25.1 Detailed Description	72
6.25.2 Constructor & Destructor Documentation	72
6.25.2.1 ODE_PC4()	72
6.25.3 Member Function Documentation	72
6.25.3.1 step()	72
6.26 ODE_RK4 Class Reference	73
6.26.1 Detailed Description	73
6.26.2 Constructor & Destructor Documentation	73
6.26.2.1 ODE_RK4()	74
6.26.3 Member Function Documentation	74
6.26.3.1 step()	74
6.27 ODETest Class Reference	74
6.27.1 Member Function Documentation	75
6.27.1.1 SetUp()	75

6.27.1.2 TearDown()	75
6.28 Params Class Reference	75
6.28.1 Detailed Description	76
6.28.2 Constructor & Destructor Documentation	76
6.28.2.1 Params() [1/3]	76
6.28.2.2 Params() [2/3]	76
6.28.2.3 Params() [3/3]	76
6.28.2.4 ~Params()	76
6.28.3 Member Function Documentation	76
6.28.3.1 getSingleton()	77
6.28.3.2 operator=()	77
6.28.4 Member Data Documentation	77
6.28.4.1 ODE_METHOD	77
6.28.4.2 STEP_TIME	77
6.29 controllers::PID Class Reference	77
6.29.1 Member Enumeration Documentation	78
6.29.1.1 AntiWindUpMode	78
6.29.2 Constructor & Destructor Documentation	78
6.29.2.1 PID() [1/2]	78
6.29.2.2 PID() [2/2]	79
6.29.3 Member Function Documentation	79
6.29.3.1 calc()	79
6.29.3.2 clear()	80
6.29.3.3 clone()	80
6.30 controllers::PID_Discrete Class Reference	80
6.30.1 Constructor & Destructor Documentation	81
6.30.1.1 PID_Discrete() [1/2]	81
6.30.1.2 PID_Discrete() [2/2]	81
6.30.2 Member Function Documentation	81
6.30.2.1 calc()	82
6.30.2.2 clear()	82
6.30.2.3 clone()	82
6.30.2.4 set_dt()	82
6.31 Rotor Struct Reference	83
6.31.1 Detailed Description	83
6.31.2 Member Data Documentation	83
6.31.2.1 direction	83
6.31.2.2 forceCoff	84
6.31.2.3 hoverSpeed	84
6.31.2.4 maxSpeed	84
6.31.2.5 timeConstant	84
6.31.2.6 torqueCoff	84

6.32 SensorParams Struct Reference	84
6.32.1 Detailed Description	85
6.32.2 Member Data Documentation	85
6.32.2.1 bias	85
6.32.2.2 name	85
6.32.2.3 refreshTime	85
6.32.2.4 sd	85
6.33 Simulation Class Reference	85
6.33.1 Constructor & Destructor Documentation	86
6.33.1.1 Simulation()	86
6.33.1.2 ~Simulation()	86
6.33.2 Member Function Documentation	86
6.33.2.1 run()	86
6.34 TimedLoop Class Reference	86
6.34.1 Detailed Description	87
6.34.2 Constructor & Destructor Documentation	87
6.34.2.1 TimedLoop()	87
6.34.3 Member Function Documentation	87
6.34.3.1 go() [1/2]	87
6.34.3.2 go() [2/2]	87
6.35 UAVparams Struct Reference	88
6.35.1 Detailed Description	89
6.35.2 Constructor & Destructor Documentation	89
6.35.2.1 UAVparams()	89
6.35.2.2 ~UAVparams()	89
6.35.3 Member Function Documentation	89
6.35.3.1 getRotorHoverSpeeds()	89
6.35.3.2 getRotorMaxSpeeds()	89
6.35.3.3 getRotorTimeContants()	90
6.35.3.4 getSingleton()	90
6.35.3.5 loadConfig()	90
6.35.4 Member Data Documentation	90
6.35.4.1 aero_coffs	90
6.35.4.2 ahrs	90
6.35.4.3 ammo	90
6.35.4.4 cargo	90
6.35.4.5 controllers	91
6.35.4.6 ekf	91
6.35.4.7 initialMode	91
6.35.4.8 initialOrientation	91
6.35.4.9 initialPosition	91
6.35.4.10 initialVelocity	91

6.35.4.11 instantRun	91
6.35.4.12 lx	91
6.35.4.13 lxy	92
6.35.4.14 lxz	92
6.35.4.15 ly	92
6.35.4.16 lyz	92
6.35.4.17 lz	92
6.35.4.18 jets	92
6.35.4.19 m	92
6.35.4.20 name	92
6.35.4.21 noOfAmmo	93
6.35.4.22 noOfCargo	93
6.35.4.23 noOfJets	93
6.35.4.24 noOfRotors	93
6.35.4.25 rotorMixer	93
6.35.4.26 rotors	93
6.35.4.27 sensors	93
6.35.4.28 surfaceMixer	93
6.35.4.29 surfaces	94
6.35.4.30 target	94
6.36 UAVstate Struct Reference	94
6.36.1 Constructor & Destructor Documentation	95
6.36.1.1 UAVstate()	95
6.36.1.2 ~UAVstate()	95
6.36.2 Member Function Documentation	96
6.36.2.1 getAcceleration()	96
6.36.2.2 getDemandedOm()	96
6.36.2.3 getNoOfRotors()	96
6.36.2.4 getOm() [1/2]	96
6.36.2.5 getOm() [2/2]	96
6.36.2.6 getOuterForce()	97
6.36.2.7 getState()	97
6.36.2.8 getX() [1/2]	97
6.36.2.9 getX() [2/2]	97
6.36.2.10 getY() [1/2]	98
6.36.2.11 getY() [2/2]	98
6.36.2.12 operator=()	98
6.36.2.13 setAcceleration()	99
6.36.2.14 setDemandedOm()	99
6.36.2.15 setForce()	99
6.36.2.16 setOm()	99
6.36.2.17 setStatus()	100

6.36.2.18 setX() [1/2]	100
6.36.2.19 setX() [2/2]	100
6.36.2.20 setY()	101
6.36.3 Friends And Related Function Documentation	101
6.36.3.1 operator<<	101
6.36.4 Member Data Documentation	101
6.36.4.1 real_time	101
6.36.4.2 state_mtx	101
6.36.4.3 status	102
6.36.4.4 status_cv	102
6.37 controllers::ZTransform Class Reference	102
6.37.1 Constructor & Destructor Documentation	102
6.37.1.1 ZTransform() [1/2]	102
6.37.1.2 ZTransform() [2/2]	103
6.37.2 Member Function Documentation	103
6.37.2.1 calc()	103
6.37.2.2 clear()	103
6.37.2.3 clone()	104
6.38 controllers::ZTransformStatic< N, D > Class Template Reference	104
6.38.1 Constructor & Destructor Documentation	104
6.38.1.1 ZTransformStatic() [1/2]	104
6.38.1.2 ZTransformStatic() [2/2]	105
6.38.2 Member Function Documentation	105
6.38.2.1 calc()	105
6.38.2.2 clear()	105
6.38.2.3 clone()	106
7 File Documentation	107
7.1 build/CMakeFiles/3.22.1/CompilerIdC/CMakeCCompilerId.c File Reference	107
7.1.1 Macro Definition Documentation	107
7.1.1.1 __has_include	108
7.1.1.2 ARCHITECTURE_ID	108
7.1.1.3 C_VERSION	108
7.1.1.4 COMPILER_ID	108
7.1.1.5 DEC	108
7.1.1.6 HEX	108
7.1.1.7 PLATFORM_ID	109
7.1.1.8 STRINGIFY	109
7.1.1.9 STRINGIFY_HELPER	109
7.1.2 Function Documentation	109
7.1.2.1 main()	109
7.1.3 Variable Documentation	109

7.1.3.1 info_arch	109
7.1.3.2 info_compiler	109
7.1.3.3 info_language_extensions_default	110
7.1.3.4 info_language_standard_default	110
7.1.3.5 info_platform	110
7.2 build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference	110
7.2.1 Macro Definition Documentation	111
7.2.1.1 __has_include	111
7.2.1.2 ARCHITECTURE_ID	111
7.2.1.3 COMPILER_ID	111
7.2.1.4 CXX_STD	111
7.2.1.5 DEC	111
7.2.1.6 HEX	112
7.2.1.7 PLATFORM_ID	112
7.2.1.8 STRINGIFY	112
7.2.1.9 STRINGIFY_HELPER	112
7.2.2 Function Documentation	112
7.2.2.1 main()	112
7.2.3 Variable Documentation	112
7.2.3.1 info_arch	113
7.2.3.2 info_compiler	113
7.2.3.3 info_language_extensions_default	113
7.2.3.4 info_language_standard_default	113
7.2.3.5 info_platform	113
7.3 build/CMakeFiles/uav.dir/src/aircraft/aircraft.cpp.o.d File Reference	115
7.4 build/CMakeFiles/uav.dir/src/aircraft/aircraft_comm.cpp.o.d File Reference	115
7.5 build/CMakeFiles/uav.dir/src/aircraft/aircraft_impulse.cpp.o.d File Reference	115
7.6 build/CMakeFiles/uav.dir/src/dynamic/forces.cpp.o.d File Reference	115
7.7 build/CMakeFiles/uav.dir/src/dynamic/matrices.cpp.o.d File Reference	115
7.8 build/CMakeFiles/uav.dir/src/main.cpp.o.d File Reference	115
7.9 build/CMakeFiles/uav.dir/src/params.cpp.o.d File Reference	115
7.10 build/CMakeFiles/uav.dir/src/simulation/atmosphere.cpp.o.d File Reference	115
7.11 build/CMakeFiles/uav.dir/src/simulation/control.cpp.o.d File Reference	115
7.12 build/CMakeFiles/uav.dir/src/simulation/simulation.cpp.o.d File Reference	115
7.13 build/CMakeFiles/uav.dir/src/simulation/uav_state.cpp.o.d File Reference	115
7.14 build/lib/UAV_common/CMakeFiles/common.dir/src/components/control_surfaces.cpp.o.d File Reference	115
7.15 build/lib/UAV_common/CMakeFiles/common.dir/src/components/drive.cpp.o.d File Reference	115
7.16 build/lib/UAV_common/CMakeFiles/common.dir/src/components/hinge.cpp.o.d File Reference	115
7.17 build/lib/UAV_common/CMakeFiles/common.dir/src/components/loads.cpp.o.d File Reference	115
7.18 build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/controller.cpp.o.d File Reference	115
7.19 build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/bang_bang.cpp.o.d File Reference	115

7.20	build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/double_setpoint.cpp.o.d File Reference	115
7.21	build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/PID.cpp.o.d File Reference	115
7.22	build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/PIFF.cpp.o.d File Reference	115
7.23	build/lib/UAV_common/CMakeFiles/common.dir/src/logger/logger.cpp.o.d File Reference	115
7.24	build/lib/UAV_common/CMakeFiles/common.dir/src/ode/ode.cpp.o.d File Reference	115
7.25	build/lib/UAV_common/CMakeFiles/common.dir/src/parser/parser.cpp.o.d File Reference	115
7.26	build/lib/UAV_common/CMakeFiles/common.dir/src/parser/uav_params.cpp.o.d File Reference	115
7.27	build/lib/UAV_common/CMakeFiles/common.dir/src/timed_loop/timed_loop.cpp.o.d File Reference	115
7.28	build/lib/UAV_common/CMakeFiles/Controller_test.dir/src/controllers/controller_test.cpp.o.d File Reference	115
7.29	build/lib/UAV_common/CMakeFiles/ODE_test.dir/src/ode/ode_test.cpp.o.d File Reference	115
7.30	lib/UAV_common/header/common.hpp File Reference	115
7.31	lib/UAV_common/scripts/controller_plots.m File Reference	116
7.31.1	Function Documentation	116
7.31.1.1	legend()	116
7.31.1.2	plot()	117
7.31.1.3	title()	117
7.31.1.4	xlabel()	117
7.31.1.5	ylabel()	117
7.31.2	Variable Documentation	117
7.31.2.1	clc	117
7.31.2.2	csvFiles	117
7.31.2.3	data	118
7.31.2.4	figure	118
7.31.2.5	folderPath	118
7.31.2.6	i	118
7.31.2.7	off	118
7.31.2.8	on	118
7.31.2.9	x	118
7.31.2.10	y	119
7.32	lib/UAV_common/src/components/aero_coefficients.hpp File Reference	119
7.33	lib/UAV_common/src/components/components.hpp File Reference	119
7.34	lib/UAV_common/src/components/control_surfaces.cpp File Reference	119
7.35	lib/UAV_common/src/components/control_surfaces.hpp File Reference	119
7.36	lib/UAV_common/src/components/drive.cpp File Reference	120
7.37	lib/UAV_common/src/components/drive.hpp File Reference	120
7.38	lib/UAV_common/src/components/hinge.cpp File Reference	120
7.38.1	Function Documentation	120
7.38.1.1	asSkewMatrix()	120
7.39	lib/UAV_common/src/components/hinge.hpp File Reference	120
7.40	lib/UAV_common/src/components/loads.cpp File Reference	121
7.41	lib/UAV_common/src/components/loads.hpp File Reference	121

7.42 lib/UAV_common/src/components/navi.hpp File Reference	121
7.43 lib/UAV_common/src/controllers/controller.cpp File Reference	121
7.44 lib/UAV_common/src/controllers/controller.hpp File Reference	122
7.45 lib/UAV_common/src/controllers/controller_test.cpp File Reference	122
7.45.1 Function Documentation	122
7.45.1.1 getMethodsToTest()	123
7.45.1.2 INSTANTIATE_TEST_SUITE_P()	123
7.45.1.3 main()	123
7.45.1.4 TEST_P() [1/2]	123
7.45.1.5 TEST_P() [2/2]	123
7.45.2 Variable Documentation	123
7.45.2.1 plot	123
7.45.2.2 plot_directory_name	124
7.46 lib/UAV_common/src/controllers/impl/bang_bang.cpp File Reference	124
7.47 lib/UAV_common/src/controllers/impl/bang_bang.hpp File Reference	124
7.48 lib/UAV_common/src/controllers/impl/double_setpoint.cpp File Reference	124
7.49 lib/UAV_common/src/controllers/impl/double_setpoint.hpp File Reference	124
7.50 lib/UAV_common/src/controllers/impl/PID.cpp File Reference	125
7.51 lib/UAV_common/src/controllers/impl/PID.hpp File Reference	125
7.52 lib/UAV_common/src/controllers/impl/PID_discrete.cpp File Reference	125
7.53 lib/UAV_common/src/controllers/impl/PID_discrete.hpp File Reference	126
7.54 lib/UAV_common/src/controllers/impl/z_trans.cpp File Reference	126
7.54.1 Function Documentation	126
7.54.1.1 splitStringToDoubleVector()	126
7.55 lib/UAV_common/src/controllers/impl/z_trans.hpp File Reference	127
7.56 lib/UAV_common/src/logger/logger.cpp File Reference	127
7.56.1 Function Documentation	127
7.56.1.1 shouldLog()	127
7.57 lib/UAV_common/src/logger/logger.hpp File Reference	128
7.57.1 Macro Definition Documentation	128
7.57.1.1 LOGGER_MASK	128
7.58 lib/UAV_common/src/ode/ode.cpp File Reference	128
7.59 lib/UAV_common/src/ode/ode.hpp File Reference	128
7.60 lib/UAV_common/src/ode/ode_impl.hpp File Reference	129
7.61 lib/UAV_common/src/ode/ode_test.cpp File Reference	129
7.61.1 Function Documentation	129
7.61.1.1 getMethodsToTest()	130
7.61.1.2 INSTANTIATE_TEST_SUITE_P()	130
7.61.1.3 main()	130
7.61.1.4 TEST_F() [1/2]	130
7.61.1.5 TEST_F() [2/2]	130
7.61.1.6 TEST_P() [1/4]	130

