

UAV drop physic

Generated by Doxygen 1.9.1



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 controllers Namespace Reference	9
5.2 def Namespace Reference	9
5.2.1 Detailed Description	9
5.2.2 Variable Documentation	9
5.2.2.1 DEFAULT_AIR_DENSITY	10
5.2.2.2 FRICTION_EPS	10
5.2.2.3 GENTLY_PUSH	10
5.2.2.4 GRAVITY_CONST	10
<b>6 Class Documentation</b>	<b>11</b>
6.1 AeroCoefficients Struct Reference	11
6.1.1 Detailed Description	11
6.1.2 Member Data Documentation	11
6.1.2.1 C0	11
6.1.2.2 Cab	12
6.1.2.3 Cpqr	12
6.1.2.4 d	12
6.1.2.5 eAR	12
6.1.2.6 S	12
6.1.2.7 stallLimit	12
6.2 AHRSPParams Struct Reference	12
6.2.1 Detailed Description	13
6.2.2 Member Data Documentation	13
6.2.2.1 alpha	13
6.2.2.2 Q	13
6.2.2.3 R	13
6.2.2.4 type	13
6.3 Ammo Class Reference	14
6.3.1 Constructor & Destructor Documentation	14
6.3.1.1 Ammo() [1/2]	14
6.3.1.2 Ammo() [2/2]	14

6.3.2 Member Function Documentation	14
6.3.2.1 getV0()	15
6.3.2.2 operator=()	15
6.3.3 Member Data Documentation	15
6.3.3.1 _V0	15
6.4 controllers::BangBang Class Reference	15
6.4.1 Constructor & Destructor Documentation	16
6.4.1.1 BangBang() [1/2]	16
6.4.1.2 BangBang() [2/2]	16
6.4.2 Member Function Documentation	16
6.4.2.1 calc()	16
6.4.2.2 clear()	17
6.4.2.3 clone()	17
6.5 Cargo Class Reference	17
6.5.1 Constructor & Destructor Documentation	18
6.5.1.1 Cargo() [1/2]	18
6.5.1.2 Cargo() [2/2]	18
6.6 Controller Class Reference	18
6.6.1 Constructor & Destructor Documentation	19
6.6.1.1 Controller()	19
6.6.1.2 ~Controller()	19
6.6.2 Member Function Documentation	19
6.6.2.1 calc() [1/2]	19
6.6.2.2 calc() [2/2]	20
6.6.2.3 clear()	20
6.6.2.4 clone()	20
6.6.2.5 ControllerFactory()	20
6.6.2.6 set_dt()	21
6.6.3 Member Data Documentation	21
6.6.3.1 _dt	21
6.7 ControllerTest Class Reference	21
6.7.1 Member Function Documentation	21
6.7.1.1 SetUp()	22
6.7.1.2 TearDown()	22
6.8 ControlSurfaces Class Reference	22
6.8.1 Detailed Description	22
6.8.2 Constructor & Destructor Documentation	22
6.8.2.1 ControlSurfaces() [1/2]	22
6.8.2.2 ControlSurfaces() [2/2]	22
6.8.3 Member Function Documentation	23
6.8.3.1 getCoefficients()	23
6.8.3.2 getNoOfSurface()	23

6.8.3.3 <code>getValues()</code>	23
6.8.3.4 <code>restoreTrim()</code>	23
6.8.3.5 <code>setValues()</code>	23
6.9 <code>controllers::DoubleSetpoint</code> Class Reference	24
6.9.1 Constructor & Destructor Documentation	24
6.9.1.1 <code>DoubleSetpoint()</code> [1/2]	24
6.9.1.2 <code>DoubleSetpoint()</code> [2/2]	25
6.9.2 Member Function Documentation	25
6.9.2.1 <code>calc()</code>	25
6.9.2.2 <code>clear()</code>	25
6.9.2.3 <code>clone()</code>	26
6.10 Drive Struct Reference	26
6.10.1 Detailed Description	26
6.10.2 Member Data Documentation	26
6.10.2.1 <code>axis</code>	26
6.10.2.2 <code>hinges</code>	27
6.10.2.3 <code>noOfHinges</code>	27
6.10.2.4 <code>position</code>	27
6.11 <code>DropTest</code> Class Reference	27
6.11.1 Member Function Documentation	27
6.11.1.1 <code>collectSample()</code>	27
6.11.1.2 <code>getParsedState()</code>	28
6.11.1.3 <code>recvState()</code>	28
6.11.1.4 <code>sendControlMessage()</code>	28
6.11.1.5 <code>SetUp()</code>	28
6.11.1.6 <code>TearDown()</code>	28
6.12 <code>EKFScalers</code> Struct Reference	28
6.12.1 Detailed Description	29
6.12.2 Member Data Documentation	29
6.12.2.1 <code>baroScaler</code>	29
6.12.2.2 <code>predictScaler</code>	29
6.12.2.3 <code>updateScaler</code>	29
6.12.2.4 <code>zScaler</code>	29
6.13 Hinge Class Reference	29
6.13.1 Detailed Description	30
6.13.2 Constructor & Destructor Documentation	30
6.13.2.1 <code>Hinge()</code> [1/3]	30
6.13.2.2 <code>Hinge()</code> [2/3]	30
6.13.2.3 <code>Hinge()</code> [3/3]	30
6.13.3 Member Function Documentation	30
6.13.3.1 <code>getRot()</code>	31
6.13.3.2 <code>operator=()</code>	31

6.13.3.3 updateValue()	31
6.14 Jet Class Reference	31
6.14.1 Detailed Description	32
6.14.2 Member Function Documentation	32
6.14.2.1 getLastThrust()	32
6.14.2.2 getThrust()	32
6.14.2.3 start()	33
6.14.3 Member Data Documentation	33
6.14.3.1 phases	33
6.14.3.2 thrust	33
6.14.3.3 time	33
6.15 Load Class Reference	34
6.15.1 Detailed Description	34
6.15.2 Constructor & Destructor Documentation	34
6.15.2.1 Load() [1/2]	34
6.15.2.2 Load() [2/2]	34
6.15.3 Member Function Documentation	35
6.15.3.1 getMass()	35
6.15.3.2 getOffset()	35
6.15.3.3 operator=()	35
6.15.3.4 release()	35
6.16 Logger Class Reference	36
6.16.1 Detailed Description	36
6.16.2 Constructor & Destructor Documentation	36
6.16.2.1 Logger()	36
6.16.2.2 ~Logger()	37
6.16.3 Member Function Documentation	37
6.16.3.1 log() [1/2]	37
6.16.3.2 log() [2/2]	37
6.16.3.3 setFmt()	38
6.16.3.4 setLogDirectory()	38
6.17 ObjParams Class Reference	38
6.17.1 Detailed Description	39
6.17.2 Constructor & Destructor Documentation	39
6.17.2.1 ObjParams() [1/2]	39
6.17.2.2 ObjParams() [2/2]	39
6.17.3 Member Function Documentation	39
6.17.3.1 getForce()	40
6.17.3.2 getWind()	40
6.17.3.3 setForce()	40
6.17.3.4 setWind()	40
6.17.4 Member Data Documentation	41

6.17.4.1 CS_coff . . . . .	41
6.17.4.2 id . . . . .	41
6.17.4.3 mass . . . . .	41
6.18 ODE Class Reference . . . . .	41
6.18.1 Detailed Description . . . . .	42
6.18.2 Member Enumeration Documentation . . . . .	42
6.18.2.1 ODEMethod . . . . .	42
6.18.3 Constructor & Destructor Documentation . . . . .	43
6.18.3.1 ODE() . . . . .	43
6.18.3.2 ~ODE() . . . . .	43
6.18.4 Member Function Documentation . . . . .	43
6.18.4.1 factory() . . . . .	43
6.18.4.2 fromString() . . . . .	43
6.18.4.3 getMicrosteps() [1/2] . . . . .	44
6.18.4.4 getMicrosteps() [2/2] . . . . .	44
6.18.4.5 step() . . . . .	44
6.19 ODE_Euler Class Reference . . . . .	45
6.19.1 Detailed Description . . . . .	45
6.19.2 Constructor & Destructor Documentation . . . . .	45
6.19.2.1 ODE_Euler() . . . . .	46
6.19.3 Member Function Documentation . . . . .	46
6.19.3.1 step() . . . . .	46
6.20 ODE_Heun Class Reference . . . . .	46
6.20.1 Detailed Description . . . . .	47
6.20.2 Constructor & Destructor Documentation . . . . .	47
6.20.2.1 ODE_Heun() . . . . .	47
6.20.3 Member Function Documentation . . . . .	47
6.20.3.1 step() . . . . .	47
6.21 ODE_PC2 Class Reference . . . . .	48
6.21.1 Detailed Description . . . . .	48
6.21.2 Constructor & Destructor Documentation . . . . .	48
6.21.2.1 ODE_PC2() . . . . .	48
6.21.3 Member Function Documentation . . . . .	48
6.21.3.1 step() . . . . .	48
6.22 ODE_PC4 Class Reference . . . . .	49
6.22.1 Detailed Description . . . . .	49
6.22.2 Constructor & Destructor Documentation . . . . .	49
6.22.2.1 ODE_PC4() . . . . .	50
6.22.3 Member Function Documentation . . . . .	50
6.22.3.1 step() . . . . .	50
6.23 ODE_RK4 Class Reference . . . . .	50
6.23.1 Detailed Description . . . . .	51