7.61.1.7 TEST_P() [2/4]	131
7.61.1.8 TEST_P() [3/4]	131
7.61.1.9 TEST_P() [4/4]	131
7.62 lib/UAV_common/src/parser/parser.cpp File Reference	131
7.62.1 Function Documentation	131
7.62.1.1 parseMatrixXd()	131
7.62.1.2 parseVectorXd()	132
7.63 lib/UAV_common/src/parser/parser.hpp File Reference	132
7.63.1 Function Documentation	132
7.63.1.1 parseMatrixXd()	133
7.63.1.2 parseVectorXd()	133
7.64 lib/UAV_common/src/parser/uav_params.cpp File Reference	133
7.64.1 Function Documentation	134
7.64.1.1 parseHinge()	134
7.65 lib/UAV_common/src/parser/uav_params.hpp File Reference	134
7.66 lib/UAV_common/src/timed_loop/status.hpp File Reference	134
7.66.1 Enumeration Type Documentation	135
7.66.1.1 Status	135
7.67 lib/UAV_common/src/timed_loop/timed_loop.cpp File Reference	136
7.68 lib/UAV_common/src/timed_loop/timed_loop.hpp File Reference	136
7.69 src/aircraft/aircraft.cpp File Reference	136
7.69.1 Function Documentation	136
7.69.1.1 clampOrientationIfNecessary()	137
7.70 src/aircraft/aircraft.hpp File Reference	137
7.71 src/aircraft/aircraft_comm.cpp File Reference	137
7.72 src/aircraft/aircraft_impulse.cpp File Reference	137
7.73 src/defines.hpp File Reference	137
7.73.1 Macro Definition Documentation	138
7.73.1.1 USE_QUATERIONS	138
7.74 src/dynamic/forces.cpp File Reference	138
7.75 src/dynamic/forces.hpp File Reference	138
7.76 src/dynamic/matrices.cpp File Reference	139
7.77 src/dynamic/matrices.hpp File Reference	139
7.78 src/main.cpp File Reference	139
7.78.1 Function Documentation	139
7.78.1.1 main()	140
7.78.1.2 parseArgs()	140
7.79 src/params.cpp File Reference	140
7.80 src/params.hpp File Reference	140
7.81 src/simulation/atmosphere.cpp File Reference	140
7.82 src/simulation/atmosphere.hpp File Reference	141
7.83 src/simulation/control.cpp File Reference	141

7.83.1 Function Documentation	141
7.83.1.1 control()	142
7.83.1.2 controlListenerJob()	142
7.83.1.3 dropCargo()	142
7.83.1.4 isNormal()	142
7.83.1.5 setAtmosphere()	142
7.83.1.6 setControlSurface()	143
7.83.1.7 setForce()	143
7.83.1.8 setHinges()	143
7.83.1.9 setSpeed()	143
7.83.1.10 shoot()	143
7.83.1.11 solidSurfColision()	143
7.83.1.12 startJet()	144
7.84 src/simulation/control.hpp File Reference	144
7.84.1 Function Documentation	144
7.84.1.1 controlListenerJob()	144
7.85 src/simulation/simulation.cpp File Reference	144
7.86 src/simulation/simulation.hpp File Reference	145
7.87 src/simulation/uav_state.cpp File Reference	145
7.87.1 Function Documentation	145
7.87.1.1 operator<<()	145
7.88 src/simulation/uav_state.hpp File Reference	146
7.89 tests/test1.cpp File Reference	146
7.89.1 Function Documentation	146
7.89.1.1 fun()	146
7.89.1.2 test1()	146
7.90 tests/test2.cpp File Reference	147
7.90.1 Function Documentation	147
7.90.1.1 main()	147
7.91 tests/test3.cpp File Reference	147
7.91.1 Function Documentation	147
7.91.1.1 fun()	147
7.91.1.2 main()	148
7.92 tests/test4.cpp File Reference	148
7.92.1 Function Documentation	148
7.92.1.1 main()	148
7.93 tests/test5.cpp File Reference	148
7.93.1 Function Documentation	148
7.93.1.1 main()	148
7.94 tests/zmq_rcv.py File Reference	149
7.95 tests/zmq_rcv_last.py File Reference	149
7.96 tests/zmq_send.py File Reference	149

7.97 tests/zmq_send_tcp.py File Reference	149
Index	151

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

controllers	11
def	
Simulation constants	11
zmq_recv	13
zmq_recv_last	13
zmq_send	14
zmq_send_tcp	15

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

AeroCoefficients	17
AHRSPParams	18
Aircraft	20
Atmosphere	30
AtmosphereInfo	32
Controller	36
controllers::BangBang	34
controllers::DoubleSetpoint	42
controllers::PID	77
controllers::PID_Discrete	80
controllers::ZTransform	102
controllers::ZTransformStatic< N, D >	104
ControlSurfaces	40
Drive	44
Jet	52
Rotor	83
EKFScalers	45
Forces	46
Hinge	50
Load	55
Ammo	28
Cargo	36
Logger	57
Matrices	59
ODE	64
ODE_Euler	68
ODE_Heun	69
ODE_PC2	70
ODE_PC4	72
ODE_RK4	73
Params	75
SensorParams	84
Simulation	85
testing::TestWithParam	

ControllerTest	39
ODETest	74
TimedLoop	86
UAVparams	88
UAVstate	94

Chapter 3

Class Index

3.1 Class List

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

AeroCoefficients	
Aerodynamic coefficient	17
AHRSParams	
AHRS parameters	18
Aircraft	
Central class in simulation	20
Ammo	28
Atmosphere	
Representation of atmosphere where aircrafts fly	30
AtmosphereInfo	
DTO containing atmosphere information	32
controllers::BangBang	34
Cargo	36
Controller	36
ControllerTest	39
ControlSurfaces	
Aircraft's control surfaces	40
controllers::DoubleSetpoint	42
Drive	
Drive propelling aircraft	44
EKFScalers	
Scalers for EKF	45
Forces	46
Hinge	
Hinge connecting aircraft with drives	50
Jet	
Jet rocket engine	52
Load	
Load of aircraft that can be dropped or launched	55
Logger	
Log vector data with timestamp in file	57
Matrices	59
ODE	
Ordinal differential equation solver	64
ODE_Euler	
Explicit Euler algorithm	68

ODE_Heun	
Second order explicit Heun algorithm	69
ODE_PC2	
Second order predictor-corrector method Second order Adams-bashforth and Adams-moulton	70
ODE_PC4	
Fourth order predictor-corrector method Fourth order Adams-bashforth and Adams-moulton	72
ODE_RK4	
Fourth order Runge Kutta algorithm	73
ODETest	74
Params	
Simulation parameters	75
controllers::PID	77
controllers::PID_Discrete	80
Rotor	
Rotor engine with controlled speed	83
SensorParams	
Base parameters of a sensor	84
Simulation	85
TimedLoop	
Simulation of real-time synchronized loop	86
UAVparams	
Parsed UAV configuration from XML	88
UAVstate	94
controllers::ZTransform	102
controllers::ZTransformStatic< N, D >	104

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

build/CMakeFiles/3.22.1/CompilerIdC/CMakeCCompilerId.c	107
build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp	110
build/CMakeFiles/uav.dir/src/main.cpp.o.d	115
build/CMakeFiles/uav.dir/src/params.cpp.o.d	115
build/CMakeFiles/uav.dir/src/aircraft/aircraft.cpp.o.d	115
build/CMakeFiles/uav.dir/src/aircraft/aircraft_comm.cpp.o.d	115
build/CMakeFiles/uav.dir/src/aircraft/aircraft_impulse.cpp.o.d	115
build/CMakeFiles/uav.dir/src/dynamic/forces.cpp.o.d	115
build/CMakeFiles/uav.dir/src/dynamic/matrices.cpp.o.d	115
build/CMakeFiles/uav.dir/src/simulation/atmosphere.cpp.o.d	115
build/CMakeFiles/uav.dir/src/simulation/control.cpp.o.d	115
build/CMakeFiles/uav.dir/src/simulation/simulation.cpp.o.d	115
build/CMakeFiles/uav.dir/src/simulation/uav_state.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/common.dir/src/components/control_surfaces.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/common.dir/src/components/drive.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/common.dir/src/components/hinge.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/common.dir/src/components/loads.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/controller.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/bang_bang.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/double_setpoint.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/PID.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/PIFF.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/common.dir/src/logger/logger.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/common.dir/src/ode/ode.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/common.dir/src/parser/parser.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/common.dir/src/parser/uav_params.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/common.dir/src/timed_loop/timed_loop.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/Controller_test.dir/src/controllers/controller_test.cpp.o.d	115
build/lib/UAV_common/CMakeFiles/ODE_test.dir/src/ode/ode_test.cpp.o.d	115
lib/UAV_common/header/common.hpp	115
lib/UAV_common/scripts/controller_plots.m	116
lib/UAV_common/src/components/aero_coefficients.hpp	119
lib/UAV_common/src/components/components.hpp	119
lib/UAV_common/src/components/control_surfaces.cpp	119
lib/UAV_common/src/components/control_surfaces.hpp	119

lib/UAV_common/src/components/drive.cpp	120
lib/UAV_common/src/components/drive.hpp	120
lib/UAV_common/src/components/hinge.cpp	120
lib/UAV_common/src/components/hinge.hpp	120
lib/UAV_common/src/components/loads.cpp	121
lib/UAV_common/src/components/loads.hpp	121
lib/UAV_common/src/components/navi.hpp	121
lib/UAV_common/src/controllers/controller.cpp	121
lib/UAV_common/src/controllers/controller.hpp	122
lib/UAV_common/src/controllers/controller_test.cpp	122
lib/UAV_common/src/controllers/impl/bang_bang.cpp	124
lib/UAV_common/src/controllers/impl/bang_bang.hpp	124
lib/UAV_common/src/controllers/impl/double_setpoint.cpp	124
lib/UAV_common/src/controllers/impl/double_setpoint.hpp	124
lib/UAV_common/src/controllers/impl/PID.cpp	125
lib/UAV_common/src/controllers/impl/PID.hpp	125
lib/UAV_common/src/controllers/impl/PID_discrete.cpp	125
lib/UAV_common/src/controllers/impl/PID_discrete.hpp	126
lib/UAV_common/src/controllers/impl/z_trans.cpp	126
lib/UAV_common/src/controllers/impl/z_trans.hpp	127
lib/UAV_common/src/logger/logger.cpp	127
lib/UAV_common/src/logger/logger.hpp	128
lib/UAV_common/src/ode/ode.cpp	128
lib/UAV_common/src/ode/ode.hpp	128
lib/UAV_common/src/ode/ode_impl.hpp	129
lib/UAV_common/src/ode/ode_test.cpp	129
lib/UAV_common/src/parser/parser.cpp	131
lib/UAV_common/src/parser/parser.hpp	132
lib/UAV_common/src/parser/uav_params.cpp	133
lib/UAV_common/src/parser/uav_params.hpp	134
lib/UAV_common/src/timed_loop/status.hpp	134
lib/UAV_common/src/timed_loop/timed_loop.cpp	136
lib/UAV_common/src/timed_loop/timed_loop.hpp	136
src/defines.hpp	137
src/main.cpp	139
src/params.cpp	140
src/params.hpp	140
src/aircraft/aircraft.cpp	136
src/aircraft/aircraft.hpp	137
src/aircraft/aircraft_comm.cpp	137
src/aircraft/aircraft_impulse.cpp	137
src/dynamic/forces.cpp	138
src/dynamic/forces.hpp	138
src/dynamic/matrices.cpp	139
src/dynamic/matrices.hpp	139
src/simulation/atmosphere.cpp	140
src/simulation/atmosphere.hpp	141
src/simulation/control.cpp	141
src/simulation/control.hpp	144
src/simulation/simulation.cpp	144
src/simulation/simulation.hpp	145
src/simulation/uav_state.cpp	145
src/simulation/uav_state.hpp	146
tests/test1.cpp	146
tests/test2.cpp	147
tests/test3.cpp	147
tests/test4.cpp	148
tests/test5.cpp	148

tests/ zmq_recv.py	149
tests/ zmq_recv_last.py	149
tests/ zmq_send.py	149
tests/ zmq_send_tcp.py	149

Chapter 5

Namespace Documentation

5.1 controllers Namespace Reference

Classes

- class [BangBang](#)
- class [DoubleSetpoint](#)
- class [PID](#)
- class [PID_Discrete](#)
- class [ZTransformStatic](#)
- class [ZTransform](#)

5.2 def Namespace Reference

[Simulation](#) constants.

Variables

- const double [GRAVITY_CONST](#) = 9.81
Gravity constant on Earth in m/s2.
- const double [FRICTION_EPS](#) = 0.001
minimal friction that is calculated (numerical float eps)
- const double [GENTLY_PUSH](#) = 0.15
artificial force 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

5.2.1 Detailed Description

[Simulation](#) constants.

5.2.2 Variable Documentation

5.2.2.1 DOUBLE_EPS

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

near zero floating point eps

5.2.2.2 FRICTION_EPS

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

minimal friction that is calculated (numerical float eps)

5.2.2.3 GENTLY_PUSH

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

artificial force coefficient. Protect again diving objects in horizontal wall

5.2.2.4 GRAVITY_CONST

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

Gravity constant on Earth in m/s².

5.2.2.5 MIXING_FUNCTION

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

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

5.2.2.6 validityOfForce

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

how many times outer force should be used

5.3 zmq_recv Namespace Reference

Variables

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

5.3.1 Variable Documentation

5.3.1.1 context

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

5.3.1.2 s

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

5.3.1.3 socket

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

5.3.1.4 topicfilter

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

5.4 zmq_recv_last Namespace Reference

Variables

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

5.4.1 Variable Documentation

5.4.1.1 context

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

5.4.1.2 s

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

5.4.1.3 socket

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

5.4.1.4 topicfilter

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

5.5 zmq_send Namespace Reference

Variables

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

5.5.1 Variable Documentation

5.5.1.1 context

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

5.5.1.2 counter

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

5.5.1.3 socket

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

5.6 zmq_send_tcp Namespace Reference

Variables

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

5.6.1 Variable Documentation

5.6.1.1 angle

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

5.6.1.2 context

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

5.6.1.3 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 AHRSPParams::alpha
```

6.2.2.2 [Q](#)

```
double AHRSPParams::Q
```

6.2.2.3 [R](#)

```
double AHRSPParams::R
```

6.2.2.4 [type](#)

```
std::string AHRSPParams::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 destructor 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, Eigen::Vector3d r)
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

<i>COR</i>	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.
<i>mi_static</i>	static friction coefficient
<i>mi_dynamic</i>	dynamic friction coefficient
<i>collisionPoint</i>	point of collision
<i>surfaceNormal</i>	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

<i>m</i>	object mass
<i>speed</i>	object's initial velocity vector in body frame
<i>r</i>	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

<i>index</i>	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,
    Eigen::Vector3d r ) [protected]
```

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

Parameters

<i>delta_m</i>	mass reduction
<i>r</i>	lost mass position in local frame

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

<i>time</i>	time of simulation
<i>state</i>	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

<i>socket</i>	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

<i>type</i>	type of drive: 'r' - rotor, 'j' - jet
<i>index</i>	index of drive
<i>hinge_index</i>	index of hinde
<i>value</i>	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

<i>angles</i>	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

<i>index</i>	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

<i>index</i>	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 destructor.
- `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 destructor.

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

<i>info</i>	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 controllers::BangBang Class Reference

```
#include <bang_bang.hpp>
```

Inheritance diagram for controllers::BangBang:

Collaboration diagram for controllers::BangBang:

Public Member Functions

- [BangBang](#) (double high, double low, double delta=0.0)
Constructor with all Bang-bang controller parameters.
- [BangBang](#) (rapidxml::xml_node<> *controller_node)
Construct controller with parameters from xml.
- double [calc](#) (double desired, double actual, [[maybe_unused]] double dt) override
calc output of controller with specific time step
- void [clear](#) () override
clear internal state
- std::unique_ptr< [Controller](#) > [clone](#) () const override
virtual clone method

Additional Inherited Members

6.7.1 Constructor & Destructor Documentation

6.7.1.1 BangBang() [1/2]

```
controllers::BangBang::BangBang (
    double high,
    double low,
    double delta = 0.0 )
```

Constructor with all Bang-bang controller parameters.

Parameters

<i>high</i>	output when error is positive
<i>low</i>	output when error is negative
<i>delta</i>	hysteresis symetrical to zero

6.7.1.2 BangBang() [2/2]

```
controllers::BangBang::BangBang (
    rapidxml::xml_node<> * controller_node )
```

Construct controller with parameters from xml.

Parameters

<i>controller_node</i>	xml node with controller params
------------------------	---------------------------------

6.7.2 Member Function Documentation

6.7.2.1 calc()

```
double controllers::BangBang::calc (
    double desired,
    double actual,
    [[maybe_unused] ] double dt ) [override]
```

calc output of controller with specific time step

Parameters

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value
<i>dt</i>	time step

Returns

output of controller

6.7.2.2 clear()

```
void controllers::BangBang::clear ( ) [override], [virtual]
```

clear internal state

Implements [Controller](#).

6.7.2.3 clone()

```
std::unique_ptr< Controller > controllers::BangBang::clone ( ) const [override], [virtual]
```

virtual clone method

Implements [Controller](#).

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

- lib/UAV_common/src/controllers/impl/[bang_bang.hpp](#)
- lib/UAV_common/src/controllers/impl/[bang_bang.cpp](#)

6.8 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

6.8.1 Constructor & Destructor Documentation

6.8.1.1 Cargo() [1/2]

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

6.8.1.2 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](#)

6.9 Controller Class Reference

```
#include <controller.hpp>
```

Inheritance diagram for Controller:

Public Member Functions

- [Controller](#) ()
Default constructor.
- [~Controller](#) ()
Empty destructor for derived classes.
- virtual void [set_dt](#) (double dt)
Set new time step.
- double [calc](#) (double desired, double actual)
calc output of controller
- virtual double [calc](#) (double desired, double actual, double dt)=0
calc output of controller with specific time step
- virtual void [clear](#) ()=0
clear internal state
- virtual std::unique_ptr< [Controller](#) > [clone](#) () const =0
virtual clone method

Static Public Member Functions

- static std::unique_ptr< [Controller](#) > [ControllerFactory](#) (rapidxml::xml_node<> *controller_node)
construct controller from given node. If xml is not valid return nullptr.

Protected Attributes

- double [_dt](#)

6.9.1 Constructor & Destructor Documentation

6.9.1.1 Controller()

```
Controller::Controller ( ) [inline]
```

Default constructor.

6.9.1.2 ~Controller()

```
Controller::~~Controller ( ) [inline]
```

Empty destructor for derived classes.

6.9.2 Member Function Documentation

6.9.2.1 calc() [1/2]

```
double Controller::calc (
    double desired,
    double actual ) [inline]
```

calc output of controller

Parameters

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value

Returns

output of controller

6.9.2.2 calc() [2/2]

```
virtual double Controller::calc (
    double desired,
    double actual,
    double dt ) [pure virtual]
```

calc output of controller with specific time step

Parameters

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value
<i>dt</i>	time step

Returns

output of controller

Implemented in [controllers::PID_Discrete](#), [controllers::PID](#), and [controllers::DoubleSetpoint](#).

6.9.2.3 clear()

```
virtual void Controller::clear ( ) [pure virtual]
```

clear internal state

Implemented in [controllers::ZTransform](#), [controllers::ZTransformStatic< N, D >](#), [controllers::PID_Discrete](#), [controllers::PID](#), [controllers::DoubleSetpoint](#), and [controllers::BangBang](#).

6.9.2.4 clone()

```
virtual std::unique_ptr<Controller> Controller::clone ( ) const [pure virtual]
```

virtual clone method

Implemented in [controllers::ZTransform](#), [controllers::ZTransformStatic< N, D >](#), [controllers::PID_Discrete](#), [controllers::PID](#), [controllers::DoubleSetpoint](#), and [controllers::BangBang](#).

6.9.2.5 ControllerFactory()

```
std::unique_ptr< Controller > Controller::ControllerFactory (
    rapidxml::xml_node<> * controller_node ) [static]
```

construct controller from given node. If xml is not valid return nullptr.

Parameters

<i>controller_node</i>	xml node with controller config
------------------------	---------------------------------

6.9.2.6 set_dt()

```
virtual void Controller::set_dt (
    double dt ) [inline], [virtual]
```

Set new time step.

Parameters

<i>dt</i>	new time step
-----------	---------------

Reimplemented in [controllers::PID_Discrete](#).

6.9.3 Member Data Documentation

6.9.3.1 _dt

```
double Controller::_dt [protected]
```

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

- lib/UAV_common/src/controllers/[controller.hpp](#)
- lib/UAV_common/src/controllers/[controller.cpp](#)

6.10 ControllerTest Class Reference

Inheritance diagram for ControllerTest:

Collaboration diagram for ControllerTest:

Protected Member Functions

- void [SetUp](#) () override
- void [TearDown](#) () override

6.10.1 Member Function Documentation

6.10.1.1 SetUp()

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

6.10.1.2 TearDown()

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

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

- lib/UAV_common/src/controllers/[controller_test.cpp](#)

6.11 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

6.11.1 Detailed Description

[Aircraft](#)'s control surfaces.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 ControlSurfaces() [1/2]

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

6.11.2.2 ControlSurfaces() [2/2]

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

Constructor.

Parameters

<i>noOfSurfaces</i>	number of independent surfaces
<i>matrix</i>	coefficients matrix
<i>min</i>	vector of min angles
<i>max</i>	vector of max angles
<i>trim</i>	vector of trim angles

6.11.3 Member Function Documentation

6.11.3.1 getCoefficients()

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

6.11.3.2 getNoOfSurface()

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

6.11.3.3 `getValues()`

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

6.11.3.4 `restoreTrim()`

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

6.11.3.5 `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](#)

6.12 `controllers::DoubleSetpoint` Class Reference

```
#include <double_setpoint.hpp>
```

Inheritance diagram for `controllers::DoubleSetpoint`:

Collaboration diagram for `controllers::DoubleSetpoint`:

Public Member Functions

- [DoubleSetpoint](#) (double high, double mid, double low, double mid_range, double delta=0.0)
Constructor with all Bang-bang controller parameters.
- [DoubleSetpoint](#) (rapidxml::xml_node<> *controller_node)
Construct controller with parameters from xml.
- double [calc](#) (double desired, double actual, double dt) override
calc output of controller with specific time step
- void [clear](#) () override
clear internal state
- std::unique_ptr< [Controller](#) > [clone](#) () const override
virtual clone method

Additional Inherited Members

6.12.1 Constructor & Destructor Documentation

6.12.1.1 DoubleSetpoint() [1/2]

```
controllers::DoubleSetpoint::DoubleSetpoint (
    double high,
    double mid,
    double low,
    double mid_range,
    double delta = 0.0 )
```

Constructor with all Bang-bang controller parameters.

Parameters

<i>high</i>	output when error is in positive range
<i>mid</i>	output when error is in center range
<i>low</i>	output when error is in negative range
<i>mid_range</i>	size of center field from zero
<i>delta</i>	hysteresis symetrical to zero

6.12.1.2 DoubleSetpoint() [2/2]

```
controllers::DoubleSetpoint::DoubleSetpoint (
    rapidxml::xml_node<> * controller_node )
```

Construct controller with parameters from xml.

Parameters

<i>controller_node</i>	xml node with controller params
------------------------	---------------------------------

6.12.2 Member Function Documentation

6.12.2.1 calc()

```
double controllers::DoubleSetpoint::calc (
    double desired,
    double actual,
    double dt ) [override], [virtual]
```

calc output of controller with specific time step

Parameters

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value
<i>dt</i>	time step

Returns

output of controller

Implements [Controller](#).

6.12.2.2 clear()

```
void controllers::DoubleSetpoint::clear ( ) [override], [virtual]
```

clear internal state

Implements [Controller](#).

6.12.2.3 clone()

```
std::unique_ptr< Controller > controllers::DoubleSetpoint::clone ( ) const [override], [virtual]
```

virtual clone method

Implements [Controller](#).

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

- [lib/UAV_common/src/controllers/impl/double_setpoint.hpp](#)
- [lib/UAV_common/src/controllers/impl/double_setpoint.cpp](#)

6.13 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]

6.13.1 Detailed Description

[Drive](#) propelling aircraft.

6.13.2 Member Data Documentation

6.13.2.1 axis

`Eigen::Vector3d Drive::axis`

6.13.2.2 hinges

`Hinge Drive::hinges[2]`

6.13.2.3 noOfHinges

`int Drive::noOfHinges`

6.13.2.4 position

`Eigen::Vector3d Drive::position`

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

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

6.14 EKFScalers Struct Reference

Scalers for EKF.

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

Public Attributes

- double [predictScaler](#)
- double [updateScaler](#)
- double [baroScaler](#)
- double [zScaler](#)

6.14.1 Detailed Description

Scalers for EKF.

6.14.2 Member Data Documentation

6.14.2.1 `baroScaler`

```
double EKFScalers::baroScaler
```

6.14.2.2 `predictScaler`

```
double EKFScalers::predictScaler
```

6.14.2.3 `updateScaler`

```
double EKFScalers::updateScaler
```

6.14.2.4 `zScaler`

```
double EKFScalers::zScaler
```

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

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

6.15 Forces Class Reference

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

Static Public Member Functions

- static Vector< double, 6 > [gravity_loads](#) (double mass, 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.

6.15.1 Member Function Documentation

6.15.1.1 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

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

Returns

loads in body frame

6.15.1.2 angularAcceleration()

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

Calculates acceleration of propellers.

Parameters

<i>demandedAngularVelocity</i>	vector of demanded angular velocities
<i>rotorAngularVelocity</i>	vector of actual angular velocities

Returns

vector of angular accelerations

6.15.1.3 generateCharacteristics()

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

Generates aerodynamics characteristics and save in csv files.

Parameters

<i>surface</i>	reference to ControlSurfaces instance
<i>aero</i>	reference to AeroCoefficients instance

6.15.1.4 gravity_loads()

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

Calculates gravity loads acting on UAV.

Parameters

<i>mass</i>	aircraft mass
<i>r_nb</i>	rotation matrix from world to body frame

Returns

gravity load in body frame

6.15.1.5 jet_lift_loads()

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

```

    Jet * jets,
    double time ) [static]

```

Calculates loads generated by jet.

Parameters

<i>noOfJets</i>	numbers of jets
<i>jets</i>	pointer to jet instance
<i>time</i>	simulation time

Returns

loads in body frame

6.15.1.6 rotor_lift_loads()

```

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

```

Calculates loads generated by rotors.

Parameters

<i>noOfRotors</i>	numbers of rotors
<i>rotors</i>	pointer to rotor instance
<i>rotorAngularVelocity</i>	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](#)

6.16 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 rotation matrix of orientation change due to hinge.

6.16.1 Detailed Description

[Hinge](#) connecting aircraft with drives.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 [Hinge\(\)](#) [1/3]

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

6.16.2.2 [Hinge\(\)](#) [2/3]

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

6.16.2.3 [Hinge\(\)](#) [3/3]

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

6.16.3 Member Function Documentation

6.16.3.1 getRot()

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

Get rotation matrix of orientation change due to hinge.

Returns

rotation matrix

6.16.3.2 operator=()

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

6.16.3.3 updateValue()

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

set new angle on hinge

Parameters

<i>newValue</i>	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](#)

6.17 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](#)

6.17.1 Detailed Description

[Jet](#) rocket engine.

6.17.2 Member Function Documentation

6.17.2.1 [getLastThrust\(\)](#)

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

get last calculated thrust

Returns

last calculated thrust

6.17.2.2 [getThrust\(\)](#)

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

get thrust in specific time

Parameters

<i>time</i>	timestamp
-------------	-----------

Returns

thrust value in Newtons

6.17.2.3 start()

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

start jet engine

Parameters

<i>time</i>	timestamp of start
-------------	--------------------

Returns

true if start succesful, false if already started

6.17.3 Member Data Documentation**6.17.3.1 phases**

```
int Jet::phases
```

6.17.3.2 thrust

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

6.17.3.3 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](#)

6.18 Load Class Reference

[Load](#) of aircraft that can be dropped 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 [getAmmount](#) ()
get ammount 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)

6.18.1 Detailed Description

[Load](#) of aircraft that can be dropped or launched.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 Load() [1/2]

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

6.18.2.2 Load() [2/2]

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

6.18.3 Member Function Documentation

6.18.3.1 getAmmount()

```
int Load::getAmmount ( ) [inline]
```

get ammount of load

Returns

ammount

6.18.3.2 getMass()

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

get mass of load

Returns

mass

6.18.3.3 getOffset()

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

get offset of load

Returns

offset vector

6.18.3.4 operator=()

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

6.18.3.5 release()

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

Try to release load.

Parameters

<i>time</i>	
-------------	--

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](#)

6.19 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](#) ()
Destructor.
- 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.

6.19.1 Detailed Description

Log vector data with timestamp in file.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 Logger()

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

Constructor.

Parameters

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

6.19.2.2 ~Logger()

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

Deconstructor.

6.19.3 Member Function Documentation**6.19.3.1 log() [1/2]**

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

Log one row.

Parameters

<i>time</i>	timestamp
<i>args</i>	list of doubles

6.19.3.2 log() [2/2]

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

Log one row.

Parameters

<i>time</i>	timestamp
<i>args</i>	list of double vectors

6.19.3.3 setFmt()

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

Set new format if was not known in constructor.

Parameters

<i>fmt</i>	new format
------------	------------

6.19.3.4 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

<i>subdirectory</i>	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](#)

6.20 Matrices Class Reference

```
#include <matrices.hpp>
```

Static Public Member Functions

- static Matrix< double, 6, 6 > [massMatrix](#) ()
Constructs 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, 6, 6 > [TMatrix](#) (Vector< double, 7 > *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.
- static Eigen::Matrix3d [asSkewSymmetricMatrix](#) (Eigen::Vector3d v)
Calculate skew symmetric matrix, used in cross products.

6.20.1 Member Function Documentation

6.20.1.1 asSkewSymmetricMatrix()