6.23.2 Constructor & Destructor Documentation	51
6.23.2.1 ODE_RK4()	51
6.23.3 Member Function Documentation	51
6.23.3.1 step()	51
6.24 ODETest Class Reference	52
6.24.1 Member Function Documentation	52
6.24.1.1 SetUp()	52
6.24.1.2 TearDown()	52
6.25 Params Class Reference	52
6.25.1 Detailed Description	53
6.25.2 Constructor & Destructor Documentation	53
6.25.2.1 Params() [1/3]	53
6.25.2.2 Params() [2/3]	53
6.25.2.3 Params() [3/3]	54
6.25.2.4 ~Params()	54
6.25.3 Member Function Documentation	54
6.25.3.1 getSingleton()	54
6.25.3.2 operator=()	54
6.25.4 Member Data Documentation	54
6.25.4.1 ODE_METHOD	54
6.25.4.2 STEP_TIME	55
6.26 controllers::PID Class Reference	55
6.26.1 Member Enumeration Documentation	55
6.26.1.1 AntiWindUpMode	55
6.26.2 Constructor & Destructor Documentation	56
6.26.2.1 PID() [1/2]	56
6.26.2.2 PID() [2/2]	56
6.26.3 Member Function Documentation	56
6.26.3.1 calc()	57
6.26.3.2 clear()	57
6.26.3.3 clone()	57
6.27 controllers::PIFF Class Reference	57
6.27.1 Member Enumeration Documentation	58
6.27.1.1 AntiWindUpMode	58
6.27.2 Constructor & Destructor Documentation	58
6.27.2.1 PIFF() [1/2]	58
6.27.2.2 PIFF() [2/2]	59
6.27.3 Member Function Documentation	59
6.27.3.1 calc()	59
6.27.3.2 clear()	60
6.27.3.3 clone()	60
6.28 Projectile Struct Reference	60



6.28.1 Member Data Documentation	60
6.28.1.1 id	61
6.28.1.2 position	61
6.28.1.3 velocity	61
6.29 Rotor Struct Reference	61
6.29.1 Detailed Description	61
6.29.2 Member Data Documentation	61
6.29.2.1 direction	62
6.29.2.2 forceCoff	62
6.29.2.3 hoverSpeed	62
6.29.2.4 maxSpeed	62
6.29.2.5 timeConstant	62
6.29.2.6 torqueCoff	62
6.30 SensorParams Struct Reference	62
6.30.1 Detailed Description	63
6.30.2 Member Data Documentation	63
6.30.2.1 bias	63
6.30.2.2 name	63
6.30.2.3 refreshTime	63
6.30.2.4 sd	63
6.31 Simulation Class Reference	63
6.31.1 Constructor & Destructor Documentation	64
6.31.1.1 Simulation()	64
6.31.1.2 ~Simulation()	64
6.31.2 Member Function Documentation	65
6.31.2.1 addCommand()	65
6.31.2.2 addObj()	65
6.31.2.3 calcImpulseForce()	65
6.31.2.4 removeObj()	67
6.31.2.5 run()	67
6.31.2.6 solidSurfColision()	67
6.31.2.7 updateForce()	68
6.31.2.8 updateWind()	68
6.32 State Class Reference	68
6.32.1 Constructor & Destructor Documentation	69
6.32.1.1 State()	69
6.32.2 Member Function Documentation	69
6.32.2.1 addObj()	69
6.32.2.2 findIndex()	70
6.32.2.3 getNoObj()	70
6.32.2.4 getParams()	70
6.32.2.5 getPos()	71

6.32.2.6	getState()	71
6.32.2.7	getVel()	71
6.32.2.8	removeObj()	72
6.32.2.9	setVel()	72
6.32.2.10	to_string()	72
6.32.2.11	updateForce()	73
6.32.2.12	updateState()	73
6.32.2.13	updateWind()	73
6.32.3	Member Data Documentation	73
6.32.3.1	real_time	74
6.32.3.2	stateMutex	74
6.32.3.3	status	74
6.33	TimedLoop Class Reference	74
6.33.1	Detailed Description	74
6.33.2	Constructor & Destructor Documentation	74
6.33.2.1	TimedLoop()	74
6.33.3	Member Function Documentation	75
6.33.3.1	go() [1/2]	75
6.33.3.2	go() [2/2]	75
6.34	UAVparams Struct Reference	75
6.34.1	Detailed Description	76
6.34.2	Constructor & Destructor Documentation	77
6.34.2.1	UAVparams()	77
6.34.2.2	~UAVparams()	77
6.34.3	Member Function Documentation	77
6.34.3.1	getRotorHoverSpeeds()	77
6.34.3.2	getRotorMaxSpeeds()	77
6.34.3.3	getRotorTimeContants()	77
6.34.3.4	getSingleton()	77
6.34.3.5	loadConfig()	78
6.34.4	Member Data Documentation	78
6.34.4.1	aero_coffs	78
6.34.4.2	ahrs	78
6.34.4.3	ammo	78
6.34.4.4	cargo	78
6.34.4.5	controllers	78
6.34.4.6	ekf	78
6.34.4.7	initialMode	79
6.34.4.8	initialOrientation	79
6.34.4.9	initialPosition	79
6.34.4.10	initialVelocity	79
6.34.4.11	instantRun	79

6.34.4.12 lx	79
6.34.4.13 lxy	79
6.34.4.14 lxz	79
6.34.4.15 ly	80
6.34.4.16 lyz	80
6.34.4.17 lz	80
6.34.4.18 jets	80
6.34.4.19 m	80
6.34.4.20 name	80
6.34.4.21 noOfAmmo	80
6.34.4.22 noOfCargo	80
6.34.4.23 noOfJets	81
6.34.4.24 noOfRotors	81
6.34.4.25 rotorMixer	81
6.34.4.26 rotors	81
6.34.4.27 sensors	81
6.34.4.28 surfaceMixer	81
6.34.4.29 surfaces	81
6.34.4.30 target	81
<b>7 File Documentation</b>	<b>83</b>
7.1 build/CMakeFiles/3.22.1/CompilerIdC/CMakeCCompilerId.c File Reference	83
7.1.1 Macro Definition Documentation	83
7.1.1.1 __has_include	84
7.1.1.2 ARCHITECTURE_ID	84
7.1.1.3 C_VERSION	84
7.1.1.4 COMPILER_ID	84
7.1.1.5 DEC	84
7.1.1.6 HEX	84
7.1.1.7 PLATFORM_ID	85
7.1.1.8 STRINGIFY	85
7.1.1.9 STRINGIFY_HELPER	85
7.1.2 Function Documentation	85
7.1.2.1 main()	85
7.1.3 Variable Documentation	85
7.1.3.1 info_arch	85
7.1.3.2 info_compiler	85
7.1.3.3 info_language_extensions_default	86
7.1.3.4 info_language_standard_default	86
7.1.3.5 info_platform	86
7.2 build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference	86
7.2.1 Macro Definition Documentation	87

7.2.1.1 <code>__has_include</code>	87
7.2.1.2 <code>ARCHITECTURE_ID</code>	87
7.2.1.3 <code>COMPILER_ID</code>	87
7.2.1.4 <code>CXX_STD</code>	87
7.2.1.5 <code>DEC</code>	87
7.2.1.6 <code>HEX</code>	88
7.2.1.7 <code>PLATFORM_ID</code>	88
7.2.1.8 <code>STRINGIFY</code>	88
7.2.1.9 <code>STRINGIFY_HELPER</code>	88
7.2.2 Function Documentation	88
7.2.2.1 <code>main()</code>	88
7.2.3 Variable Documentation	88
7.2.3.1 <code>info_arch</code>	89
7.2.3.2 <code>info_compiler</code>	89
7.2.3.3 <code>info_language_extensions_default</code>	89
7.2.3.4 <code>info_language_standard_default</code>	89
7.2.3.5 <code>info_platform</code>	89
7.3 <code>build/CMakeFiles/drop.dir/src/main.cpp.o.d</code> File Reference	91
7.4 <code>build/CMakeFiles/drop.dir/src/params.cpp.o.d</code> File Reference	91
7.5 <code>build/CMakeFiles/drop.dir/src/simulation.cpp.o.d</code> File Reference	91
7.6 <code>build/CMakeFiles/drop.dir/src/state.cpp.o.d</code> File Reference	91
7.7 <code>build/CMakeFiles/integration_test.dir/tests/integration_test.cpp.o.d</code> File Reference	91
7.8 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/components/control_surfaces.cpp.o.d</code> File Reference	91
7.9 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/components/drive.cpp.o.d</code> File Reference	91
7.10 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/components/hinge.cpp.o.d</code> File Reference	91
7.11 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/components/loads.cpp.o.d</code> File Reference	91
7.12 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/controller.cpp.o.d</code> File Reference	91
7.13 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/bang_bang.cpp.o.d</code> File Reference	91
7.14 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/double_setpoint.cpp.o.d</code> File Reference	91
7.15 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/PID.cpp.o.d</code> File Reference	91
7.16 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/PID/PID.cpp.o.d</code> File Reference	91
7.17 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/PIFF.cpp.o.d</code> File Reference	91
7.18 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/logger/logger.cpp.o.d</code> File Reference	91
7.19 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/ode/ode.cpp.o.d</code> File Reference	91
7.20 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/parser/parser.cpp.o.d</code> File Reference	91
7.21 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/parser/uav_params.cpp.o.d</code> File Reference	91
7.22 <code>build/lib/UAV_common/CMakeFiles/common.dir/src/timed_loop/timed_loop.cpp.o.d</code> File Reference	91
7.23 <code>build/lib/UAV_common/CMakeFiles/Controller_test.dir/src/controllers/controller_test.cpp.o.d</code> File Reference	91
7.24 <code>build/lib/UAV_common/CMakeFiles/ODE_test.dir/src/ode/ode_test.cpp.o.d</code> File Reference	91
7.25 <code>lib/UAV_common/header/common.hpp</code> File Reference	91