```
Eigen::Matrix3d Matrices::asSkewSymmetricMatrix (
    Eigen::Vector3d v ) [static]
```

Calculate skew symmetric matrix, used in cross products.

Parameters

<i>v</i>	vector 3d
----------	-----------

Returns

skew symmetric matrix

6.20.1.2 gyroMatrix()

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

Calculates gyroscopic matrix.

Parameters

<i>x</i>	velocity vector
----------	-----------------

Returns

gyroscopic matrix

6.20.1.3 massMatrix()

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

Constucts initial mass matrix, based on parameters.

Returns

mass matrix

6.20.1.4 OM_conj()

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

Calculates conjugation matrix for angular velocity.

Parameters

<i>x</i>	vector of velocites
----------	---------------------

Returns

conjugation matrix

6.20.1.5 quaterionsToRPY()

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

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

Parameters

<i>y</i>	position & orientation vector
----------	-------------------------------

Returns

position & orientation vector. Orientation is given in RPY

6.20.1.6 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

<i>y</i>	actual position & orietation vector (RPY)
----------	---

Returns

rotation matrix

6.20.1.7 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

<i>y</i>	actual position & orietation vector (quaterions)
----------	--

Returns

rotation matrix

6.20.1.8 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

<i>alpha</i>	angle of attack
<i>beta</i>	angle of slide

Returns

rotation matrix

6.20.1.9 RPYtoQuaterion()

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

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

Parameters

<i>y</i>	position & orientation vector
----------	-------------------------------

Returns

position & orientation vector. Orientation is given in quaterions

6.20.1.10 TMatrix() [1/2]

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

Calculates transformation matrix from body to world frame for velocities.

Parameters

<i>y</i>	actual position vector
----------	------------------------

Returns

transformation matrix

6.20.1.11 TMatrix() [2/2]

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

Calculates transformation matrix from body to world frame for velocities.

Parameters

y	actual position vector, quaterion
---	-----------------------------------

Returns

transformation matrix

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

- src/dynamic/[matrices.hpp](#)
- src/dynamic/[matrices.cpp](#)

6.21 ODE Class Reference

Ordinal differential equation solver.

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

Inheritance diagram for ODE:

Public Types

- enum [ODEMethod](#) {
[Euler](#) , [Heun](#) , [RK4](#) , [PC2](#) ,
[PC4](#) , [NONE](#) }
Supported solving method.

Public Member Functions

- [ODE](#) (int micro_steps)
Constructor.
- virtual [~ODE](#) ()
Virtual destructor.
- 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.

6.21.1 Detailed Description

Ordinal differential equation solver.

6.21.2 Member Enumeration Documentation

6.21.2.1 ODEMethod

enum [ODE::ODEMethod](#)

Supported solving method.

Enumerator

Euler	
Heun	
RK4	
PC2	
PC4	
NONE	

6.21.3 Constructor & Destructor Documentation

6.21.3.1 ODE()

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

Constructor.

6.21.3.2 ~ODE()

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

Virtual destructor.

6.21.4 Member Function Documentation

6.21.4.1 factory()

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

Factory constructing ODE solvers.

Parameters

<i>method</i>	type of desired method
---------------	------------------------

Returns

instance of ODE solver

6.21.4.2 fromString()

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

Parse solving method from string.

Parameters

<i>str</i>	input string
------------	--------------

Returns

solving method if parsed, NONE if unknown

6.21.4.3 getMicrosteps() [1/2]

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

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

Returns

microsteps

6.21.4.4 getMicrosteps() [2/2]

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

Get microsteps of given method.

Parameters

<i>method</i>	method type
---------------	-------------

Returns

number of microstep in one algorithm step

6.21.4.5 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

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

Returns

Implemented in [ODE_PC4](#), [ODE_PC2](#), [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](#)

6.22 ODE_Euler Class Reference

Explicit Euler algorithm.

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

Inheritance diagram for ODE_Euler:

Collaboration diagram for ODE_Euler:

Public Member Functions

- [ODE_Euler](#) ()
- Eigen::VectorXd [step](#) (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun, double h) override
One step of explicit solving algorithm.

Additional Inherited Members

6.22.1 Detailed Description

Explicit Euler algorithm.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 ODE_Euler()

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

6.22.3 Member Function Documentation

6.22.3.1 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

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

Returns

Implements [ODE](#).

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

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

6.23 ODE_Heun Class Reference

Second order explicit Heun algorithm.

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

Inheritance diagram for ODE_Heun:

Collaboration diagram for ODE_Heun:

Public Member Functions

- [ODE_Heun](#) ()
- `Eigen::VectorXd` [step](#) (double t, `Eigen::VectorXd` y0, `std::function`< `Eigen::VectorXd`(double, `Eigen::VectorXd`)> rhs_fun, double h) override
One step of explicit solving algorithm.

Additional Inherited Members

6.23.1 Detailed Description

Second order explicit Heun algorithm.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 ODE_Heun()

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

6.23.3 Member Function Documentation

6.23.3.1 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

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

Returns

Implements [ODE](#).

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

- [lib/UAV_common/src/ode/ode_impl.hpp](#)

6.24 ODE_PC2 Class Reference

Second order predictor-corrector method Second order Adams-bashforth and Adams-moulton.

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

Inheritance diagram for ODE_PC2:

Collaboration diagram for ODE_PC2:

Public Member Functions

- [ODE_PC2](#) ()
- `Eigen::VectorXd` [step](#) (double t, `Eigen::VectorXd` y0, `std::function< Eigen::VectorXd(double, Eigen::VectorXd)>` rhs_fun, double h) override
One step of explicit solving algorithm.

Additional Inherited Members

6.24.1 Detailed Description

Second order predictor-corrector method Second order Adams-bashforth and Adams-moulton.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 ODE_PC2()

```
ODE_PC2::ODE_PC2 ( ) [inline]
```

6.24.3 Member Function Documentation

6.24.3.1 step()

```
Eigen::VectorXd ODE_PC2::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

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

Returns

Implements [ODE](#).

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

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

6.25 ODE_PC4 Class Reference

Fourth order predictor-corrector method Fourth order Adams-bashforth and Adams-moulton.

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

Inheritance diagram for ODE_PC4:

Collaboration diagram for ODE_PC4:

Public Member Functions

- [ODE_PC4](#) ()
- `Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun, double h) override`
One step of explicit solving algorithm.

Additional Inherited Members

6.25.1 Detailed Description

Fourth order predictor-corrector method Fourth order Adams-bashforth and Adams-moulton.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 ODE_PC4()

```
ODE_PC4::ODE_PC4 ( ) [inline]
```

6.25.3 Member Function Documentation

6.25.3.1 step()

```
Eigen::VectorXd ODE_PC4::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

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

Returns

Implements [ODE](#).

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

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

6.26 ODE_RK4 Class Reference

Fourth order Runge Kutta algorithm.

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

Inheritance diagram for ODE_RK4:

Collaboration diagram for ODE_RK4:

Public Member Functions

- [ODE_RK4](#) ()
- `Eigen::VectorXd` [step](#) (double t, `Eigen::VectorXd` y0, `std::function< Eigen::VectorXd(double, Eigen::VectorXd)>` rhs_fun, double h) override
One step of explicit solving algorithm.

Additional Inherited Members

6.26.1 Detailed Description

Fourth order Runge Kutta algorithm.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 ODE_RK4()

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

6.26.3 Member Function Documentation

6.26.3.1 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

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

Returns

Implements [ODE](#).

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

- [lib/UAV_common/src/ode/ode_impl.hpp](#)

6.27 ODETest Class Reference

Inheritance diagram for ODETest:

Collaboration diagram for ODETest:

Protected Member Functions

- void [SetUp](#) () override
- void [TearDown](#) () override

6.27.1 Member Function Documentation

6.27.1.1 SetUp()

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

6.27.1.2 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](#)

6.28 Params Class Reference

[Simulation](#) parameters.

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

Public Member Functions

- [Params](#) ()
Constructor.
- [Params](#) (const [Params](#) &)=delete
- [Params](#) & [operator=](#) (const [Params](#) &)=delete
- [Params](#) ([Params](#) &&)=delete
- [~Params](#) ()
Destructor.

Static Public Member Functions

- static const [Params](#) * [getSingleton](#) ()
Get singleton of [Params](#).

Public Attributes

- double [STEP_TIME](#)
Step time of simulation. Step of [ODE](#) solving methods.
- std::string [ODE_METHOD](#)
[ODE](#) solving method used in simulation.

6.28.1 Detailed Description

[Simulation](#) parameters.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 Params() [1/3]

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

Constructor.

6.28.2.2 Params() [2/3]

```
Params::Params (
    const Params & ) [delete]
```

6.28.2.3 Params() [3/3]

```
Params::Params (
    Params && ) [delete]
```

6.28.2.4 ~Params()

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

Deconstructor.

6.28.3 Member Function Documentation

6.28.3.1 getSingleton()

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

Get singleton of [Params](#).

Returns

const pointer to [Params](#) instance. Return nullptr if not initialized

6.28.3.2 operator=()

```
Params& Params::operator= (
    const Params & ) [delete]
```

6.28.4 Member Data Documentation

6.28.4.1 ODE_METHOD

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

[ODE](#) solving method used in simulation.

6.28.4.2 STEP_TIME

```
double Params::STEP_TIME
```

Step time of simulation. Step of [ODE](#) solving methods.

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

- [src/params.hpp](#)
- [src/params.cpp](#)

6.29 controllers::PID Class Reference

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

Inheritance diagram for controllers::PID:

Collaboration diagram for controllers::PID:

Public Types

- enum class [AntiWindUpMode](#) { [NONE](#) , [CLAMPING](#) }
Methods of handling windup in controller.

Public Member Functions

- [PID](#) (double Kp, double Ki, double Kd, double Kff=0.0, double min=-std::numeric_limits< double >::max(), double max=std::numeric_limits< double >::max(), [AntiWindUpMode](#) antiWindUp=[AntiWindUpMode::CLAMPING](#))
Constructor with all [PID](#) controller parameters.
- [PID](#) (rapidxml::xml_node<> *controller_node)
Construct controller with parameters from xml.
- double [calc](#) (double desired, double actual, double dt) override
calc output of controller with specific time step
- void [clear](#) () override
clear internal state
- std::unique_ptr< [Controller](#) > [clone](#) () const override
virtual clone method

Additional Inherited Members

6.29.1 Member Enumeration Documentation

6.29.1.1 AntiWindUpMode

```
enum controllers::PID::AntiWindUpMode [strong]
```

Methods of handling windup in controller.

Enumerator

NONE	
CLAMPING	

6.29.2 Constructor & Destructor Documentation

6.29.2.1 PID() [1/2]

```
PID::PID (
    double Kp,
    double Ki,
```

```
double Kd,
double Kff = 0.0,
double min = -std::numeric_limits<double>::max(),
double max = std::numeric_limits<double>::max(),
AntiWindUpMode antiWindUp = AntiWindUpMode::CLAMPING )
```

Constructor with all [PID](#) controller parameters.

Parameters

<i>Kp</i>	P term
<i>Ki</i>	I term
<i>Kd</i>	D term
<i>Kff</i>	FF term
<i>min</i>	saturation - lower range limit
<i>max</i>	saturation - upper range limit
<i>antiWindUp</i>	antiwindup method

6.29.2.2 PID() [2/2]

```
PID::PID (
    rapidxml::xml_node<> * controller_node )
```

Construct controller with parameters from xml.

Parameters

<i>controller_node</i>	xml node with controller params
------------------------	---------------------------------

6.29.3 Member Function Documentation

6.29.3.1 calc()

```
double controllers::PID::calc (
    double desired,
    double actual,
    double dt ) [override], [virtual]
```

calc output of controller with specific time step

Parameters

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value
<i>dt</i>	time step

Returns

output of controller

Implements [Controller](#).

6.29.3.2 clear()

```
void PID::clear ( ) [override], [virtual]
```

clear internal state

Implements [Controller](#).

6.29.3.3 clone()

```
std::unique_ptr< Controller > PID::clone ( ) const [override], [virtual]
```

virtual clone method

Implements [Controller](#).

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

- lib/UAV_common/src/controllers/impl/[PID.hpp](#)
- lib/UAV_common/src/controllers/impl/[PID.cpp](#)

6.30 controllers::PID_Discrete Class Reference

```
#include <PID_discrete.hpp>
```

Inheritance diagram for controllers::PID_Discrete:

Collaboration diagram for controllers::PID_Discrete:

Public Member Functions

- [PID_Discrete](#) (double Kp, double Ki, double Kd, double Kf=0.0, double N=100.0, double min=-std::numeric_limits< double >::max(), double max=std::numeric_limits< double >::max())
Constructor with all [PID](#) controller parameters.
- [PID_Discrete](#) (rapidxml::xml_node<> *controller_node)
Construct controller with parameters from xml.
- double [calc](#) (double desired, double actual, double dt) override
calc output of controller with specific time step
- void [set_dt](#) (double dt) override
Set new time step.
- void [clear](#) () override
clear internal state
- std::unique_ptr< [Controller](#) > [clone](#) () const override
virtual clone method

Additional Inherited Members

6.30.1 Constructor & Destructor Documentation

6.30.1.1 PID_Discrete() [1/2]

```
controllers::PID_Discrete::PID_Discrete (
    double Kp,
    double Ki,
    double Kd,
    double Kff = 0.0,
    double N = 100.0,
    double min = -std::numeric_limits<double>::max(),
    double max = std::numeric_limits<double>::max() )
```

Constructor with all [PID](#) controller parameters.

Parameters

<i>Kp</i>	P term
<i>Ki</i>	I term
<i>Kd</i>	D term
<i>Kff</i>	FF term
<i>min</i>	saturation - lower range limit
<i>max</i>	saturation - upper range limit
<i>antiWindUp</i>	antiwindup method

6.30.1.2 PID_Discrete() [2/2]

```
controllers::PID_Discrete::PID_Discrete (
    rapidxml::xml_node<> * controller_node )
```

Construct controller with parameters from xml.

Parameters

<i>controller_node</i>	xml node with controller params
------------------------	---------------------------------

6.30.2 Member Function Documentation

6.30.2.1 calc()

```
double controllers::PID_Discrete::calc (
    double desired,
    double actual,
    double dt ) [override], [virtual]
```

calc output of controller with specific time step

Parameters

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value
<i>dt</i>	time step

Returns

output of controller

Implements [Controller](#).

6.30.2.2 clear()

```
void controllers::PID_Discrete::clear ( ) [override], [virtual]
```

clear internal state

Implements [Controller](#).

6.30.2.3 clone()

```
std::unique_ptr< Controller > controllers::PID_Discrete::clone ( ) const [override], [virtual]
```

virtual clone method

Implements [Controller](#).

6.30.2.4 set_dt()

```
void controllers::PID_Discrete::set_dt (
    double dt ) [override], [virtual]
```

Set new time step.

Parameters

<i>dt</i>	new time step
-----------	---------------

Reimplemented from [Controller](#).

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

- [lib/UAV_common/src/controllers/impl/PID_discrete.hpp](#)
- [lib/UAV_common/src/controllers/impl/PID_discrete.cpp](#)

6.31 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](#)

6.31.1 Detailed Description

[Rotor](#) engine with controlled speed.

6.31.2 Member Data Documentation

6.31.2.1 direction

```
int Rotor::direction
```

6.31.2.2 forceCoff

```
double Rotor::forceCoff
```

6.31.2.3 hoverSpeed

```
double Rotor::hoverSpeed
```

6.31.2.4 maxSpeed

```
double Rotor::maxSpeed
```

6.31.2.5 timeConstant

```
double Rotor::timeConstant
```

6.31.2.6 torqueCoff

```
double Rotor::torqueCoff
```

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

- [lib/UAV_common/src/components/drive.hpp](#)

6.32 SensorParams Struct Reference

Base parameters of a sensor.

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

Public Attributes

- `std::string` [name](#)
- `double` [sd](#)
- `Eigen::Vector3d` [bias](#)
- `double` [refreshTime](#)

6.32.1 Detailed Description

Base parameters of a sensor.

6.32.2 Member Data Documentation

6.32.2.1 bias

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

6.32.2.2 name

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

6.32.2.3 refreshTime

```
double SensorParams::refreshTime
```

6.32.2.4 sd

```
double SensorParams::sd
```

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

- [lib/UAV_common/src/components/navi.hpp](#)

6.33 Simulation Class Reference

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

Public Member Functions

- [Simulation](#) ()
Default constructor.
- [~Simulation](#) ()
Destructor.
- void [run](#) ()
Run simulation.

6.33.1 Constructor & Destructor Documentation

6.33.1.1 Simulation()

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

Default constructor.

6.33.1.2 ~Simulation()

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

Destructor.

6.33.2 Member Function Documentation

6.33.2.1 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](#)

6.34 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

6.34.1 Detailed Description

[Simulation](#) of real-time synchronized loop.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 TimedLoop()

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

Constructor.

Parameters

<i>periodInMs</i>	loop period in milliseconds
<i>func</i>	function that should be called in loop
<i>status</i>	reference to controlling status

6.34.3 Member Function Documentation

6.34.3.1 `go()` [1/2]

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

start infinite loop

6.34.3.2 `go()` [2/2]

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

start loop for specific cycle numbers

Parameters

<i>loops</i>	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

6.35 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 [getRotorTimeConstants](#) () 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](#)
- Eigen::Vector3d [target](#)
- double [m](#)
- double [lx](#)
- double [ly](#)
- double [lz](#)
- double [lxy](#)
- double [lxz](#)
- double [lyz](#)
- int [noOfRotors](#)
- std::unique_ptr< [Rotor](#)[]> [rotors](#)
- int [noOfJets](#)
- std::unique_ptr< [Jet](#)[]> [jets](#)
- [ControlSurfaces](#) [surfaces](#)
- [AeroCoefficients](#) [aero_coffs](#)

- `std::map< std::string, std::unique_ptr< Controller > > controllers`
- `std::vector< SensorParams > sensors`
- `AHRSParams ahrs`
- `EKFSalers ekf`
- `Eigen::MatrixX4d rotorMixer`
- `Eigen::MatrixX4d surfaceMixer`
- `int noOfAmmo`
- `std::unique_ptr< Ammo[]> ammo`
- `int noOfCargo`
- `std::unique_ptr< Cargo[]> cargo`

6.35.1 Detailed Description

Parsed UAV configuration from XML.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 UAVparams()

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

Initialize default data.

6.35.2.2 ~UAVparams()

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

6.35.3 Member Function Documentation

6.35.3.1 getRotorHoverSpeeds()

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

6.35.3.2 getRotorMaxSpeeds()

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

6.35.3.3 getRotorTimeContants()

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

6.35.3.4 getSingleton()

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

6.35.3.5 loadConfig()

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

6.35.4 Member Data Documentation

6.35.4.1 aero_coffs

```
AeroCoefficients UAVparams::aero_coffs
```

6.35.4.2 ahrs

```
AHRSParams UAVparams::ahrs
```

6.35.4.3 ammo

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

6.35.4.4 cargo

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

6.35.4.5 controllers

```
std::map<std::string, std::unique_ptr<Controller> > UAVparams::controllers
```

6.35.4.6 ekf

```
EKFScalers UAVparams::ekf
```

6.35.4.7 initialMode

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

6.35.4.8 initialOrientation

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

6.35.4.9 initialPosition

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

6.35.4.10 initialVelocity

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

6.35.4.11 instantRun

```
bool UAVparams::instantRun
```

6.35.4.12 Ix

```
double UAVparams::Ix
```

6.35.4.13 lxy

```
double UAVparams::lxy
```

6.35.4.14 lxz

```
double UAVparams::lxz
```

6.35.4.15 ly

```
double UAVparams::ly
```

6.35.4.16 lyz

```
double UAVparams::lyz
```

6.35.4.17 lz

```
double UAVparams::lz
```

6.35.4.18 jets

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

6.35.4.19 m

```
double UAVparams::m
```

6.35.4.20 name

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


6.35.4.21 noOfAmmo

```
int UAVparams::noOfAmmo
```

6.35.4.22 noOfCargo

```
int UAVparams::noOfCargo
```

6.35.4.23 noOfJets

```
int UAVparams::noOfJets
```

6.35.4.24 noOfRotors

```
int UAVparams::noOfRotors
```

6.35.4.25 rotorMixer

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

6.35.4.26 rotors

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

6.35.4.27 sensors

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

6.35.4.28 surfaceMixer

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

6.35.4.29 surfaces

`ControlSurfaces UAVparams::surfaces`

6.35.4.30 target

`Eigen::Vector3d UAVparams::target`

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`

6.36 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.

6.36.1 Constructor & Destructor Documentation

6.36.1.1 UAVstate()

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

Default constructor.

6.36.1.2 ~UAVstate()

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

Deconstructor.

6.36.2 Member Function Documentation

6.36.2.1 getAcceleration()

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

Returns aircraft acceleration.

Returns

acceleraton vector

6.36.2.2 getDemandedOm()

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

Returns rotor's demanded angular velocities vector.

Returns

rotor's demanded angular velocities vector rad/s

6.36.2.3 getNoOfRotors()

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

Returns number of rotors.

Returns

number of rotors

6.36.2.4 getOm() [1/2]

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

Returns rotor's angular velocities vector.

Returns

rotor's angular velocities vector

6.36.2.5 getOm() [2/2]

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

Return angular velocity vector from given state.

Parameters

<i>state</i>	source state
--------------	--------------

Returns

angular velocity vector

6.36.2.6 getOuterForce()

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

Returns outer force applied to aircraft.

Returns

outer force N

6.36.2.7 getState()

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

Returns raw state vector.

Returns

state vector

6.36.2.8 getX() [1/2]

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

Returns velocity vector X.

Returns

velocity vector X

6.36.2.9 getX() [2/2]

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

Returns velocity vector X from given state.

Parameters

<i>state</i>	source state
--------------	--------------

Returns

velocity vector X

6.36.2.10 getY() [1/2]

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

Returns position vector Y from state (quaterions)

Returns

position vector Y

6.36.2.11 getY() [2/2]

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

Returns position vector Y from given state (quaterions)

Parameters

<i>state</i>	source state
--------------	--------------

Returns

position vector Y

6.36.2.12 operator=()

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

Assigns given raw state vector to state.

6.36.2.13 setAcceleration()

```
void UAVstate::setAcceleration (
    Eigen::Vector< double, 6 > newAcce1 )
```

Set acceleration vector.

Parameters

<i>newAcce1</i>	new acceleration vector
-----------------	-------------------------

6.36.2.14 setDemandedOm()

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

Set demanded angular velocity vector.

Parameters

<i>newOm</i>	new demanded angular velocity vector
--------------	--------------------------------------

6.36.2.15 setForce()

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

Set outer load.

Parameters

<i>force</i>	applied force
<i>torque</i>	applied torque

6.36.2.16 setOm()

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

Set angular velocity vector in given state.

Parameters

<i>state</i>	state that should be updated
<i>Om</i>	new angular velocity vector

6.36.2.17 setStatus()

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

Sets new timed loop status.

Parameters

<i>newStatus</i>	new status
------------------	------------

6.36.2.18 setX() [1/2]

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

Set velocity vector X.

Parameters

<i>newX</i>	new velocity vector X
-------------	-----------------------

6.36.2.19 setX() [2/2]

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

Set velocity vector X in given state.

Parameters

<i>state</i>	state that should be updated
<i>X</i>	new velocity vector X

6.36.2.20 setY()

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

Sets position vector Y in given state (quaterions)

Parameters

<i>state</i>	state that should be updated
<i>Y</i>	new position vector

6.36.3 Friends And Related Function Documentation

6.36.3.1 operator<<

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

Serializes state to stream.

6.36.4 Member Data Documentation

6.36.4.1 real_time

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

simulation time

6.36.4.2 state_mtx

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

state mutex

6.36.4.3 status

`Status` UAVstate::status

Timed loop status.

6.36.4.4 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](#)

6.37 controllers::ZTransform Class Reference

`#include <z_trans.hpp>`

Inheritance diagram for controllers::ZTransform:

Collaboration diagram for controllers::ZTransform:

Public Member Functions

- [ZTransform](#) (const std::vector< double > &num, const std::vector< double > &den, double min=-std::numeric_limits< double >::max(), double max=std::numeric_limits< double >::max())
Constructor of Z-Transform controller.
- [ZTransform](#) (rapidxml::xml_node<> *controller_node)
Construct controller with parameters from xml.
- double [calc](#) (double desired, double actual, [[maybe_unused]] double dt) override
calc output of controller
- void [clear](#) () override
clear internal state
- std::unique_ptr< [Controller](#) > [clone](#) () const override
virtual clone method

Additional Inherited Members

6.37.1 Constructor & Destructor Documentation

6.37.1.1 ZTransform() [1/2]

```
ZTransform::ZTransform (
    const std::vector< double > & num,
    const std::vector< double > & den,
    double min = -std::numeric_limits<double>::max(),
    double max = std::numeric_limits<double>::max() )
```

Constructor of Z-Transform controller.

Parameters

<i>min</i>	saturation - lower range limit
<i>max</i>	saturation - upper range limit

6.37.1.2 ZTransform() [2/2]

```
controllers::ZTransform::ZTransform (
    rapidxml::xml_node<> * controller_node )
```

Construct controller with parameters from xml.

Parameters

<i>controller_node</i>	xml node with controller params
------------------------	---------------------------------

6.37.2 Member Function Documentation**6.37.2.1 calc()**

```
double ZTransform::calc (
    double desired,
    double actual,
    [[maybe_unused] ] double dt ) [override]
```

calc output of controller

Parameters

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value

Returns

output of controller

6.37.2.2 clear()

```
void ZTransform::clear ( ) [override], [virtual]
```

clear internal state

Implements [Controller](#).

6.37.2.3 clone()

```
std::unique_ptr< Controller > ZTransform::clone ( ) const [override], [virtual]
```

virtual clone method

Implements [Controller](#).

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

- [lib/UAV_common/src/controllers/impl/z_trans.hpp](#)
- [lib/UAV_common/src/controllers/impl/z_trans.cpp](#)

6.38 controllers::ZTransformStatic< N, D > Class Template Reference

```
#include <z_trans.hpp>
```

Inheritance diagram for controllers::ZTransformStatic< N, D >:

Collaboration diagram for controllers::ZTransformStatic< N, D >:

Public Member Functions

- [ZTransformStatic](#) (const std::array< double, N > &num, const std::array< double, D > &den, double min=-std::numeric_limits< double >::max(), double max=std::numeric_limits< double >::max())
Constructor of Z-Transform controller.
- [ZTransformStatic](#) (rapidxml::xml_node<> *controller_node)=delete
Construct controller with parameters from xml.
- double [calc](#) (double desired, double actual, [[maybe_unused]] double dt) override
calc output of controller
- void [clear](#) () override
clear internal state
- std::unique_ptr< [Controller](#) > [clone](#) () const override
virtual clone method

Additional Inherited Members

6.38.1 Constructor & Destructor Documentation

6.38.1.1 ZTransformStatic() [1/2]

```
template<unsigned int N, unsigned int D>
controllers::ZTransformStatic< N, D >::ZTransformStatic (
    const std::array< double, N > & num,
    const std::array< double, D > & den,
    double min = -std::numeric_limits<double>::max(),
    double max = std::numeric_limits<double>::max() )
```

Constructor of Z-Transform controller.

Parameters

<i>min</i>	saturation - lower range limit
<i>max</i>	saturation - upper range limit

6.38.1.2 ZTransformStatic() [2/2]

```
template<unsigned int N, unsigned int D>
controllers::ZTransformStatic< N, D >::ZTransformStatic (
    rapidxml::xml_node<> * controller_node ) [delete]
```

Construct controller with parameters from xml.

Parameters

<i>controller_node</i>	xml node with controller params
------------------------	---------------------------------

6.38.2 Member Function Documentation

6.38.2.1 calc()

```
template<unsigned int N, unsigned int D>
double controllers::ZTransformStatic< N, D >::calc (
    double desired,
    double actual,
    [[maybe_unused]] double dt ) [override]
```

calc output of controller

Parameters

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value

Returns

output of controller

6.38.2.2 clear()

```
template<unsigned int N, unsigned int D>
void controllers::ZTransformStatic< N, D >::clear [override], [virtual]
```

clear internal state

Implements [Controller](#).

6.38.2.3 clone()

```
template<unsigned int N, unsigned int D>
std::unique_ptr< Controller > controllers::ZTransformStatic< N, D >::clone [override], [virtual]
```

virtual clone method

Implements [Controller](#).

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

- [lib/UAV_common/src/controllers/impl/z_trans.hpp](#)

Chapter 7

File Documentation

7.1 build/CMakeFiles/3.22.1/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- `#define __has_include(x) 0`
- `#define COMPILER_ID ""`
- `#define STRINGIFY_HELPER(X) #X`
- `#define STRINGIFY(X) STRINGIFY_HELPER(X)`
- `#define PLATFORM_ID`
- `#define ARCHITECTURE_ID`
- `#define DEC(n)`
- `#define HEX(n)`
- `#define C_VERSION`

Functions

- `int main (int argc, char *argv[])`

Variables

- `char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"`
- `char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"`
- `char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"`
- `const char * info_language_standard_default`
- `const char * info_language_extensions_default`

7.1.1 Macro Definition Documentation

7.1.1.1 `__has_include`

```
#define __has_include(  
    x ) 0
```

7.1.1.2 `ARCHITECTURE_ID`

```
#define ARCHITECTURE_ID
```

7.1.1.3 `C_VERSION`

```
#define C_VERSION
```

7.1.1.4 `COMPILER_ID`

```
#define COMPILER_ID ""
```

7.1.1.5 `DEC`

```
#define DEC(  
    n )
```

Value:

```
('0' + ((n) / 10000000) % 10), \
('0' + ((n) / 1000000) % 10), \
('0' + ((n) / 100000) % 10), \
('0' + ((n) / 10000) % 10), \
('0' + ((n) / 1000) % 10), \
('0' + ((n) / 100) % 10), \
('0' + ((n) / 10) % 10), \
('0' + ((n) % 10))
```

7.1.1.6 `HEX`

```
#define HEX(  
    n )
```

Value:

```
('0' + ((n) >> 28 & 0xF)), \
('0' + ((n) >> 24 & 0xF)), \
('0' + ((n) >> 20 & 0xF)), \
('0' + ((n) >> 16 & 0xF)), \
('0' + ((n) >> 12 & 0xF)), \
('0' + ((n) >> 8 & 0xF)), \
('0' + ((n) >> 4 & 0xF)), \
('0' + ((n) & 0xF))
```


7.1.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

7.1.1.8 STRINGIFY

```
#define STRINGIFY(  
    X ) STRINGIFY_HELPER(X)
```

7.1.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(  
    X ) #X
```

7.1.2 Function Documentation

7.1.2.1 main()

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

7.1.3 Variable Documentation

7.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

7.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

7.1.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["  
  "OFF"  
"]"
```

7.1.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
=  
  "INFO" ":" "standard_default[" C_VERSION "]"
```

7.1.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

7.2 build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- #define `__has_include(x)` 0
- #define `COMPILER_ID` ""
- #define `STRINGIFY_HELPER(X)` #X
- #define `STRINGIFY(X)` `STRINGIFY_HELPER(X)`
- #define `PLATFORM_ID`
- #define `ARCHITECTURE_ID`
- #define `DEC(n)`
- #define `HEX(n)`
- #define `CXX_STD` `__cplusplus`

Functions

- int `main` (int argc, char *argv[])

Variables

- char const * `info_compiler` = "INFO" ":" "compiler[" `COMPILER_ID` "]"
- char const * `info_platform` = "INFO" ":" "platform[" `PLATFORM_ID` "]"
- char const * `info_arch` = "INFO" ":" "arch[" `ARCHITECTURE_ID` "]"
- const char * `info_language_standard_default`
- const char * `info_language_extensions_default`

7.2.1 Macro Definition Documentation

7.2.1.1 `__has_include`

```
#define __has_include(  
    x ) 0
```

7.2.1.2 `ARCHITECTURE_ID`

```
#define ARCHITECTURE_ID
```

7.2.1.3 `COMPILER_ID`

```
#define COMPILER_ID ""
```

7.2.1.4 `CXX_STD`

```
#define CXX_STD __cplusplus
```

7.2.1.5 `DEC`

```
#define DEC(  
    n )
```

Value:

```
('0' + ((n) / 10000000) % 10), \  
( '0' + ((n) / 1000000) % 10), \  
( '0' + ((n) / 100000) % 10), \  
( '0' + ((n) / 10000) % 10), \  
( '0' + ((n) / 1000) % 10), \  
( '0' + ((n) / 100) % 10), \  
( '0' + ((n) / 10) % 10), \  
( '0' + ((n) % 10))
```

7.2.1.6 HEX

```
#define HEX(  
    n )
```

Value:

```
('0' + ((n)>>28 & 0xF)), \  
( '0' + ((n)>>24 & 0xF)), \  
( '0' + ((n)>>20 & 0xF)), \  
( '0' + ((n)>>16 & 0xF)), \  
( '0' + ((n)>>12 & 0xF)), \  
( '0' + ((n)>>8  & 0xF)), \  
( '0' + ((n)>>4  & 0xF)), \  
( '0' + ((n)    & 0xF))
```

7.2.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

7.2.1.8 STRINGIFY

```
#define STRINGIFY(  
    X ) STRINGIFY\_HELPER(X)
```

7.2.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(  
    X ) #X
```

7.2.2 Function Documentation

7.2.2.1 main()

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

7.2.3 Variable Documentation

7.2.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

7.2.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

7.2.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["  
  "OFF"  
"]"
```