7.26 lib/UAV_common/scripts/controller_plots.m File Reference	92
7.26.1 Function Documentation	92
7.26.1.1 legend()	92
7.26.1.2 plot()	93
7.26.1.3 title()	93
7.26.1.4 xlabel()	93
7.26.1.5 ylabel()	93
7.26.2 Variable Documentation	93
7.26.2.1 clc	93
7.26.2.2 csvFiles	93
7.26.2.3 data	94
7.26.2.4 figure	94
7.26.2.5 folderPath	94
7.26.2.6 i	94
7.26.2.7 off	94
7.26.2.8 on	94
7.26.2.9 x	94
7.26.2.10 y	95
7.27 lib/UAV_common/src/components/aero_coefficients.hpp File Reference	95
7.28 lib/UAV_common/src/components/components.hpp File Reference	95
7.29 lib/UAV_common/src/components/control_surfaces.cpp File Reference	95
7.30 lib/UAV_common/src/components/control_surfaces.hpp File Reference	95
7.31 lib/UAV_common/src/components/drive.cpp File Reference	96
7.32 lib/UAV_common/src/components/drive.hpp File Reference	96
7.33 lib/UAV_common/src/components/hinge.cpp File Reference	96
7.33.1 Function Documentation	96
7.33.1.1 asSkewMatrix()	96
7.34 lib/UAV_common/src/components/hinge.hpp File Reference	96
7.35 lib/UAV_common/src/components/loads.cpp File Reference	97
7.36 lib/UAV_common/src/components/loads.hpp File Reference	97
7.37 lib/UAV_common/src/components/navi.hpp File Reference	97
7.38 lib/UAV_common/src/controllers/controller.cpp File Reference	97
7.39 lib/UAV_common/src/controllers/controller.hpp File Reference	98
7.40 lib/UAV_common/src/controllers/controller_test.cpp File Reference	98
7.40.1 Function Documentation	98
7.40.1.1 getMethodsToTest()	99
7.40.1.2 INSTANTIATE_TEST_SUITE_P()	99
7.40.1.3 main()	99
7.40.1.4 TEST_P() [1/2]	99
7.40.1.5 TEST_P() [2/2]	99
7.40.2 Variable Documentation	99
7.40.2.1 plot	99

7.40.2.2 plot_directory_name . . . . .	100
7.41 lib/UAV_common/src/controllers/impl/bang_bang.cpp File Reference . . . . .	100
7.42 lib/UAV_common/src/controllers/impl/bang_bang.hpp File Reference . . . . .	100
7.43 lib/UAV_common/src/controllers/impl/double_setpoint.cpp File Reference . . . . .	100
7.44 lib/UAV_common/src/controllers/impl/double_setpoint.hpp File Reference . . . . .	100
7.45 lib/UAV_common/src/controllers/impl/PID.cpp File Reference . . . . .	101
7.46 lib/UAV_common/src/controllers/impl/PID.hpp File Reference . . . . .	101
7.47 lib/UAV_common/src/controllers/impl/PIFF.cpp File Reference . . . . .	101
7.48 lib/UAV_common/src/controllers/impl/PIFF.hpp File Reference . . . . .	101
7.49 lib/UAV_common/src/logger/logger.cpp File Reference . . . . .	102
7.49.1 Function Documentation . . . . .	102
7.49.1.1 shouldLog() . . . . .	102
7.50 lib/UAV_common/src/logger/logger.hpp File Reference . . . . .	102
7.50.1 Macro Definition Documentation . . . . .	103
7.50.1.1 LOGGER_MASK . . . . .	103
7.51 lib/UAV_common/src/ode/ode.cpp File Reference . . . . .	103
7.52 lib/UAV_common/src/ode/ode.hpp File Reference . . . . .	103
7.53 lib/UAV_common/src/ode/ode_impl.hpp File Reference . . . . .	103
7.54 lib/UAV_common/src/ode/ode_test.cpp File Reference . . . . .	104
7.54.1 Function Documentation . . . . .	104
7.54.1.1 getMethodsToTest() . . . . .	104
7.54.1.2 INSTANTIATE_TEST_SUITE_P() . . . . .	104
7.54.1.3 main() . . . . .	104
7.54.1.4 TEST_F() [1/2] . . . . .	105
7.54.1.5 TEST_F() [2/2] . . . . .	105
7.54.1.6 TEST_P() [1/4] . . . . .	105
7.54.1.7 TEST_P() [2/4] . . . . .	105
7.54.1.8 TEST_P() [3/4] . . . . .	105
7.54.1.9 TEST_P() [4/4] . . . . .	105
7.55 lib/UAV_common/src/parser/parser.cpp File Reference . . . . .	106
7.55.1 Function Documentation . . . . .	106
7.55.1.1 parseMatrixXd() . . . . .	106
7.55.1.2 parseVectorXd() . . . . .	106
7.56 lib/UAV_common/src/parser/parser.hpp File Reference . . . . .	107
7.56.1 Function Documentation . . . . .	107
7.56.1.1 parseMatrixXd() . . . . .	107
7.56.1.2 parseVectorXd() . . . . .	108
7.57 lib/UAV_common/src/parser/uav_params.cpp File Reference . . . . .	108
7.57.1 Function Documentation . . . . .	108
7.57.1.1 parseHinge() . . . . .	108
7.58 lib/UAV_common/src/parser/uav_params.hpp File Reference . . . . .	109
7.59 lib/UAV_common/src/timed_loop/status.hpp File Reference . . . . .	109

7.59.1 Enumeration Type Documentation . . . . .	109
7.59.1.1 Status . . . . .	109
7.60 lib/UAV_common/src/timed_loop/timed_loop.cpp File Reference . . . . .	110
7.61 lib/UAV_common/src/timed_loop/timed_loop.hpp File Reference . . . . .	110
7.62 src/defines.hpp File Reference . . . . .	110
7.63 src/main.cpp File Reference . . . . .	111
7.63.1 Function Documentation . . . . .	111
7.63.1.1 main() . . . . .	111
7.63.1.2 parseArgs() . . . . .	111
7.64 src/params.cpp File Reference . . . . .	111
7.65 src/params.hpp File Reference . . . . .	112
7.66 src/simulation.cpp File Reference . . . . .	112
7.66.1 Function Documentation . . . . .	112
7.66.1.1 isNormal() . . . . .	112
7.67 src/simulation.hpp File Reference . . . . .	113
7.68 src/state.cpp File Reference . . . . .	113
7.69 src/state.hpp File Reference . . . . .	113
7.70 tests/integration_test.cpp File Reference . . . . .	114
7.70.1 Function Documentation . . . . .	115
7.70.1.1 main() . . . . .	115
7.70.1.2 TEST() . . . . .	115
7.70.1.3 TEST_F() [1/8] . . . . .	115
7.70.1.4 TEST_F() [2/8] . . . . .	115
7.70.1.5 TEST_F() [3/8] . . . . .	115
7.70.1.6 TEST_F() [4/8] . . . . .	116
7.70.1.7 TEST_F() [5/8] . . . . .	116
7.70.1.8 TEST_F() [6/8] . . . . .	116
7.70.1.9 TEST_F() [7/8] . . . . .	116
7.70.1.10 TEST_F() [8/8] . . . . .	116
7.70.2 Variable Documentation . . . . .	117
7.70.2.1 communicationFolder . . . . .	117
7.70.2.2 ctx . . . . .	117
7.70.2.3 programPath . . . . .	117
7.71 tests/projectile_parser.hpp File Reference . . . . .	117
7.71.1 Function Documentation . . . . .	117
7.71.1.1 checkInputErrors() . . . . .	117
7.71.1.2 parseInput() . . . . .	117





# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">controllers</a> . . . . .	9
<a href="#">def</a>	
<a href="#">Simulation</a> constants . . . . .	9



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

AeroCoefficients . . . . .	11
AHRSPParams . . . . .	12
Controller . . . . .	18
controllers::BangBang . . . . .	15
controllers::DoubleSetpoint . . . . .	24
controllers::PID . . . . .	55
controllers::PIFF . . . . .	57
ControlSurfaces . . . . .	22
Drive . . . . .	26
Jet . . . . .	31
Rotor . . . . .	61
EKFScalers . . . . .	28
Hinge . . . . .	29
Load . . . . .	34
Ammo . . . . .	14
Cargo . . . . .	17
Logger . . . . .	36
ObjParams . . . . .	38
ODE . . . . .	41
ODE_Euler . . . . .	45
ODE_Heun . . . . .	46
ODE_PC2 . . . . .	48
ODE_PC4 . . . . .	49
ODE_RK4 . . . . .	50
Params . . . . .	52
Projectile . . . . .	60
SensorParams . . . . .	62
Simulation . . . . .	63
State . . . . .	68
testing::Test . . . . .	
DropTest . . . . .	27
testing::TestWithParam . . . . .	
ControllerTest . . . . .	21
ODETest . . . . .	52
TimedLoop . . . . .	74
UAVparams . . . . .	75



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">AeroCoefficients</a>	
Aerodynamic coefficient	11
<a href="#">AHRSParams</a>	
AHRS parameters	12
<a href="#">Ammo</a>	14
<a href="#">controllers::BangBang</a>	15
<a href="#">Cargo</a>	17
<a href="#">Controller</a>	18
<a href="#">ControllerTest</a>	21
<a href="#">ControlSurfaces</a>	
Aircraft's control surfaces	22
<a href="#">controllers::DoubleSetpoint</a>	24
<a href="#">Drive</a>	
Drive propelling aircraft	26
<a href="#">DropTest</a>	27
<a href="#">EKFScalers</a>	
Scalers for EKF	28
<a href="#">Hinge</a>	
Hinge connecting aircraft with drives	29
<a href="#">Jet</a>	
Jet rocket engine	31
<a href="#">Load</a>	
Load of aircraft that can be dropped or launched	34
<a href="#">Logger</a>	
Log vector data with timestamp in file	36
<a href="#">ObjParams</a>	
Single obj parameters	38
<a href="#">ODE</a>	
Ordinal differential equation solver	41
<a href="#">ODE_Euler</a>	
Explicit Euler algorithm	45
<a href="#">ODE_Heun</a>	
Second order explicit Heun algorithm	46
<a href="#">ODE_PC2</a>	
Second order predictor-corrector method Second order Adams-bashforth and Adams-moulton	48

<a href="#">ODE_PC4</a>	
Fourth order predictor-corrector method Fourth order Adams-bashforth and Adams-moulton . .	49
<a href="#">ODE_RK4</a>	
Fourth order Runge Kutta algorithm . . . . .	50
<a href="#">ODETest</a>	52
<a href="#">Params</a>	
Simulation parameters . . . . .	52
<a href="#">controllers::PID</a>	55
<a href="#">controllers::PIFF</a>	57
<a href="#">Projectile</a>	60
<a href="#">Rotor</a>	
Rotor engine with controlled speed . . . . .	61
<a href="#">SensorParams</a>	
Base parameters of a sensor . . . . .	62
<a href="#">Simulation</a>	63
<a href="#">State</a>	68
<a href="#">TimedLoop</a>	
Simulation of real-time synchronized loop . . . . .	74
<a href="#">UAVparams</a>	
Parsed UAV configuration from XML . . . . .	75

## 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	83
build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp	86
build/CMakeFiles/drop.dir/src/main.cpp.o.d	91
build/CMakeFiles/drop.dir/src/params.cpp.o.d	91
build/CMakeFiles/drop.dir/src/simulation.cpp.o.d	91
build/CMakeFiles/drop.dir/src/state.cpp.o.d	91
build/CMakeFiles/integration_test.dir/tests/integration_test.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/components/control_surfaces.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/components/drive.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/components/hinge.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/components/loads.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/controller.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/bang_bang.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/double_setpoint.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/PID.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/PIFF.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/logger/logger.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/ode/ode.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/parser/parser.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/parser/uav_params.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/PID/PID.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/common.dir/src/timed_loop/timed_loop.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/Controller_test.dir/src/controllers/controller_test.cpp.o.d	91
build/lib/UAV_common/CMakeFiles/ODE_test.dir/src/ode/ode_test.cpp.o.d	91
lib/UAV_common/header/common.hpp	91
lib/UAV_common/scripts/controller_plots.m	92
lib/UAV_common/src/components/aero_coefficients.hpp	95
lib/UAV_common/src/components/components.hpp	95
lib/UAV_common/src/components/control_surfaces.cpp	95
lib/UAV_common/src/components/control_surfaces.hpp	95
lib/UAV_common/src/components/drive.cpp	96
lib/UAV_common/src/components/drive.hpp	96
lib/UAV_common/src/components/hinge.cpp	96
lib/UAV_common/src/components/hinge.hpp	96
lib/UAV_common/src/components/loads.cpp	97

lib/UAV_common/src/components/loads.hpp	97
lib/UAV_common/src/components/navi.hpp	97
lib/UAV_common/src/controllers/controller.cpp	97
lib/UAV_common/src/controllers/controller.hpp	98
lib/UAV_common/src/controllers/controller_test.cpp	98
lib/UAV_common/src/controllers/impl/bang_bang.cpp	100
lib/UAV_common/src/controllers/impl/bang_bang.hpp	100
lib/UAV_common/src/controllers/impl/double_setpoint.cpp	100
lib/UAV_common/src/controllers/impl/double_setpoint.hpp	100
lib/UAV_common/src/controllers/impl/PID.cpp	101
lib/UAV_common/src/controllers/impl/PID.hpp	101
lib/UAV_common/src/controllers/impl/PIFF.cpp	101
lib/UAV_common/src/controllers/impl/PIFF.hpp	101
lib/UAV_common/src/logger/logger.cpp	102
lib/UAV_common/src/logger/logger.hpp	102
lib/UAV_common/src/ode/ode.cpp	103
lib/UAV_common/src/ode/ode.hpp	103
lib/UAV_common/src/ode/ode_impl.hpp	103
lib/UAV_common/src/ode/ode_test.cpp	104
lib/UAV_common/src/parser/parser.cpp	106
lib/UAV_common/src/parser/parser.hpp	107
lib/UAV_common/src/parser/uav_params.cpp	108
lib/UAV_common/src/parser/uav_params.hpp	109
lib/UAV_common/src/timed_loop/status.hpp	109
lib/UAV_common/src/timed_loop/timed_loop.cpp	110
lib/UAV_common/src/timed_loop/timed_loop.hpp	110
src/defines.hpp	110
src/main.cpp	111
src/params.cpp	111
src/params.hpp	112
src/simulation.cpp	112
src/simulation.hpp	113
src/state.cpp	113
src/state.hpp	113
tests/integration_test.cpp	114
tests/projectile_parser.hpp	117



## Chapter 5

# Namespace Documentation

### 5.1 controllers Namespace Reference

#### Classes

- class [BangBang](#)
- class [DoubleSetpoint](#)
- class [PID](#)
- class [PIFF](#)

### 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 [DEFAULT\\_AIR\\_DENSITY](#) = 1.224  
*Dry air density in normal conditions in kg/m3.*

#### 5.2.1 Detailed Description

[Simulation](#) constants.

#### 5.2.2 Variable Documentation

#### 5.2.2.1 DEFAULT\_AIR\_DENSITY

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

Dry air density in normal conditions in kg/m3.

#### 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.

## 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 Ammo Class Reference

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

Inheritance diagram for Ammo:

Collaboration diagram for Ammo:

### Public Member Functions

- [Ammo](#) ()=default
- [Ammo](#) (int ammount, double [reload](#), Eigen::Vector3d offset, double mass, Eigen::Vector3d V0)
- [Ammo](#) & [operator=](#) (const [Ammo](#) &other)
- Eigen::Vector3d [getV0](#) ()  
*get start velocity of ammo when launched*

### Protected Attributes

- Eigen::Vector3d [\\_V0](#)

### Additional Inherited Members

#### 6.3.1 Constructor & Destructor Documentation

##### 6.3.1.1 [Ammo\(\)](#) [1/2]

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

##### 6.3.1.2 [Ammo\(\)](#) [2/2]

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

#### 6.3.2 Member Function Documentation

### 6.3.2.1 getV0()

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

get start velocity of ammo when launched

Returns

start velocity vector

### 6.3.2.2 operator=()

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

## 6.3.3 Member Data Documentation

### 6.3.3.1 \_V0

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

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

- lib/UAV\_common/src/components/[loads.hpp](#)
- lib/UAV\_common/src/components/[loads.cpp](#)

## 6.4 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

### 6.4.1 Constructor & Destructor Documentation

#### 6.4.1.1 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

#### 6.4.1.2 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
------------------------	---------------------------------

### 6.4.2 Member Function Documentation

#### 6.4.2.1 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

**6.4.2.2 clear()**

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

clear internal state

Implements [Controller](#).

**6.4.2.3 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](#)

**6.5 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

### 6.5.1 Constructor & Destructor Documentation

#### 6.5.1.1 Cargo() [1/2]

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

#### 6.5.1.2 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](#)

## 6.6 Controller Class Reference

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

Inheritance diagram for Controller:

### Public Member Functions

- [Controller](#) ()  
*Default constructor.*
- [~Controller](#) ()  
*Empty destructor for derived classes.*
- 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](#)

## 6.6.1 Constructor & Destructor Documentation

### 6.6.1.1 Controller()

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

Default constructor.

### 6.6.1.2 ~Controller()

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

Empty deconstructor for derived classes.

## 6.6.2 Member Function Documentation

### 6.6.2.1 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

**6.6.2.2 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::PIFF](#), [controllers::PID](#), and [controllers::DoubleSetpoint](#).

**6.6.2.3 clear()**

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

clear internal state

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

**6.6.2.4 clone()**

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

virtual clone method

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

**6.6.2.5 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
------------------------	---------------------------------

## 6.6.2.6 set\_dt()

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

Set new time step.

## Parameters

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

## 6.6.3 Member Data Documentation

## 6.6.3.1 \_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](#)

## 6.7 ControllerTest Class Reference

Inheritance diagram for ControllerTest:

Collaboration diagram for ControllerTest:

## Protected Member Functions

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

## 6.7.1 Member Function Documentation

### 6.7.1.1 SetUp()

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

### 6.7.1.2 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](#)

## 6.8 ControlSurfaces Class Reference

Aircraft's control surfaces.

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

### Public Member Functions

- [ControlSurfaces](#) ()
  - [ControlSurfaces](#) (int noOfSurfaces, Eigen::Matrix< double, 6,-1 > matrix, Eigen::VectorXd min, Eigen::VectorXd max, Eigen::VectorXd trim)
- Constructor.*
- Eigen::Vector< double, 6 > [getCoefficients](#) () const
  - bool [setValues](#) (Eigen::VectorXd new\_values)
  - void [restoreTrim](#) ()
  - int [getNoOfSurface](#) () const
  - Eigen::VectorXd [getValues](#) () const

### 6.8.1 Detailed Description

Aircraft's control surfaces.

### 6.8.2 Constructor & Destructor Documentation

#### 6.8.2.1 ControlSurfaces() [1/2]

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

#### 6.8.2.2 ControlSurfaces() [2/2]

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

Constructor.

## Parameters

<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

## 6.8.3 Member Function Documentation

### 6.8.3.1 getCoefficients()

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

### 6.8.3.2 getNoOfSurface()

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

### 6.8.3.3 getValues()

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

### 6.8.3.4 restoreTrim()

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

### 6.8.3.5 setValues()

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

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

- [lib/UAV\\_common/src/components/control\\_surfaces.hpp](#)
- [lib/UAV\\_common/src/components/control\\_surfaces.cpp](#)

## 6.9 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

#### 6.9.1 Constructor & Destructor Documentation

##### 6.9.1.1 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 symmetrical to zero



### 6.9.1.2 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
------------------------	---------------------------------

## 6.9.2 Member Function Documentation

### 6.9.2.1 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](#).

### 6.9.2.2 clear()

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

clear internal state

Implements [Controller](#).

### 6.9.2.3 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](#)

## 6.10 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]

### 6.10.1 Detailed Description

[Drive](#) propelling aircraft.