7.2.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
= "INFO" ":" "standard_default["  
  "98"  
"]"
```

7.2.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```


7.3 build/CMakeFiles/uav.dir/src/aircraft/aircraft.cpp.o.d File Reference**7.4 build/CMakeFiles/uav.dir/src/aircraft/aircraft_comm.cpp.o.d File Reference****7.5 build/CMakeFiles/uav.dir/src/aircraft/aircraft_impulse.cpp.o.d File Reference****7.6 build/CMakeFiles/uav.dir/src/dynamic/forces.cpp.o.d File Reference****7.7 build/CMakeFiles/uav.dir/src/dynamic/matrices.cpp.o.d File Reference****7.8 build/CMakeFiles/uav.dir/src/main.cpp.o.d File Reference****7.9 build/CMakeFiles/uav.dir/src/params.cpp.o.d File Reference****7.10 build/CMakeFiles/uav.dir/src/simulation/atmosphere.cpp.o.d File Reference****7.11 build/CMakeFiles/uav.dir/src/simulation/control.cpp.o.d File Reference****7.12 build/CMakeFiles/uav.dir/src/simulation/simulation.cpp.o.d File Reference****7.13 build/CMakeFiles/uav.dir/src/simulation/uav_state.cpp.o.d File Reference****7.14 build/lib/UAV_common/CMakeFiles/common.dir/src/components/control_surfaces.cpp.o.d File Reference****7.15 build/lib/UAV_common/CMakeFiles/common.dir/src/components/drive.cpp.o.d File Reference****7.16 build/lib/UAV_common/CMakeFiles/common.dir/src/components/hinge.cpp.o.d File Reference****7.17 build/lib/UAV_common/CMakeFiles/common.dir/src/components/loads.cpp.o.d File Reference****7.18 build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/controller.cpp.o.d File Reference****7.19 build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/controller.cpp.o.d File Reference**

```
#include "../src/ode/ode.hpp"
#include "../src/controllers/controller.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:

7.31 lib/UAV_common/scripts/controller_plots.m File Reference

Functions

- `plot` (`x`, `y`, 'DisplayName', `csvFiles(i).name`)
- `end xlabel` ('Czas')
- `ylabel` ('Wartość regulowana')
- `title` ('Test regulatorów')
- `legend` ('Location', 'Best')

Variables

- `clc`
- `clear folderPath` = '../build/controller_plots/'
- `csvFiles` = `dir(fullfile(folderPath, '*.csv'))`
- `figure`
- `hold on`
- `for i`
- `data` = `readmatrix(filePath)`
- `x` = `data(:, 1)`
- `y` = `data(:, 2)`
- `hold off`

7.31.1 Function Documentation

7.31.1.1 `legend()`

```
legend (
    'Location' ,
    'Best' )
```


7.31.1.2 plot()

```
plot (
    x ,
    y ,
    'DisplayName' ,
    csvFiles(i). name )
```

7.31.1.3 title()

```
title (
    'Test regulatorów' )
```

7.31.1.4 xlabel()

```
end xlabel (
    'Czas' )
```

7.31.1.5 ylabel()

```
ylabel (
    'Wartość regulowana' )
```

7.31.2 Variable Documentation

7.31.2.1 clc

```
clc
```

7.31.2.2 csvFiles

```
csvFiles = dir(fullfile(folderPath, '*.csv'))
```

7.31.2.3 data

```
data = readmatrix(filePath)
```

7.31.2.4 figure

```
figure
```

7.31.2.5 folderPath

```
clear folderPath = '../build/controller_plots/'
```

7.31.2.6 i

```
for i
```

Initial value:

```
= 1:length(csvFiles)  
    filePath = fullfile(folderPath, csvFiles(i).name)
```

7.31.2.7 off

```
hold off
```

7.31.2.8 on

```
hold on
```

7.31.2.9 x

```
x = data(:, 1)
```

7.31.2.10 y

```
y = data(:, 2)
```

7.32 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.

7.33 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:

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

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

Include dependency graph for `control_surfaces.cpp`:

7.35 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.

7.36 lib/UAU_common/src/components/drive.cpp File Reference

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

Include dependency graph for drive.cpp:

7.37 lib/UAU_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.

7.38 lib/UAU_common/src/components/hinge.cpp File Reference

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

Include dependency graph for hinge.cpp:

Functions

- [Eigen::Matrix3d asSkewMatrix](#) ([Eigen::Vector3d](#) v)

7.38.1 Function Documentation

7.38.1.1 asSkewMatrix()

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

7.39 lib/UAU_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.

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

```
#include "loads.hpp"
#include <limits>
Include dependency graph for loads.cpp:
```

7.41 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 dropped or launched.
- class [Ammo](#)
- class [Cargo](#)

7.42 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.

7.43 lib/UAV_common/src/controllers/controller.cpp File Reference

```
#include "controller.hpp"
#include "impl/PID.hpp"
#include "impl/PID_discrete.hpp"
#include "impl/bang_bang.hpp"
#include "impl/double_setpoint.hpp"
#include "impl/z_trans.hpp"
#include <cstring>
#include <stdexcept>
Include dependency graph for controller.cpp:
```

7.44 lib/UAV_common/src/controllers/controller.hpp File Reference

```
#include <memory>
#include "rapidxml/rapidxml.hpp"
```

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

Classes

- class [Controller](#)

7.45 lib/UAV_common/src/controllers/controller_test.cpp File Reference

```
#include "impl/PID.hpp"
#include "impl/PID_discrete.hpp"
#include "impl/bang_bang.hpp"
#include "impl/double_setpoint.hpp"
#include "impl/z_trans.hpp"
#include <gtest/gtest.h>
#include <memory>
#include <filesystem>
#include <fstream>
```

Include dependency graph for controller_test.cpp:

Classes

- class [ControllerTest](#)

Functions

- `std::vector< std::shared_ptr< Controller > > getMethodsToTest ()`
- `TEST_P (ControllerTest, TestConstFunction)`
- `TEST_P (ControllerTest, SimpleObjectControl)`
- `INSTANTIATE_TEST_SUITE_P (TestDerivedClasses, ControllerTest, testing::ValuesIn(getMethodsToTest()))`
- `int main (int argc, char **argv)`

Variables

- `constexpr bool plot = true`
- `constexpr auto plot_directory_name = "controller_plots"`

7.45.1 Function Documentation

7.45.1.1 getMethodsToTest()

```
std::vector<std::shared_ptr<Controller> > getMethodsToTest ( )
```

7.45.1.2 INSTANTIATE_TEST_SUITE_P()

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

7.45.1.3 main()

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

7.45.1.4 TEST_P() [1/2]

```
TEST_P (
    ControllerTest ,
    SimpleObjectControl )
```

7.45.1.5 TEST_P() [2/2]

```
TEST_P (
    ControllerTest ,
    TestConstFunction )
```

7.45.2 Variable Documentation

7.45.2.1 plot

```
constexpr bool plot = true [constexpr]
```

7.45.2.2 `plot_directory_name`

```
constexpr auto plot_directory_name = "controller_plots" [constexpr]
```

7.46 `lib/UAV_common/src/controllers/impl/bang_bang.cpp` File Reference

```
#include "bang_bang.hpp"  
#include <cstring>  
#include <string>  
Include dependency graph for bang_bang.cpp:
```

7.47 `lib/UAV_common/src/controllers/impl/bang_bang.hpp` File Reference

```
#include <memory>  
#include "rapidxml/rapidxml.hpp"  
#include "../controller.hpp"  
Include dependency graph for bang_bang.hpp: This graph shows which files directly or indirectly include this file:
```

Classes

- class [controllers::BangBang](#)

Namespaces

- [controllers](#)

7.48 `lib/UAV_common/src/controllers/impl/double_setpoint.cpp` File Reference

```
#include "double_setpoint.hpp"  
#include <cstring>  
#include <string>  
Include dependency graph for double_setpoint.cpp:
```

7.49 `lib/UAV_common/src/controllers/impl/double_setpoint.hpp` File Reference

```
#include <memory>  
#include "rapidxml/rapidxml.hpp"  
#include "../controller.hpp"  
Include dependency graph for double_setpoint.hpp: This graph shows which files directly or indirectly include this file:
```


Classes

- class [controllers::DoubleSetpoint](#)

Namespaces

- [controllers](#)

7.50 lib/UAV_common/src/controllers/impl/PID.cpp File Reference

```
#include "PID.hpp"
#include <algorithm>
#include <cstring>
#include <string>
#include <stdexcept>
Include dependency graph for PID.cpp:
```

7.51 lib/UAV_common/src/controllers/impl/PID.hpp File Reference

```
#include <memory>
#include <limits>
#include "rapidxml/rapidxml.hpp"
#include "../controller.hpp"
#include "z_trans.hpp"
Include dependency graph for PID.hpp: This graph shows which files directly or indirectly include this file:
```

Classes

- class [controllers::PID](#)

Namespaces

- [controllers](#)

7.52 lib/UAV_common/src/controllers/impl/PID_discrete.cpp File Reference

```
#include "PID_discrete.hpp"
#include <iostream>
#include <string>
#include <cstring>
Include dependency graph for PID_discrete.cpp:
```

7.53 lib/UAV_common/src/controllers/impl/PID_discrete.hpp File Reference

```
#include <memory>
#include <limits>
#include "rapidxml/rapidxml.hpp"
#include "../controller.hpp"
#include "z_trans.hpp"
```

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

Classes

- class [controllers::PID_Discrete](#)

Namespaces

- [controllers](#)

7.54 lib/UAV_common/src/controllers/impl/z_trans.cpp File Reference

```
#include "z_trans.hpp"
#include <sstream>
#include <iterator>
#include <string>
#include <cstring>
```

Include dependency graph for z_trans.cpp:

Functions

- `std::vector< double > splitStringToDoubleVector (const std::string &input)`

7.54.1 Function Documentation

7.54.1.1 `splitStringToDoubleVector()`

```
std::vector<double> splitStringToDoubleVector (
    const std::string & input )
```

7.55 lib/UAV_common/src/controllers/impl/z_trans.hpp File Reference

```
#include <memory>
#include <limits>
#include <array>
#include <vector>
#include <algorithm>
#include <numeric>
#include <stdexcept>
#include <cassert>
#include "rapidxml/rapidxml.hpp"
#include "../controller.hpp"
```

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

Classes

- class [controllers::ZTransformStatic< N, D >](#)
- class [controllers::ZTransform](#)

Namespaces

- [controllers](#)

7.56 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)

7.56.1 Function Documentation

7.56.1.1 shouldLog()

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

7.57 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

7.57.1 Macro Definition Documentation

7.57.1.1 [LOGGER_MASK](#)

```
#define LOGGER\_MASK -1
```

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

```
#include "ode.hpp"
#include "ode_impl.hpp"
Include dependency graph for ode.cpp:
```

7.59 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.

7.60 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.
- class [ODE_PC2](#)
Second order predictor-corrector method Second order Adams-bashforth and Adams-moulton.
- class [ODE_PC4](#)
Fourth order predictor-corrector method Fourth order Adams-bashforth and Adams-moulton.

7.61 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)`
- `TEST_P (ODETest, TestHarmonicOscillator)`
- `INSTANTIATE_TEST_SUITE_P (TestDerivedClasses, ODETest, testing::ValuesIn(getMethodsToTest()))`
- `int main (int argc, char **argv)`

7.61.1 Function Documentation

7.61.1.1 getMethodsToTest()

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

7.61.1.2 INSTANTIATE_TEST_SUITE_P()

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

7.61.1.3 main()

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

7.61.1.4 TEST_F() [1/2]

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

7.61.1.5 TEST_F() [2/2]

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

7.61.1.6 TEST_P() [1/4]

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

7.61.1.7 TEST_P() [2/4]

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

7.61.1.8 TEST_P() [3/4]

```
TEST_P (
    ODETest ,
    TestHarmonicOscillator )
```

7.61.1.9 TEST_P() [4/4]

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

7.62 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.

7.62.1 Function Documentation**7.62.1.1 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

<i>input</i>	input string
<i>R</i>	number of rows
<i>C</i>	number of columns
<i>delimiter</i>	delimiter

Returns

parsed matrix

7.62.1.2 parseVectorXd()

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

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

Parameters

<i>str</i>	input string
<i>noOfElem</i>	length of vector
<i>delimiter</i>	delimiter

Returns

parsed vector

7.63 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.

7.63.1 Function Documentation

7.63.1.1 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

<i>input</i>	input string
<i>R</i>	number of rows
<i>C</i>	number of columns
<i>delimiter</i>	delimiter

Returns

parsed matrix

7.63.1.2 parseVectorXd()

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

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

Parameters

<i>str</i>	input string
<i>noOfElem</i>	length of vector
<i>delimiter</i>	delimiter

Returns

parsed vector

7.64 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)

7.64.1 Function Documentation

7.64.1.1 [parseHinge\(\)](#)

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

7.65 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 "../controllers/controller.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.

7.66 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

7.66.1 Enumeration Type Documentation

7.66.1.1 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

7.67 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:
```

7.68 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.

7.69 src/aircraft/aircraft.cpp File Reference

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

Functions

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

7.69.1 Function Documentation

7.69.1.1 clampOrientationIfNecessary()

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

7.70 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

7.71 src/aircraft/aircraft_comm.cpp File Reference

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

Include dependency graph for aircraft_comm.cpp:

7.72 src/aircraft/aircraft_impulse.cpp File Reference

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

Include dependency graph for aircraft_impulse.cpp:

7.73 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::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

7.73.1 Macro Definition Documentation

7.73.1.1 USE_QUATERIONS

```
#define USE_QUATERIONS 1
```

define to use quaterion instead of RPY angles

7.74 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:

7.75 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](#)

7.76 src/dynamic/matrices.cpp File Reference

```
#include <Eigen/Dense>
#include <cmath>
#include "matrices.hpp"
#include "common.hpp"
Include dependency graph for matrices.cpp:
```

7.77 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](#)

7.78 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 "params.hpp"
Include dependency graph for main.cpp:
```

Functions

- void [parseArgs](#) (int argc, char **argv, [UAVparams](#) *params, [Params](#) &p)
Parse CL arguments.
- int [main](#) (int argc, char **argv)

7.78.1 Function Documentation

7.78.1.1 main()

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

7.78.1.2 parseArgs()

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

Parse CL arguments.

Parameters

<i>argc</i>	number of argument
<i>argv</i>	argument array
<i>params</i>	pointer to UAVparams instant that should be filled
<i>p</i>	internal params reference

7.79 src/params.cpp File Reference

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

Include dependency graph for params.cpp:

7.80 src/params.hpp File Reference

```
#include <string>
```

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

Classes

- class [Params](#)
Simulation parameters.

7.81 src/simulation/atmosphere.cpp File Reference

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

Include dependency graph for atmosphere.cpp:

7.82 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.

7.83 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.

7.83.1 Function Documentation

7.83.1.1 control()

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

7.83.1.2 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

<i>ctx</i>	zero mq context
<i>address</i>	address of listener socket
<i>aircraft</i>	pointer to aircraft

7.83.1.3 dropCargo()

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

7.83.1.4 isNormal()

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

7.83.1.5 setAtmosphere()

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

7.83.1.6 setControlSurface()

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

7.83.1.7 setForce()

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

7.83.1.8 setHinges()

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

7.83.1.9 setSpeed()

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

7.83.1.10 shoot()

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

7.83.1.11 solidSurfColision()

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

7.83.1.12 startJet()

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

7.84 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.