### 6.10.2 Member Data Documentation

#### 6.10.2.1 axis

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

### 6.10.2.2 hinges

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

### 6.10.2.3 noOfHinges

```
int Drive::noOfHinges
```

### 6.10.2.4 position

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

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

- [lib/UAV\\_common/src/components/drive.hpp](#)

## 6.11 DropTest Class Reference

Inheritance diagram for DropTest:

Collaboration diagram for DropTest:

### Protected Member Functions

- void [SetUp](#) () override
- void [TearDown](#) () override
- void [sendControlMessage](#) (std::string msg, bool should\_response=true, bool should\_response\_ok=true)
- int [recvState](#) (std::string &response\_str)
- auto [getParsedState](#) ()
- void [collectSample](#) (const int N=10)

### 6.11.1 Member Function Documentation

#### 6.11.1.1 collectSample()

```
void DropTest::collectSample (  
    const int N = 10 ) [inline], [protected]
```

#### 6.11.1.2 getParsedState()

```
auto DropTest::getParsedState ( ) [inline], [protected]
```

#### 6.11.1.3 recvState()

```
int DropTest::recvState (
    std::string & response_str ) [inline], [protected]
```

#### 6.11.1.4 sendControlMessage()

```
void DropTest::sendControlMessage (
    std::string msg,
    bool should_response = true,
    bool should_response_ok = true ) [inline], [protected]
```

#### 6.11.1.5 SetUp()

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

#### 6.11.1.6 TearDown()

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

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

- tests/[integration\\_test.cpp](#)

## 6.12 EKFScalers Struct Reference

Scalers for EKF.

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

### Public Attributes

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

### 6.12.1 Detailed Description

Scalers for EKF.

### 6.12.2 Member Data Documentation

#### 6.12.2.1 baroScaler

```
double EKFScalers::baroScaler
```

#### 6.12.2.2 predictScaler

```
double EKFScalers::predictScaler
```

#### 6.12.2.3 updateScaler

```
double EKFScalers::updateScaler
```

#### 6.12.2.4 zScaler

```
double EKFScalers::zScaler
```

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

- [lib/UAV\\_common/src/components/navi.hpp](#)

## 6.13 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.*

### 6.13.1 Detailed Description

[Hinge](#) connecting aircraft with drives.

### 6.13.2 Constructor & Destructor Documentation

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

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

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

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

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

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

### 6.13.3 Member Function Documentation

### 6.13.3.1 getRot()

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

Get rotation matrix of orientation change due to hinge.

#### Returns

rotation matrix

### 6.13.3.2 operator=()

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

### 6.13.3.3 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](#)

## 6.14 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](#)

### 6.14.1 Detailed Description

[Jet](#) rocket engine.

### 6.14.2 Member Function Documentation

#### 6.14.2.1 [getLastThrust\(\)](#)

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

get last calculated thrust

##### Returns

last calculated thrust

#### 6.14.2.2 [getThrust\(\)](#)

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

get thrust in specific time

##### Parameters

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



**Returns**

thrust value in Newtons

**6.14.2.3 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

**6.14.3 Member Data Documentation****6.14.3.1 phases**

```
int Jet::phases
```

**6.14.3.2 thrust**

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

**6.14.3.3 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](#)

## 6.15 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 [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)

### 6.15.1 Detailed Description

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

### 6.15.2 Constructor & Destructor Documentation

#### 6.15.2.1 Load() [1/2]

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

#### 6.15.2.2 Load() [2/2]

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

## 6.15.3 Member Function Documentation

### 6.15.3.1 getMass()

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

get mass of load

#### Returns

mass

### 6.15.3.2 getOffset()

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

get offset of load

#### Returns

offset vector

### 6.15.3.3 operator=()

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

### 6.15.3.4 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](#)

## 6.16 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.*

#### 6.16.1 Detailed Description

Log vector data with timestamp in file.

#### 6.16.2 Constructor & Destructor Documentation

##### 6.16.2.1 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>

**6.16.2.2 ~Logger()**

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

Deconstructor.

**6.16.3 Member Function Documentation****6.16.3.1 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

**6.16.3.2 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

### 6.16.3.3 setFmt()

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

Set new format if was not known in constructor.

#### Parameters

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

### 6.16.3.4 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](#)

## 6.17 ObjParams Class Reference

Single obj parameters.

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

### Public Member Functions

- [ObjParams](#) (double [mass](#), double [CS\\_coff](#))  
*Constructor.*
- [ObjParams](#) ([ObjParams](#) &&rhs)  
*Moving constructor.*
- void [setWind](#) (Eigen::Vector3d newWind)  
*Set wind vector affecting on object.*
- Eigen::Vector3d [getWind](#) ()  
*Get wind vector.*
- void [setForce](#) (Eigen::Vector3d newForce)  
*Set outer force applied to object.*
- Eigen::Vector3d [getForce](#) ()  
*Get outer force.*

## Public Attributes

- const int `id`  
*object id*
- const double `mass`  
*object mass*
- const double `CS_coff`  
*aerodynamic drag force cofficent multiplred by aerodynamic field*

### 6.17.1 Detailed Description

Single obj parameters.

### 6.17.2 Constructor & Destructor Documentation

#### 6.17.2.1 ObjParams() [1/2]

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

Constructor.

##### Parameters

<code>mass</code>	object mass
<code>CS_coff</code>	aerodynamic drag force cofficent multiplied by aerodynamic field

#### 6.17.2.2 ObjParams() [2/2]

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

Moving constructor.

##### Parameters

<code>rhs</code>	other instant that should be consumed
------------------	---------------------------------------

### 6.17.3 Member Function Documentation

### 6.17.3.1 getForce()

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

Get outer force.

#### Returns

outer force vector in N

### 6.17.3.2 getWind()

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

Get wind vector.

#### Returns

wind speed vector in m/s

### 6.17.3.3 setForce()

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

Set outer force applied to object.

#### Parameters

<i>newForce</i>	new force vector in N
-----------------	-----------------------

### 6.17.3.4 setWind()

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

Set wind vector affecting on object.

#### Parameters

<i>newWind</i>	new wind speed vector in m/s
----------------	------------------------------



## 6.17.4 Member Data Documentation

### 6.17.4.1 CS\_coff

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

aerodynamic drag force coefficient multiplied by aerodynamic field

### 6.17.4.2 id

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

object id

### 6.17.4.3 mass

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

object mass

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

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

## 6.18 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.*

### 6.18.1 Detailed Description

Ordinal differential equation solver.

### 6.18.2 Member Enumeration Documentation

#### 6.18.2.1 ODEMethod

enum [ODE::ODEMethod](#)

Supported solving method.

Enumerator

Euler	
Heun	
RK4	
PC2	
PC4	
NONE	

### 6.18.3 Constructor & Destructor Documentation

#### 6.18.3.1 ODE()

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

Constructor.

#### 6.18.3.2 ~ODE()

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

Virtual destructor.

### 6.18.4 Member Function Documentation

#### 6.18.4.1 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

#### 6.18.4.2 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

**6.18.4.3 getMicrosteps() [1/2]**

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

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

**Returns**

microsteps

**6.18.4.4 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

**6.18.4.5 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](#)

## 6.19 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

#### 6.19.1 Detailed Description

Explicit Euler algorithm.

#### 6.19.2 Constructor & Destructor Documentation

### 6.19.2.1 ODE\_Euler()

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

## 6.19.3 Member Function Documentation

### 6.19.3.1 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](#)

## 6.20 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

### 6.20.1 Detailed Description

Second order explicit Heun algorithm.

### 6.20.2 Constructor & Destructor Documentation

#### 6.20.2.1 ODE\_Heun()

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

### 6.20.3 Member Function Documentation

#### 6.20.3.1 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`

## 6.21 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

#### 6.21.1 Detailed Description

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

#### 6.21.2 Constructor & Destructor Documentation

##### 6.21.2.1 ODE\_PC2()

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

#### 6.21.3 Member Function Documentation

##### 6.21.3.1 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`

## 6.22 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

#### 6.22.1 Detailed Description

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

#### 6.22.2 Constructor & Destructor Documentation

### 6.22.2.1 ODE\_PC4()

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

## 6.22.3 Member Function Documentation

### 6.22.3.1 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](#)

## 6.23 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

### 6.23.1 Detailed Description

Fourth order Runge Kutta algorithm.

### 6.23.2 Constructor & Destructor Documentation

#### 6.23.2.1 ODE\_RK4()

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

### 6.23.3 Member Function Documentation

#### 6.23.3.1 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](#)

## 6.24 ODETest Class Reference

Inheritance diagram for ODETest:

Collaboration diagram for ODETest:

### Protected Member Functions

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

### 6.24.1 Member Function Documentation

#### 6.24.1.1 [SetUp\(\)](#)

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

#### 6.24.1.2 [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](#)

## 6.25 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.*

### 6.25.1 Detailed Description

[Simulation](#) parameters.

### 6.25.2 Constructor & Destructor Documentation

#### 6.25.2.1 Params() [1/3]

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

Constructor.

#### 6.25.2.2 Params() [2/3]

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

### 6.25.2.3 Params() [3/3]

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

### 6.25.2.4 ~Params()

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

Deconstructor.

## 6.25.3 Member Function Documentation

### 6.25.3.1 getSingleton()

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

Get singleton of [Params](#).

#### Returns

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

### 6.25.3.2 operator=()

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

## 6.25.4 Member Data Documentation

### 6.25.4.1 ODE\_METHOD

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

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

### 6.25.4.2 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](#)

## 6.26 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 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

## 6.26.1 Member Enumeration Documentation

### 6.26.1.1 AntiWindUpMode

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

Methods of handling windup in controller.

## Enumerator

NONE	
CLAMPING	

## 6.26.2 Constructor & Destructor Documentation

### 6.26.2.1 PID() [1/2]

```
PID::PID (
    double Kp,
    double Ki,
    double Kd,
    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>min</i>	saturation - lower range limit
<i>max</i>	saturation - upper range limit
<i>antiWindUp</i>	antiwindup method

### 6.26.2.2 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
------------------------	---------------------------------

## 6.26.3 Member Function Documentation



### 6.26.3.1 calc()

```
double 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](#).

### 6.26.3.2 clear()

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

clear internal state

Implements [Controller](#).

### 6.26.3.3 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](#)

## 6.27 controllers::PIFF Class Reference

```
#include <PIFF.hpp>
```

Inheritance diagram for controllers::PIFF:

Collaboration diagram for controllers::PIFF:

## Public Types

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

## Public Member Functions

- [PIFF](#) (double Kp, double Ki, double Kff, double min=-std::numeric\_limits< double >::max(), double max=std::numeric\_limits< double >::max(), [AntiWindUpMode](#) antiWindUp=[AntiWindUpMode::CLAMPING](#))  
*Constructor with all [PID](#) controller parameters.*
- [PIFF](#) (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

### 6.27.1 Member Enumeration Documentation

#### 6.27.1.1 AntiWindUpMode

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

Methods of handling windup in controller.

Enumerator

NONE	
CLAMPING	

### 6.27.2 Constructor & Destructor Documentation

#### 6.27.2.1 PIFF() [1/2]

```
PIFF::PIFF (
    double Kp,
    double Ki,
```

```
double Kff,
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>Kff</i>	FF term
<i>min</i>	saturation - lower range limit
<i>max</i>	saturation - upper range limit
<i>antiWindUp</i>	antiwindup method

### 6.27.2.2 PIFF() [2/2]

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

Construct controller with parameters from xml.

#### Parameters

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

## 6.27.3 Member Function Documentation

### 6.27.3.1 calc()

```
double PIFF::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](#).

**6.27.3.2 clear()**

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

clear internal state

Implements [Controller](#).

**6.27.3.3 clone()**

```
std::unique_ptr< Controller > PIFF::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/[PIFF.hpp](#)
- lib/UAV\_common/src/controllers/impl/[PIFF.cpp](#)

## 6.28 Projectile Struct Reference

```
#include <projectile_parser.hpp>
```

**Public Attributes**

- int [id](#)
- Eigen::Vector3d [position](#)
- Eigen::Vector3d [velocity](#)

**6.28.1 Member Data Documentation**

### 6.28.1.1 id

```
int Projectile::id
```

### 6.28.1.2 position

```
Eigen::Vector3d Projectile::position
```

### 6.28.1.3 velocity

```
Eigen::Vector3d Projectile::velocity
```

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

- tests/[projectile\\_parser.hpp](#)

## 6.29 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](#)

### 6.29.1 Detailed Description

[Rotor](#) engine with controlled speed.

### 6.29.2 Member Data Documentation

#### 6.29.2.1 direction

```
int Rotor::direction
```

#### 6.29.2.2 forceCoff

```
double Rotor::forceCoff
```

#### 6.29.2.3 hoverSpeed

```
double Rotor::hoverSpeed
```

#### 6.29.2.4 maxSpeed

```
double Rotor::maxSpeed
```

#### 6.29.2.5 timeConstant

```
double Rotor::timeConstant
```

#### 6.29.2.6 torqueCoff

```
double Rotor::torqueCoff
```

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

- [lib/UAV\\_common/src/components/drive.hpp](#)

## 6.30 SensorParams Struct Reference

Base parameters of a sensor.

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

## Public Attributes

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

### 6.30.1 Detailed Description

Base parameters of a sensor.

### 6.30.2 Member Data Documentation

#### 6.30.2.1 [bias](#)

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

#### 6.30.2.2 [name](#)

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

#### 6.30.2.3 [refreshTime](#)

```
double SensorParams::refreshTime
```

#### 6.30.2.4 [sd](#)

```
double SensorParams::sd
```

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

- lib/UAV\_common/src/components/[navi.hpp](#)

## 6.31 Simulation Class Reference

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

## Public Member Functions

- [Simulation](#) (const [Params](#) &params)  
*Constructor.*
- [~Simulation](#) ()  
*Destructor.*
- void [run](#) ()  
*Run simulation.*
- int [addObj](#) (double mass, double CS, Eigen::Vector3d pos, Eigen::Vector3d vel=Eigen::Vector3d())  
*Add new object to simulation.*
- void [removeObj](#) (int id)  
*Remove object from simulation.*
- void [addCommand](#) (std::string msg, zmq::socket\_t &sock)  
*Handle add new object command.*
- void [updateWind](#) (std::string msg, zmq::socket\_t &sock)  
*Handle update wind command.*
- void [updateForce](#) (std::string msg, zmq::socket\_t &sock)  
*Handle update force command.*
- void [solidSurfColision](#) (std::string &msg\_str, zmq::socket\_t &sock)  
*Handle solid surface collision command.*
- void [calcImpulseForce](#) (int id, double COR, double mi\_static, double mi\_dynamic, Eigen::Vector3d surface↔Normal)  
*Calculates object state after collision with given surface.*

### 6.31.1 Constructor & Destructor Documentation

#### 6.31.1.1 Simulation()

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

Constructor.

Parameters

<i>params</i>	simulation params
---------------	-------------------

#### 6.31.1.2 ~Simulation()

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

Destructor.



## 6.31.2 Member Function Documentation

### 6.31.2.1 addCommand()

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

Handle add new object command.

#### Parameters

<i>msg</i>	message content
<i>sock</i>	zmq socket reply is send by

### 6.31.2.2 addObj()

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

Add new object to simulation.

#### Parameters

<i>mass</i>	obj mass
<i>CS</i>	aerodynamic drag force cofficent multiplied by aerodynamic field
<i>pos</i>	start position of object
<i>vel</i>	start velocity of object

#### Returns

id of added object

### 6.31.2.3 calcImpulseForce()

```
void Simulation::calcImpulseForce (
    int id,
    double COR,
    double mi_static,
```

```
double mi_dynamic,  
Eigen::Vector3d surfaceNormal )
```

Calculates object state after collision with given surface.

## Parameters

<i>id</i>	object id
<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>surfaceNormal</i>	surface normal vector

**6.31.2.4 removeObj()**

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

Remove object from simulation.

## Parameters

<i>id</i>	object id
-----------	-----------

**6.31.2.5 run()**

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

Run simulation.

**6.31.2.6 solidSurfColision()**

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

Handle solid surface collision command.

## Parameters

<i>msg</i>	message content
<i>sock</i>	zmq socket reply is send by

### 6.31.2.7 updateForce()

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

Handle update force command.

#### Parameters

<i>msg</i>	message content
<i>sock</i>	zmq socket reply is send by

### 6.31.2.8 updateWind()

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

Handle update wind command.

#### Parameters

<i>msg</i>	message content
<i>sock</i>	zmq socket reply is send by

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

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

## 6.32 State Class Reference

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

### Public Member Functions

- [State](#) ()  
*Constructor.*
- [Eigen::VectorXd getState](#) ()  
*Get full state as vector.*
- void [updateState](#) (Eigen::VectorXd newState)  
*Update state.*
- void [updateWind](#) (int id, Eigen::Vector3d newWind)  
*update wind speed for obj specified by id*

- void `updateForce` (int id, Eigen::Vector3d newForce)  
*update outer force applied to object specified by id*
- int `addObj` (double mass, double CS\_coff, Eigen::Vector3d pos, Eigen::Vector3d vel=Eigen::Vector3d())  
*Add new object to simulation.*
- void `removeObj` (int id)  
*remove object specified by id*
- std::string `to_string` ()  
*Serialize state to string.*
- int `findIndex` (int id)  
*Find index of object specified by id.*
- int `getNoObj` ()  
*Get number of active object in simulation.*
- `ObjParams` \* `getParams` (int index)  
*get params of object specified by index*
- Eigen::Vector3d `getPos` (int index)  
*Get position of object specified by index.*
- Eigen::Vector3d `getVel` (int index)  
*Get velocity of object specified by index.*
- void `setVel` (int index, Eigen::Vector3d newVel)  
*Override velocity of object, for example after collision.*

## Public Attributes

- std::mutex `stateMutex`  
*mutex to manipule on state responses*
- double `real_time`  
*time of simulation*
- `Status` `status`  
*status for timed loop*

## 6.32.1 Constructor & Destructor Documentation

### 6.32.1.1 State()

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

Constructor.

## 6.32.2 Member Function Documentation

### 6.32.2.1 addObj()

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

Add new object to simulation.

**Parameters**

<i>mass</i>	mass of object
<i>CS_coff</i>	aerodynamic drag force cofficent multiplied by aerodynamic field
<i>pos</i>	start position
<i>vel</i>	start velocity

**Returns**

id of added object

**6.32.2.2 findIndex()**

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

Find index of object specified by id.

**Parameters**

<i>id</i>	object id
-----------	-----------

**Returns**

object index

**6.32.2.3 getNoObj()**

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

Get number of active object in simulation.

**Returns**

number of object

**6.32.2.4 getParams()**

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

get params of object specified by index

**Parameters**

<i>index</i>	index of object
--------------	-----------------

**Returns**

pointer to object params

**6.32.2.5 getPos()**

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

Get position of object specified by index.

**Parameters**

<i>index</i>	index of object
--------------	-----------------

**Returns**

position vector

**6.32.2.6 getState()**

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

Get full state as vector.

**Returns**

state vector

**6.32.2.7 getVel()**

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

Get velocity of object specified by index.

**Parameters**

<i>index</i>	index of object
--------------	-----------------

**Returns**

velocity of object

**6.32.2.8 removeObj()**

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

remove object specified by id

**Parameters**

<i>id</i>	id of removing object
-----------	-----------------------

**6.32.2.9 setVel()**

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

Override velocity of object, for example after collision.

**Parameters**

<i>index</i>	index of object
<i>newVel</i>	new velocity vector

**6.32.2.10 to\_string()**

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

Serialize state to string.

**Returns**

serialized state



### 6.32.2.11 updateForce()

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

update outer force applied to object specified by id

#### Parameters

<i>id</i>	id of updated obj
<i>newForce</i>	new force value

### 6.32.2.12 updateState()

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

Update state.

#### Parameters

<i>newState</i>	new state vector
-----------------	------------------

### 6.32.2.13 updateWind()

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

update wind speed for obj specified by id

#### Parameters

<i>id</i>	id of updated obj
<i>newWind</i>	new wind speed vector

## 6.32.3 Member Data Documentation

### 6.32.3.1 real\_time

`double State::real_time`  
time of simulation

### 6.32.3.2 stateMutex

`std::mutex State::stateMutex`  
mutex to manipulate on state responses

### 6.32.3.3 status

`Status State::status`  
status for timed loop

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

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

## 6.33 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*

### 6.33.1 Detailed Description

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

### 6.33.2 Constructor & Destructor Documentation

#### 6.33.2.1 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

### 6.33.3 Member Function Documentation

#### 6.33.3.1 go() [1/2]

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

start infinite loop

#### 6.33.3.2 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

## 6.34 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](#)

### 6.34.1 Detailed Description

Parsed UAV configuration from XML.

## 6.34.2 Constructor & Destructor Documentation

### 6.34.2.1 UAVparams()

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

Initialize default data.

### 6.34.2.2 ~UAVparams()

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

## 6.34.3 Member Function Documentation

### 6.34.3.1 getRotorHoverSpeeds()

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

### 6.34.3.2 getRotorMaxSpeeds()

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

### 6.34.3.3 getRotorTimeContants()

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

### 6.34.3.4 getSingleton()

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

#### 6.34.3.5 loadConfig()

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

### 6.34.4 Member Data Documentation

#### 6.34.4.1 aero\_coffs

[AeroCoefficients](#) UAVparams::aero\_coffs

#### 6.34.4.2 ahrs

[AHRSParams](#) UAVparams::ahrs

#### 6.34.4.3 ammo

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

#### 6.34.4.4 cargo

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

#### 6.34.4.5 controllers

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

#### 6.34.4.6 ekf

[EKFScalers](#) UAVparams::ekf

#### 6.34.4.7 initialMode

`std::string UAVparams::initialMode`

#### 6.34.4.8 initialOrientation

`Eigen::Vector3d UAVparams::initialOrientation`

#### 6.34.4.9 initialPosition

`Eigen::Vector3d UAVparams::initialPosition`

#### 6.34.4.10 initialVelocity

`Eigen::Vector3d UAVparams::initialVelocity`

#### 6.34.4.11 instantRun

`bool UAVparams::instantRun`

#### 6.34.4.12 Ix

`double UAVparams::Ix`

#### 6.34.4.13 Ixy

`double UAVparams::Ixy`

#### 6.34.4.14 Ixz

`double UAVparams::Ixz`

**6.34.4.15 ly**