7.84.1 Function Documentation

7.84.1.1 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

<i>ctx</i>	zero mq context
<i>address</i>	address of listener socket
<i>aircraft</i>	pointer to aircraft

7.85 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 "../params.hpp"
Include dependency graph for simulation.cpp:
```

7.86 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](#)

7.87 src/simulation/uav_state.cpp File Reference

```
#include "uav_state.hpp"
#include "../dynamic/matrices.hpp"
#include "../params.hpp"
Include dependency graph for uav_state.cpp:
```

Functions

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

7.87.1 Function Documentation

7.87.1.1 `operator<<()`

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

7.88 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](#)

7.89 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](#) ()

7.89.1 Function Documentation

7.89.1.1 fun()

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

7.89.1.2 test1()

```
int test1 ( )
```

7.90 tests/test2.cpp File Reference

```
#include <iostream>
#include <Eigen/Dense>
#include "constants.hpp"
Include dependency graph for test2.cpp:
```

Functions

- int [main](#) ()

7.90.1 Function Documentation

7.90.1.1 main()

```
int main ( )
```

7.91 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](#) ()

7.91.1 Function Documentation

7.91.1.1 fun()

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

7.91.1.2 main()

```
int main ( )
```

7.92 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](#) ()

7.92.1 Function Documentation

7.92.1.1 main()

```
int main ( )
```

7.93 tests/test5.cpp File Reference

```
#include <iostream>
#include <thread>
#include <chrono>
#include "timed_loop.hpp"
Include dependency graph for test5.cpp:
```

Functions

- int [main](#) ()

7.93.1 Function Documentation

7.93.1.1 main()