```
double UAVparams::Iy
```

**6.34.4.16 Iyz**

```
double UAVparams::Iyz
```

**6.34.4.17 Iz**

```
double UAVparams::Iz
```

**6.34.4.18 jets**

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

**6.34.4.19 m**

```
double UAVparams::m
```

**6.34.4.20 name**

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

**6.34.4.21 noOfAmmo**

```
int UAVparams::noOfAmmo
```

**6.34.4.22 noOfCargo**

```
int UAVparams::noOfCargo
```



#### 6.34.4.23 noOfJets

```
int UAVparams::noOfJets
```

#### 6.34.4.24 noOfRotors

```
int UAVparams::noOfRotors
```

#### 6.34.4.25 rotorMixer

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

#### 6.34.4.26 rotors

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

#### 6.34.4.27 sensors

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

#### 6.34.4.28 surfaceMixer

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

#### 6.34.4.29 surfaces

```
ControlSurfaces UAVparams::surfaces
```

#### 6.34.4.30 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](#)



## 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/drop.dir/src/main.cpp.o.d File Reference

### 7.4 build/CMakeFiles/drop.dir/src/params.cpp.o.d File Reference

### 7.5 build/CMakeFiles/drop.dir/src/simulation.cpp.o.d File Reference

### 7.6 build/CMakeFiles/drop.dir/src/state.cpp.o.d File Reference

### 7.7 build/CMakeFiles/integration\_test.dir/tests/integration\_test.cpp.o.d File Reference

### 7.8 build/lib/UAV\_common/CMakeFiles/common.dir/src/components/control\_surfaces.cpp.o.d File Reference

### 7.9 build/lib/UAV\_common/CMakeFiles/common.dir/src/components/drive.cpp.o.d File Reference

### 7.10 build/lib/UAV\_common/CMakeFiles/common.dir/src/components/hinge.cpp.o.d File Reference

### 7.11 build/lib/UAV\_common/CMakeFiles/common.dir/src/components/loads.cpp.o.d File Reference

### 7.12 build/lib/UAV\_common/CMakeFiles/common.dir/src/controllers/controller.cpp.o.d File Reference

### 7.13 build/lib/UAV\_common/CMakeFiles/common.dir/src/controllers/impl/bang\_bang.cpp.o.d File Reference

### 7.14 build/lib/UAV\_common/CMakeFiles/common.dir/src/controllers/impl/double\_setpoint.cpp.o.d File Reference

### 7.15 build/lib/UAV\_common/CMakeFiles/common.dir/src/controllers/impl/PID.cpp.o.d File Reference

### 7.16 build/lib/UAV\_common/CMakeFiles/common.dir/src/PID/PID.cpp.o.d File Reference

### 7.17 build/lib/UAV\_common/CMakeFiles/common.dir/src/controllers/impl/PIFF.cpp.o.d File Reference

### 7.18 build/lib/UAV\_common/CMakeFiles/common.dir/src/logger/logger.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:

## 7.26 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`

### 7.26.1 Function Documentation

#### 7.26.1.1 `legend()`

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

### 7.26.1.2 plot()

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

### 7.26.1.3 title()

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

### 7.26.1.4 xlabel()

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

### 7.26.1.5 ylabel()

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

## 7.26.2 Variable Documentation

### 7.26.2.1 clc

```
clc
```

### 7.26.2.2 csvFiles

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

#### 7.26.2.3 data

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

#### 7.26.2.4 figure

```
figure
```

#### 7.26.2.5 folderPath

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

#### 7.26.2.6 i

```
for i
```

##### Initial value:

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

#### 7.26.2.7 off

```
hold off
```

#### 7.26.2.8 on

```
hold on
```

#### 7.26.2.9 x

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

## 7.26.2.10 y

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

## 7.27 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.*

## 7.28 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:

## 7.29 lib/UAV\_common/src/components/control\_surfaces.cpp File Reference

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

## 7.30 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.*

## 7.31 lib/UAU\_common/src/components/drive.cpp File Reference

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

Include dependency graph for drive.cpp:

## 7.32 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.

## 7.33 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)

### 7.33.1 Function Documentation

#### 7.33.1.1 asSkewMatrix()

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

## 7.34 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.*

## 7.35 lib/UAV\_common/src/components/loads.cpp File Reference

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

## 7.36 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](#)

## 7.37 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.*

## 7.38 lib/UAV\_common/src/controllers/controller.cpp File Reference

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

## 7.39 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](#)

## 7.40 lib/UAV\_common/src/controllers/controller\_test.cpp File Reference

```
#include "impl/PID.hpp"
#include "impl/PIFF.hpp"
#include "impl/bang_bang.hpp"
#include "impl/double_setpoint.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"`

### 7.40.1 Function Documentation

#### 7.40.1.1 getMethodsToTest()

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

#### 7.40.1.2 INSTANTIATE\_TEST\_SUITE\_P()

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

#### 7.40.1.3 main()

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

#### 7.40.1.4 TEST\_P() [1/2]

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

#### 7.40.1.5 TEST\_P() [2/2]

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

### 7.40.2 Variable Documentation

#### 7.40.2.1 plot

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

#### 7.40.2.2 `plot_directory_name`

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

### 7.41 `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:
```

### 7.42 `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](#)

### 7.43 `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:
```

### 7.44 `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](#)

## 7.45 lib/UAV\_common/src/controllers/impl/PID.cpp File Reference

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

## 7.46 lib/UAV\_common/src/controllers/impl/PID.hpp File Reference

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

## Classes

- class [controllers::PID](#)

## Namespaces

- [controllers](#)

## 7.47 lib/UAV\_common/src/controllers/impl/PIFF.cpp File Reference

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

## 7.48 lib/UAV\_common/src/controllers/impl/PIFF.hpp File Reference

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

## Classes

- class [controllers::PIFF](#)

## Namespaces

- [controllers](#)

## 7.49 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)

### 7.49.1 Function Documentation

#### 7.49.1.1 shouldLog()

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

## 7.50 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

### 7.50.1 Macro Definition Documentation

#### 7.50.1.1 [LOGGER\\_MASK](#)

```
#define LOGGER_MASK -1
```

## 7.51 lib/UAV\_common/src/ode/ode.cpp File Reference

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

## 7.52 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.*

## 7.53 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.*

## 7.54 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)`

### 7.54.1 Function Documentation

#### 7.54.1.1 [getMethodsToTest\(\)](#)

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

#### 7.54.1.2 [INSTANTIATE\\_TEST\\_SUITE\\_P\(\)](#)

```
INSTANTIATE\_TEST\_SUITE\_P (
    TestDerivedClasses ,
    ODETest ,
    testing::ValuesIn(getMethodsToTest ()) )
```

#### 7.54.1.3 [main\(\)](#)

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



#### 7.54.1.4 TEST\_F() [1/2]

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

#### 7.54.1.5 TEST\_F() [2/2]

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

#### 7.54.1.6 TEST\_P() [1/4]

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

#### 7.54.1.7 TEST\_P() [2/4]

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

#### 7.54.1.8 TEST\_P() [3/4]

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

#### 7.54.1.9 TEST\_P() [4/4]

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

## 7.55 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.*

### 7.55.1 Function Documentation

#### 7.55.1.1 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

#### 7.55.1.2 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

## 7.56 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.*

### 7.56.1 Function Documentation

#### 7.56.1.1 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

### 7.56.1.2 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

## 7.57 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)

### 7.57.1 Function Documentation

#### 7.57.1.1 parseHinge()

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

## 7.58 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.*

## 7.59 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*

### 7.59.1 Enumeration Type Documentation

#### 7.59.1.1 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

## 7.60 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:
```

## 7.61 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.*

## 7.62 src/defines.hpp File Reference

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

### Namespaces

- [def](#)  
*Simulation constants.*

### Variables

- const double [def::GRAVITY\\_CONST](#) = 9.81  
*Gravity constant on Earth in m/s2.*
- const double [def::FRICTION\\_EPS](#) = 0.001  
*minimal friction that is calculated (numerical float eps)*
- const double [def::GENTLY\\_PUSH](#) = 0.15  
*artificial force coefficient. Protect again diving objects in horizontal wall*
- const double [def::DEFAULT\\_AIR\\_DENSITY](#) = 1.224  
*Dry air density in normal conditions in kg/m3.*

## 7.63 src/main.cpp File Reference