```
int main ( )
```


7.94 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()

7.95 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()

7.96 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

7.97 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, 29](#)
 - [__has_include](#)
 - [CMakeCCompilerId.c, 107](#)
 - [CMakeCXXCompilerId.cpp, 111](#)
 - [_dt](#)
 - [Controller, 39](#)
 - [~Aircraft](#)
 - [Aircraft, 21](#)
 - [~Atmosphere](#)
 - [Atmosphere, 30](#)
 - [~Controller](#)
 - [Controller, 37](#)
 - [~Logger](#)
 - [Logger, 58](#)
 - [~ODE](#)
 - [ODE, 65](#)
 - [~Params](#)
 - [Params, 76](#)
 - [~Simulation](#)
 - [Simulation, 86](#)
 - [~UAVparams](#)
 - [UAVparams, 89](#)
 - [~UAVstate](#)
 - [UAVstate, 95](#)
- [aero](#)
 - [Aircraft, 26](#)
- [aero_coffs](#)
 - [UAVparams, 90](#)
- [AeroCoefficients, 17](#)
 - [C0, 17](#)
 - [Cab, 17](#)
 - [Cpqr, 18](#)
 - [d, 18](#)
 - [eAR, 18](#)
 - [S, 18](#)
 - [stallLimit, 18](#)
- [aerodynamic_loads](#)
 - [Forces, 47](#)
- [ahrs](#)
 - [UAVparams, 90](#)
- [AHRSParams, 18](#)
 - [alpha, 19](#)
 - [Q, 19](#)
 - [R, 19](#)
 - [type, 19](#)
- [air_density](#)
 - [AtmosphereInfo, 33](#)
- [air_pressure](#)
 - [AtmosphereInfo, 33](#)
- [air_temperature](#)
 - [AtmosphereInfo, 33](#)
- [Aircraft, 20](#)
 - [~Aircraft, 21](#)
 - [aero, 26](#)
 - [Aircraft, 21](#)
 - [ammo, 26](#)
 - [calcImpulseForce, 21](#)
 - [calcMomentumConservanceConservation, 22](#)
 - [cargo, 26](#)
 - [dropCargo, 22](#)
 - [invMassMatrix, 26](#)
 - [jets, 26](#)
 - [massMatrix, 26](#)
 - [mtx, 26](#)
 - [noOfAmmo, 27](#)
 - [noOfCargo, 27](#)
 - [noOfJets, 27](#)
 - [noOfRotors, 27](#)
 - [ode, 27](#)
 - [reduceMass, 23](#)
 - [RHS, 23](#)
 - [rotors, 27](#)
 - [sendState, 23](#)
 - [setHinge, 24](#)
 - [setSurface, 24](#)
 - [shootAmmo, 24](#)
 - [startJet, 25](#)
 - [state, 27](#)
 - [surfaces, 27](#)
 - [trim, 25](#)
 - [update, 25](#)
- [aircraft.cpp](#)
 - [clampOrientationIfNecessery, 136](#)
- [alpha](#)
 - [AHRSParams, 19](#)
- [Ammo, 28](#)
 - [_V0, 29](#)
 - [Ammo, 28](#)
 - [getV0, 29](#)
 - [operator=, 29](#)
- [ammo](#)
 - [Aircraft, 26](#)
 - [UAVparams, 90](#)
- [angle](#)
 - [zmq_send_tcp, 15](#)
- [angularAcceleration](#)
 - [Forces, 47](#)

- Controller, 38
- controllers::BangBang, 35
- controllers::DoubleSetpoint, 44
- controllers::PID, 80
- controllers::PID_Discrete, 82
- controllers::ZTransform, 103
- controllers::ZTransformStatic< N, D >, 105
- clone
 - Controller, 38
 - controllers::BangBang, 35
 - controllers::DoubleSetpoint, 44
 - controllers::PID, 80
 - controllers::PID_Discrete, 82
 - controllers::ZTransform, 104
 - controllers::ZTransformStatic< N, D >, 106
- CMakeCCompilerId.c
 - __has_include, 107
 - ARCHITECTURE_ID, 108
 - C_VERSION, 108
 - COMPILER_ID, 108
 - DEC, 108
 - HEX, 108
 - info_arch, 109
 - info_compiler, 109
 - info_language_extensions_default, 109
 - info_language_standard_default, 110
 - info_platform, 110
 - main, 109
 - PLATFORM_ID, 108
 - STRINGIFY, 109
 - STRINGIFY_HELPER, 109
- CMakeCXXCompilerId.cpp
 - __has_include, 111
 - ARCHITECTURE_ID, 111
 - COMPILER_ID, 111
 - CXX_STD, 111
 - DEC, 111
 - HEX, 111
 - info_arch, 112
 - info_compiler, 113
 - info_language_extensions_default, 113
 - info_language_standard_default, 113
 - info_platform, 113
 - main, 112
 - PLATFORM_ID, 112
 - STRINGIFY, 112
 - STRINGIFY_HELPER, 112
- COMPILER_ID
 - CMakeCCompilerId.c, 108
 - CMakeCXXCompilerId.cpp, 111
- context
 - zmq_recv, 13
 - zmq_recv_last, 14
 - zmq_send, 14
 - zmq_send_tcp, 15
- control
 - control.cpp, 141
- control.cpp
 - control, 141
 - controlListenerJob, 142
 - dropCargo, 142
 - isNormal, 142
 - setAtmosphere, 142
 - setControlSurface, 142
 - setForce, 143
 - setHinges, 143
 - setSpeed, 143
 - shoot, 143
 - solidSurfCollision, 143
 - startJet, 143
- control.hpp
 - controlListenerJob, 144
- Controller, 36
 - _dt, 39
 - ~Controller, 37
 - calc, 37, 38
 - clear, 38
 - clone, 38
 - Controller, 37
 - ControllerFactory, 38
 - set_dt, 39
- controller_plots.m
 - clc, 117
 - csvFiles, 117
 - data, 117
 - figure, 118
 - folderPath, 118
 - i, 118
 - legend, 116
 - off, 118
 - on, 118
 - plot, 116
 - title, 117
 - x, 118
 - xlabel, 117
 - y, 118
 - ylabel, 117
- controller_test.cpp
 - getMethodsToTest, 122
 - INstantiate_TEST_SUITE_P, 123
 - main, 123
 - plot, 123
 - plot_directory_name, 123
 - TEST_P, 123
- ControllerFactory
 - Controller, 38
- controllers, 11
 - UAVparams, 90
- controllers::BangBang, 34
 - BangBang, 34
 - calc, 35
 - clear, 35
 - clone, 35
- controllers::DoubleSetpoint, 42
 - calc, 43
 - clear, 44

- clone, 44
- DoubleSetpoint, 42, 43
- controllers::PID, 77
 - AntiWindUpMode, 78
 - calc, 79
 - CLAMPING, 78
 - clear, 80
 - clone, 80
 - NONE, 78
 - PID, 78, 79
- controllers::PID_Discrete, 80
 - calc, 81
 - clear, 82
 - clone, 82
 - PID_Discrete, 81
 - set_dt, 82
- controllers::ZTransform, 102
 - calc, 103
 - clear, 103
 - clone, 104
 - ZTransform, 102, 103
- controllers::ZTransformStatic< N, D >, 104
 - calc, 105
 - clear, 105
 - clone, 106
 - ZTransformStatic, 104, 105
- ControllerTest, 39
 - SetUp, 40
 - TearDown, 40
- controlListenerJob
 - control.cpp, 142
 - control.hpp, 144
- ControlSurfaces, 40
 - ControlSurfaces, 41
 - getCoefficients, 41
 - getNoOfSurface, 41
 - getValues, 41
 - restoreTrim, 42
 - setValues, 42
- counter
 - zmq_send, 14
- Cpqr
 - AeroCoefficients, 18
- csvFiles
 - controller_plots.m, 117
- CXX_STD
 - CMakeCXXCompilerId.cpp, 111
- d
 - AeroCoefficients, 18
- data
 - controller_plots.m, 117
- DEC
 - CMakeCCompilerId.c, 108
 - CMakeCXXCompilerId.cpp, 111
- def, 11
 - DOUBLE_EPS, 12
 - FRICTION_EPS, 12
 - GENTLY_PUSH, 12
 - GRAVITY_CONST, 12
 - MIXING_FUNCTION, 12
 - validityOfForce, 12
- defines.hpp
 - USE_QUATERIONS, 138
- direction
 - Rotor, 83
- DOUBLE_EPS
 - def, 12
- DoubleSetpoint
 - controllers::DoubleSetpoint, 42, 43
- Drive, 44
 - axis, 45
 - hinges, 45
 - noOfHinges, 45
 - position, 45
- dropCargo
 - Aircraft, 22
 - control.cpp, 142
- eAR
 - AeroCoefficients, 18
- ekf
 - UAVparams, 91
- EKFScalers, 45
 - baroScaler, 46
 - predictScaler, 46
 - updateScaler, 46
 - zScaler, 46
- Euler
 - ODE, 65
- exiting
 - status.hpp, 136
- factory
 - ODE, 66
- figure
 - controller_plots.m, 118
- folderPath
 - controller_plots.m, 118
- forceCoff
 - Rotor, 83
- Forces, 46
 - aerodynamic_loads, 47
 - angularAcceleration, 47
 - generateCharacteristics, 49
 - gravity_loads, 49
 - jet_lift_loads, 49
 - rotor_lift_loads, 50
- FRICTION_EPS
 - def, 12
- fromString
 - ODE, 66
- fun
 - test1.cpp, 146
 - test3.cpp, 147
- generateCharacteristics
 - Forces, 49

- GENTLY_PUSH
 - def, [12](#)
- getAcceleration
 - UAVstate, [96](#)
- getAirDensity
 - Atmosphere, [31](#)
- getAirPressure
 - Atmosphere, [31](#)
- getAirTemperature
 - Atmosphere, [31](#)
- getAmmount
 - Load, [56](#)
- getCoefficients
 - ControlSurfaces, [41](#)
- getDemandedOm
 - UAVstate, [96](#)
- getLastThrust
 - Jet, [53](#)
- getMass
 - Load, [56](#)
- getMethodsToTest
 - controller_test.cpp, [122](#)
 - ode_test.cpp, [129](#)
- getMicrosteps
 - ODE, [66](#), [67](#)
- getNoOfRotors
 - UAVstate, [96](#)
- getNoOfSurface
 - ControlSurfaces, [41](#)
- getOffset
 - Load, [56](#)
- getOm
 - UAVstate, [96](#)
- getOuterForce
 - UAVstate, [97](#)
- getRot
 - Hinge, [51](#)
- getRotorHoverSpeeds
 - UAVparams, [89](#)
- getRotorMaxSpeeds
 - UAVparams, [89](#)
- getRotorTimeContants
 - UAVparams, [89](#)
- getSingleton
 - Atmosphere, [31](#)
 - Params, [76](#)
 - UAVparams, [90](#)
- getState
 - UAVstate, [97](#)
- getThrust
 - Jet, [53](#)
- getV0
 - Ammo, [29](#)
- getValues
 - ControlSurfaces, [41](#)
- getWind
 - Atmosphere, [32](#)
- getX
 - UAVstate, [97](#)
- getY
 - UAVstate, [98](#)
- go
 - TimedLoop, [87](#)
- GRAVITY_CONST
 - def, [12](#)
- gravity_loads
 - Forces, [49](#)
- gyroMatrix
 - Matrices, [60](#)
- Heun
 - ODE, [65](#)
- HEX
 - CMakeCCompilerId.c, [108](#)
 - CMakeCXXCompilerId.cpp, [111](#)
- Hinge, [50](#)
 - getRot, [51](#)
 - Hinge, [51](#)
 - operator=, [52](#)
 - updateValue, [52](#)
- hinge.cpp
 - asSkewMatrix, [120](#)
- hinges
 - Drive, [45](#)
- hoverSpeed
 - Rotor, [84](#)
- i
 - controller_plots.m, [118](#)
- idle
 - status.hpp, [136](#)
- info_arch
 - CMakeCCompilerId.c, [109](#)
 - CMakeCXXCompilerId.cpp, [112](#)
- info_compiler
 - CMakeCCompilerId.c, [109](#)
 - CMakeCXXCompilerId.cpp, [113](#)
- info_language_extensions_default
 - CMakeCCompilerId.c, [109](#)
 - CMakeCXXCompilerId.cpp, [113](#)
- info_language_standard_default
 - CMakeCCompilerId.c, [110](#)
 - CMakeCXXCompilerId.cpp, [113](#)
- info_platform
 - CMakeCCompilerId.c, [110](#)
 - CMakeCXXCompilerId.cpp, [113](#)
- initialMode
 - UAVparams, [91](#)
- initialOrientation
 - UAVparams, [91](#)
- initialPosition
 - UAVparams, [91](#)
- initialVelocity
 - UAVparams, [91](#)
- INstantiate_TEST_SUITE_P
 - controller_test.cpp, [123](#)
 - ode_test.cpp, [130](#)

- instantRun
 - UAVparams, 91
- invMassMatrix
 - Aircraft, 26
- isNormal
 - control.cpp, 142
- lx
 - UAVparams, 91
- lxy
 - UAVparams, 91
- lxz
 - UAVparams, 92
- ly
 - UAVparams, 92
- lyz
 - UAVparams, 92
- lz
 - UAVparams, 92
- Jet, 52
 - getLastThrust, 53
 - getThrust, 53
 - phases, 54
 - start, 54
 - thrust, 54
 - time, 54
- jet_lift_loads
 - Forces, 49
- jets
 - Aircraft, 26
 - UAVparams, 92
- legend
 - controller_plots.m, 116
- lib/UAV_common/header/common.hpp, 115
- lib/UAV_common/scripts/controller_plots.m, 116
- lib/UAV_common/src/components/aero_coefficients.hpp, 119
- lib/UAV_common/src/components/components.hpp, 119
- lib/UAV_common/src/components/control_surfaces.cpp, 119
- lib/UAV_common/src/components/control_surfaces.hpp, 119
- lib/UAV_common/src/components/drive.cpp, 120
- lib/UAV_common/src/components/drive.hpp, 120
- lib/UAV_common/src/components/hinge.cpp, 120
- lib/UAV_common/src/components/hinge.hpp, 120
- lib/UAV_common/src/components/loads.cpp, 121
- lib/UAV_common/src/components/loads.hpp, 121
- lib/UAV_common/src/components/navi.hpp, 121
- lib/UAV_common/src/controllers/controller.cpp, 121
- lib/UAV_common/src/controllers/controller.hpp, 122
- lib/UAV_common/src/controllers/controller_test.cpp, 122
- lib/UAV_common/src/controllers/impl/bang_bang.cpp, 124
- lib/UAV_common/src/controllers/impl/bang_bang.hpp, 124
- lib/UAV_common/src/controllers/impl/double_setpoint.cpp, 124
- lib/UAV_common/src/controllers/impl/double_setpoint.hpp, 124
- lib/UAV_common/src/controllers/impl/PID.cpp, 125
- lib/UAV_common/src/controllers/impl/PID.hpp, 125
- lib/UAV_common/src/controllers/impl/PID_discrete.cpp, 125
- lib/UAV_common/src/controllers/impl/PID_discrete.hpp, 126
- lib/UAV_common/src/controllers/impl/z_trans.cpp, 126
- lib/UAV_common/src/controllers/impl/z_trans.hpp, 127
- lib/UAV_common/src/logger/logger.cpp, 127
- lib/UAV_common/src/logger/logger.hpp, 128
- lib/UAV_common/src/ode/ode.cpp, 128
- lib/UAV_common/src/ode/ode.hpp, 128
- lib/UAV_common/src/ode/ode_impl.hpp, 129
- lib/UAV_common/src/ode/ode_test.cpp, 129
- lib/UAV_common/src/parser/parser.cpp, 131
- lib/UAV_common/src/parser/parser.hpp, 132
- lib/UAV_common/src/parser/uav_params.cpp, 133
- lib/UAV_common/src/parser/uav_params.hpp, 134
- lib/UAV_common/src/timed_loop/status.hpp, 134
- lib/UAV_common/src/timed_loop/timed_loop.cpp, 136
- lib/UAV_common/src/timed_loop/timed_loop.hpp, 136
- Load, 55
 - getAmmount, 56
 - getMass, 56
 - getOffset, 56
 - Load, 55
 - operator=, 56
 - release, 56
- loadConfig
 - UAVparams, 90
- log
 - Logger, 58
- Logger, 57
 - ~Logger, 58
 - log, 58
 - Logger, 57
 - setFmt, 59
 - setLogDirectory, 59
- logger.cpp
 - shouldLog, 127
- logger.hpp
 - LOGGER_MASK, 128
- LOGGER_MASK
 - logger.hpp, 128
- m
 - UAVparams, 92
- main
 - CMakeCCompilerId.c, 109
 - CMakeCXXCompilerId.cpp, 112
 - controller_test.cpp, 123
 - main.cpp, 139
 - ode_test.cpp, 130
 - test2.cpp, 147
 - test3.cpp, 147

- test4.cpp, 148
- test5.cpp, 148
- main.cpp
 - main, 139
 - parseArgs, 140
- massMatrix
 - Aircraft, 26
 - Matrices, 61
- Matrices, 59
 - asSkewSymmetricMatrix, 60
 - gyroMatrix, 60
 - massMatrix, 61
 - OM_conj, 61
 - quaterionsToRPY, 61
 - R_nb, 62
 - R_wind_b, 62
 - RPYtoQuaterion, 63
 - TMatrix, 63
- maxSpeed
 - Rotor, 84
- MIXING_FUNCTION
 - def, 12
- mtx
 - Aircraft, 26
- name
 - SensorParams, 85
 - UAVparams, 92
- NONE
 - controllers::PID, 78
 - ODE, 65
- noOfAmmo
 - Aircraft, 27
 - UAVparams, 92
- noOfCargo
 - Aircraft, 27
 - UAVparams, 93
- noOfHinges
 - Drive, 45
- noOfJets
 - Aircraft, 27
 - UAVparams, 93
- noOfRotors
 - Aircraft, 27
 - UAVparams, 93
- ODE, 64
 - ~ODE, 65
 - Euler, 65
 - factory, 66
 - fromString, 66
 - getMicrosteps, 66, 67
 - Heun, 65
 - NONE, 65
 - ODE, 65
 - ODEMethod, 65
 - PC2, 65
 - PC4, 65
 - RK4, 65
 - step, 67
- ode
 - Aircraft, 27
- ODE_Euler, 68
 - ODE_Euler, 68
 - step, 68
- ODE_Heun, 69
 - ODE_Heun, 69
 - step, 70
- ODE_METHOD
 - Params, 77
- ODE_PC2, 70
 - ODE_PC2, 71
 - step, 71
- ODE_PC4, 72
 - ODE_PC4, 72
 - step, 72
- ODE_RK4, 73
 - ODE_RK4, 73
 - step, 74
- ode_test.cpp
 - getMethodsToTest, 129
 - INstantiate_TEST_SUITE_P, 130
 - main, 130
 - TEST_F, 130
 - TEST_P, 130, 131
- ODEMethod
 - ODE, 65
- ODETest, 74
 - SetUp, 75
 - TearDown, 75
- off
 - controller_plots.m, 118
- OM_conj
 - Matrices, 61
- on
 - controller_plots.m, 118
- operator<<
 - uav_state.cpp, 145
 - UAVstate, 101
- operator=
 - Ammo, 29
 - Hinge, 52
 - Load, 56
 - Params, 77
 - UAVstate, 98
- Params, 75
 - ~Params, 76
 - getSingleton, 76
 - ODE_METHOD, 77
 - operator=, 77
 - Params, 76
 - STEP_TIME, 77
- parseArgs
 - main.cpp, 140
- parseHinge
 - uav_params.cpp, 134
- parseMatrixXd

- parser.cpp, 131
 - parser.hpp, 132
- parser.cpp
 - parseMatrixXd, 131
 - parseVectorXd, 132
- parser.hpp
 - parseMatrixXd, 132
 - parseVectorXd, 133
- parseVectorXd
 - parser.cpp, 132
 - parser.hpp, 133
- PC2
 - ODE, 65
- PC4
 - ODE, 65
- phases
 - Jet, 54
- PID
 - controllers::PID, 78, 79
- PID_Discrete
 - controllers::PID_Discrete, 81
- PLATFORM_ID
 - CMakeCCompilerId.c, 108
 - CMakeCXXCompilerId.cpp, 112
- plot
 - controller_plots.m, 116
 - controller_test.cpp, 123
- plot_directory_name
 - controller_test.cpp, 123
- position
 - Drive, 45
- predictScaler
 - EKFScalers, 46
- Q
 - AHRSPParams, 19
- quaternionsToRPY
 - Matrices, 61
- R
 - AHRSPParams, 19
- R_nb
 - Matrices, 62
- R_wind_b
 - Matrices, 62
- real_time
 - UAVstate, 101
- reduceMass
 - Aircraft, 23
- refreshTime
 - SensorParams, 85
- release
 - Load, 56
- reload
 - status.hpp, 136
- restoreTrim
 - ControlSurfaces, 42
- RHS
 - Aircraft, 23
- RK4
 - ODE, 65
- Rotor, 83
 - direction, 83
 - forceCoff, 83
 - hoverSpeed, 84
 - maxSpeed, 84
 - timeConstant, 84
 - torqueCoff, 84
- rotor_lift_loads
 - Forces, 50
- rotorMixer
 - UAVparams, 93
- rotors
 - Aircraft, 27
 - UAVparams, 93
- RPYtoQuaterion
 - Matrices, 63
- run
 - Simulation, 86
- running
 - status.hpp, 136
- S
 - AeroCoefficients, 18
- s
 - zmq_recv, 13
 - zmq_recv_last, 14
- sd
 - SensorParams, 85
- sendState
 - Aircraft, 23
- SensorParams, 84
 - bias, 85
 - name, 85
 - refreshTime, 85
 - sd, 85
- sensors
 - UAVparams, 93
- set_dt
 - Controller, 39
 - controllers::PID_Discrete, 82
- setAcceleration
 - UAVstate, 98
- setAtmosphere
 - control.cpp, 142
- setControlSurface
 - control.cpp, 142
- setDemandedOm
 - UAVstate, 99
- setFmt
 - Logger, 59
- setForce
 - control.cpp, 143
 - UAVstate, 99
- setHinge
 - Aircraft, 24
- setHinges
 - control.cpp, 143

- setLogDirectory
 - Logger, [59](#)
- setOm
 - UAVstate, [99](#)
- setSpeed
 - control.cpp, [143](#)
- setStatus
 - UAVstate, [100](#)
- setSurface
 - Aircraft, [24](#)
- SetUp
 - ControllerTest, [40](#)
 - ODETest, [75](#)
- setValues
 - ControlSurfaces, [42](#)
- setX
 - UAVstate, [100](#)
- setY
 - UAVstate, [100](#)
- shoot
 - control.cpp, [143](#)
- shootAmmo
 - Aircraft, [24](#)
- shouldLog
 - logger.cpp, [127](#)
- Simulation, [85](#)
 - ~Simulation, [86](#)
 - run, [86](#)
 - Simulation, [86](#)
- socket
 - zmq_recv, [13](#)
 - zmq_recv_last, [14](#)
 - zmq_send, [15](#)
 - zmq_send_tcp, [15](#)
- solidSurfColision
 - control.cpp, [143](#)
- splitStringToDoubleVector
 - z_trans.cpp, [126](#)
- src/aircraft/aircraft.cpp, [136](#)
- src/aircraft/aircraft.hpp, [137](#)
- src/aircraft/aircraft_comm.cpp, [137](#)
- src/aircraft/aircraft_impulse.cpp, [137](#)
- src/defines.hpp, [137](#)
- src/dynamic/forces.cpp, [138](#)
- src/dynamic/forces.hpp, [138](#)
- src/dynamic/matrices.cpp, [139](#)
- src/dynamic/matrices.hpp, [139](#)
- src/main.cpp, [139](#)
- src/params.cpp, [140](#)
- src/params.hpp, [140](#)
- src/simulation/atmosphere.cpp, [140](#)
- src/simulation/atmosphere.hpp, [141](#)
- src/simulation/control.cpp, [141](#)
- src/simulation/control.hpp, [144](#)
- src/simulation/simulation.cpp, [144](#)
- src/simulation/simulation.hpp, [145](#)
- src/simulation/uav_state.cpp, [145](#)
- src/simulation/uav_state.hpp, [146](#)
- stallLimit
 - AeroCoefficients, [18](#)
- start
 - Jet, [54](#)
- startJet
 - Aircraft, [25](#)
 - control.cpp, [143](#)
- state
 - Aircraft, [27](#)
- state_mtx
 - UAVstate, [101](#)
- Status
 - status.hpp, [135](#)
- status
 - UAVstate, [101](#)
- status.hpp
 - exiting, [136](#)
 - idle, [136](#)
 - reload, [136](#)
 - running, [136](#)
 - Status, [135](#)
- status_cv
 - UAVstate, [102](#)
- step
 - ODE, [67](#)
 - ODE_Euler, [68](#)
 - ODE_Heun, [70](#)
 - ODE_PC2, [71](#)
 - ODE_PC4, [72](#)
 - ODE_RK4, [74](#)
- STEP_TIME
 - Params, [77](#)
- STRINGIFY
 - CMakeCCompilerId.c, [109](#)
 - CMakeCXXCompilerId.cpp, [112](#)
- STRINGIFY_HELPER
 - CMakeCCompilerId.c, [109](#)
 - CMakeCXXCompilerId.cpp, [112](#)
- surfaceMixer
 - UAVparams, [93](#)
- surfaces
 - Aircraft, [27](#)
 - UAVparams, [93](#)
- target
 - UAVparams, [94](#)
- TearDown
 - ControllerTest, [40](#)
 - ODETest, [75](#)
- test1
 - test1.cpp, [146](#)
- test1.cpp
 - fun, [146](#)
 - test1, [146](#)
- test2.cpp
 - main, [147](#)
- test3.cpp
 - fun, [147](#)
 - main, [147](#)

- test4.cpp
 - main, [148](#)
- test5.cpp
 - main, [148](#)
- TEST_F
 - ode_test.cpp, [130](#)
- TEST_P
 - controller_test.cpp, [123](#)
 - ode_test.cpp, [130](#), [131](#)
- tests/test1.cpp, [146](#)
- tests/test2.cpp, [147](#)
- tests/test3.cpp, [147](#)
- tests/test4.cpp, [148](#)
- tests/test5.cpp, [148](#)
- tests/zmq_recv.py, [149](#)
- tests/zmq_recv_last.py, [149](#)
- tests/zmq_send.py, [149](#)
- tests/zmq_send_tcp.py, [149](#)
- thrust
 - Jet, [54](#)
- time
 - Jet, [54](#)
- timeConstant
 - Rotor, [84](#)
- TimedLoop, [86](#)
 - go, [87](#)
 - TimedLoop, [87](#)
- title
 - controller_plots.m, [117](#)
- TMatrix
 - Matrices, [63](#)
- topicfilter
 - zmq_recv, [13](#)
 - zmq_recv_last, [14](#)
- torqueCoff
 - Rotor, [84](#)
- trim
 - Aircraft, [25](#)
- type
 - AHRSParams, [19](#)
- uav_params.cpp
 - parseHinge, [134](#)
- uav_state.cpp
 - operator<<, [145](#)
- UAVparams, [88](#)
 - ~UAVparams, [89](#)
 - aero_coffs, [90](#)
 - ahrs, [90](#)
 - ammo, [90](#)
 - cargo, [90](#)
 - controllers, [90](#)
 - ekf, [91](#)
 - getRotorHoverSpeeds, [89](#)
 - getRotorMaxSpeeds, [89](#)
 - getRotorTimeContants, [89](#)
 - getSingleton, [90](#)
 - initialMode, [91](#)
 - initialOrientation, [91](#)
 - initialPosition, [91](#)
 - initialVelocity, [91](#)
 - instantRun, [91](#)
 - lx, [91](#)
 - lxy, [91](#)
 - lxz, [92](#)
 - ly, [92](#)
 - lyz, [92](#)
 - lz, [92](#)
 - jets, [92](#)
 - loadConfig, [90](#)
 - m, [92](#)
 - name, [92](#)
 - noOfAmmo, [92](#)
 - noOfCargo, [93](#)
 - noOfJets, [93](#)
 - noOfRotors, [93](#)
 - rotorMixer, [93](#)
 - rotors, [93](#)
 - sensors, [93](#)
 - surfaceMixer, [93](#)
 - surfaces, [93](#)
 - target, [94](#)
 - UAVparams, [89](#)
- UAVstate, [94](#)
 - ~UAVstate, [95](#)
 - getAcceleration, [96](#)
 - getDemandedOm, [96](#)
 - getNoOfRotors, [96](#)
 - getOm, [96](#)
 - getOuterForce, [97](#)
 - getState, [97](#)
 - getX, [97](#)
 - getY, [98](#)
 - operator<<, [101](#)
 - operator=, [98](#)
 - real_time, [101](#)
 - setAcceleration, [98](#)
 - setDemandedOm, [99](#)
 - setForce, [99](#)
 - setOm, [99](#)
 - setStatus, [100](#)
 - setX, [100](#)
 - setY, [100](#)
 - state_mtx, [101](#)
 - status, [101](#)
 - status_cv, [102](#)
 - UAVstate, [95](#)
- update
 - Aircraft, [25](#)
 - Atmosphere, [32](#)
- updateScaler
 - EKFScalers, [46](#)
- updateValue
 - Hinge, [52](#)
- USE_QUATERIONS
 - defines.hpp, [138](#)
- validityOfForce

- def, [12](#)
- wind
 - AtmosphereInfo, [33](#)
- x
 - controller_plots.m, [118](#)
- xlabel
 - controller_plots.m, [117](#)
- y
 - controller_plots.m, [118](#)
- ylabel
 - controller_plots.m, [117](#)
- z_trans.cpp
 - splitStringToDoubleVector, [126](#)
- zmq_rcv, [13](#)
 - context, [13](#)
 - s, [13](#)
 - socket, [13](#)
 - topicfilter, [13](#)
- zmq_rcv_last, [13](#)
 - context, [14](#)
 - s, [14](#)
 - socket, [14](#)
 - topicfilter, [14](#)
- zmq_send, [14](#)
 - context, [14](#)
 - counter, [14](#)
 - socket, [15](#)
- zmq_send_tcp, [15](#)
 - angle, [15](#)
 - context, [15](#)
 - socket, [15](#)
- zScaler
 - EKFScalers, [46](#)
- ZTransform
 - controllers::ZTransform, [102](#), [103](#)
- ZTransformStatic
 - controllers::ZTransformStatic< N, D >, [104](#), [105](#)