```
#include <iostream>
#include <Eigen/Dense>
#include <cxxopts.hpp>
#include "simulation.hpp"
#include "common.hpp"
#include "params.hpp"
Include dependency graph for main.cpp:
```

### Functions

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

### 7.63.1 Function Documentation

#### 7.63.1.1 main()

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

#### 7.63.1.2 parseArgs()

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

Parse CL arguments.

#### Parameters

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

## 7.64 src/params.cpp File Reference

```
#include "params.hpp"
```

```
#include <iostream>
```

Include dependency graph for params.cpp:

## 7.65 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.*

## 7.66 src/simulation.cpp File Reference

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

```
#include <zmq.hpp>
```

```
#include <iostream>
```

```
#include <cstdio>
```

```
#include <thread>
```

```
#include <mutex>
```

```
#include <functional>
```

```
#include <map>
```

```
#include <filesystem>
```

```
#include "simulation.hpp"
```

```
#include "common.hpp"
```

```
#include "state.hpp"
```

Include dependency graph for simulation.cpp:

### Functions

- bool [isNormal](#) (double factor)

### 7.66.1 Function Documentation

#### 7.66.1.1 isNormal()

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



## 7.67 src/simulation.hpp File Reference

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

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

### Classes

- class [Simulation](#)

## 7.68 src/state.cpp File Reference

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

Include dependency graph for state.cpp:

## 7.69 src/state.hpp File Reference

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

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

### Classes

- class [ObjParams](#)  
*Single obj parameters.*
- class [State](#)

## 7.70 tests/integration\_test.cpp File Reference

```
#include <gtest/gtest.h>
#include <zmq.hpp>
#include <Eigen/Dense>
#include <cstdlib>
#include <string>
#include <iostream>
#include <sstream>
#include <thread>
#include <future>
#include <cmath>
#include <chrono>
#include "projectile_parser.hpp"
Include dependency graph for integration_test.cpp:
```

### Classes

- class [DropTest](#)

### Functions

- [TEST](#) (IntegrationTest, ProgramShowsHelp)  
*Test if program show correct usage.*
- [TEST\\_F](#) ([DropTest](#), ProgramRunsAndExitsCorrectly)  
*Test if program run and exit on command correctly.*
- [TEST\\_F](#) ([DropTest](#), ProgramRunsAndSendState)  
*Test if program run send valid state.*
- [TEST\\_F](#) ([DropTest](#), AddAndRemoveProjectiles)  
*Test if program handle add object and remove object command correctly.*
- [TEST\\_F](#) ([DropTest](#), CheckFreeFall)  
*Test if program simulates free fall correctly.*
- [TEST\\_F](#) ([DropTest](#), LikeParachute)  
*Test if program simulates air drag correctly.*
- [TEST\\_F](#) ([DropTest](#), ForceInfluence)  
*Test if program simulates outer forces influence correctly.*
- [TEST\\_F](#) ([DropTest](#), SolidSurfaceCollision)  
*Test if program simulates solid surface collision correctly.*
- [TEST\\_F](#) ([DropTest](#), StrongWindInfluence)  
*Test if program simulates strong wind influence correctly.*
- int [main](#) (int argc, char \*\*argv)

### Variables

- const std::string [programPath](#) = "./drop"
- const std::string [communicationFolder](#) = "ipc:///tmp/drop\_shot"
- zmq::context\_t [ctx](#)

## 7.70.1 Function Documentation

### 7.70.1.1 main()

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

### 7.70.1.2 TEST()

```
TEST (
    IntegrationTest ,
    ProgramShowsHelp )
```

Test if program show correct usage.

### 7.70.1.3 TEST\_F() [1/8]

```
TEST_F (
    DropTest ,
    AddAndRemoveProjectiles )
```

Test if program handle add object and remove object command correctly.

### 7.70.1.4 TEST\_F() [2/8]

```
TEST_F (
    DropTest ,
    CheckFreeFall )
```

Test if program simulates free fall correctly.

### 7.70.1.5 TEST\_F() [3/8]

```
TEST_F (
    DropTest ,
    ForceInfluence )
```

Test if program simulates outer forces influence correctly.

**7.70.1.6 TEST\_F() [4/8]**

```
TEST_F (
    DropTest ,
    LikeParachute )
```

Test if program simulates air drag correctly.

**7.70.1.7 TEST\_F() [5/8]**

```
TEST_F (
    DropTest ,
    ProgramRunsAndExitsCorrectly )
```

Test if program run and exit on command correctly.

**7.70.1.8 TEST\_F() [6/8]**

```
TEST_F (
    DropTest ,
    ProgramRunsAndSendState )
```

Test if program run send valid state.

**7.70.1.9 TEST\_F() [7/8]**

```
TEST_F (
    DropTest ,
    SolidSurfaceCollision )
```

Test if program simulates solid surface collision correctly.

**7.70.1.10 TEST\_F() [8/8]**

```
TEST_F (
    DropTest ,
    StrongWindInfluence )
```

Test if program simulates strong wind influence correctly.

## 7.70.2 Variable Documentation

### 7.70.2.1 communicationFolder

```
const std::string communicationFolder = "ipc:///tmp/drop_shot"
```

### 7.70.2.2 ctx

```
zmq::context_t ctx
```

### 7.70.2.3 programPath

```
const std::string programPath = "./drop"
```

## 7.71 tests/projectile\_parser.hpp File Reference

```
#include <iostream>
#include <sstream>
#include <vector>
#include <Eigen/Dense>
```

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

### Classes

- struct [Projectile](#)

### Functions

- bool [checkInputErrors](#) (std::stringstream &ss, char expectedDelimiter, const std::string &errorMessage)
- std::pair< double, std::vector< [Projectile](#) > > [parseInput](#) (const std::string &input)

## 7.71.1 Function Documentation

### 7.71.1.1 checkInputErrors()

```
bool checkInputErrors (
    std::stringstream & ss,
    char expectedDelimiter,
    const std::string & errorMessage )
```

### 7.71.1.2 parseInput()

```
std::pair<double, std::vector<Projectile> > parseInput (
    const std::string & input )
```



# Index

- `_V0`
    - Ammo, [15](#)
  - `__has_include`
    - CMakeCCompilerId.c, [83](#)
    - CMakeCXXCompilerId.cpp, [87](#)
  - `_dt`
    - Controller, [21](#)
  - `~Controller`
    - Controller, [19](#)
  - `~Logger`
    - Logger, [37](#)
  - `~ODE`
    - ODE, [43](#)
  - `~Params`
    - Params, [54](#)
  - `~Simulation`
    - Simulation, [64](#)
  - `~UAVparams`
    - UAVparams, [77](#)
- `addCommand`
  - Simulation, [65](#)
- `addObj`
  - Simulation, [65](#)
  - State, [69](#)
- `aero_coffs`
  - UAVparams, [78](#)
- `AeroCoefficients`, [11](#)
  - C0, [11](#)
  - Cab, [11](#)
  - Cpqr, [12](#)
  - d, [12](#)
  - eAR, [12](#)
  - S, [12](#)
  - stallLimit, [12](#)
- `ahrs`
  - UAVparams, [78](#)
- `AHRSPParams`, [12](#)
  - alpha, [13](#)
  - Q, [13](#)
  - R, [13](#)
  - type, [13](#)
- `alpha`
  - AHRSPParams, [13](#)
- `Ammo`, [14](#)
  - `_V0`, [15](#)
  - Ammo, [14](#)
  - `getV0`, [14](#)
  - operator=, [15](#)
- `ammo`

- UAVparams, [78](#)
- `AntiWindUpMode`
  - controllers::PID, [55](#)
  - controllers::PIFF, [58](#)
- `ARCHITECTURE_ID`
  - CMakeCCompilerId.c, [84](#)
  - CMakeCXXCompilerId.cpp, [87](#)
- `asSkewMatrix`
  - hinge.cpp, [96](#)
- `axis`
  - Drive, [26](#)
- `BangBang`
  - controllers::BangBang, [16](#)
- `baroScaler`
  - EKFScalers, [29](#)
- `bias`
  - SensorParams, [63](#)
- `build/CMakeFiles/3.22.1/CompilerIdC/CMakeCCompilerId.c`, [83](#)
- `build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp`, [86](#)
- `build/CMakeFiles/drop.dir/src/main.cpp.o.d`, [91](#)
- `build/CMakeFiles/drop.dir/src/params.cpp.o.d`, [91](#)
- `build/CMakeFiles/drop.dir/src/simulation.cpp.o.d`, [91](#)
- `build/CMakeFiles/drop.dir/src/state.cpp.o.d`, [91](#)
- `build/CMakeFiles/integration_test.dir/tests/integration_test.cpp.o.d`, [91](#)
- `build/lib/UAV_common/CMakeFiles/common.dir/src/components/control_s`, [91](#)
- `build/lib/UAV_common/CMakeFiles/common.dir/src/components/drive.cpp`, [91](#)
- `build/lib/UAV_common/CMakeFiles/common.dir/src/components/hinge.cpp`, [91](#)
- `build/lib/UAV_common/CMakeFiles/common.dir/src/components/loads.cpp`, [91](#)
- `build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/controller.c`, [91](#)
- `build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/bang_`, [91](#)
- `build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/double`, [91](#)
- `build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/PID.cp`, [91](#)
- `build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/PIFF.c`, [91](#)
- `build/lib/UAV_common/CMakeFiles/common.dir/src/logger/logger.cpp.o.d`, [91](#)
- `build/lib/UAV_common/CMakeFiles/common.dir/src/ode/ode.cpp.o.d`, [91](#)

- build/lib/UAV\_common/CMakeFiles/common.dir/src/parser/parser.cpp, 91
- build/lib/UAV\_common/CMakeFiles/common.dir/src/parser/uav\_parser.cpp, 91
- build/lib/UAV\_common/CMakeFiles/common.dir/src/PID/PID.cpp, 91
- build/lib/UAV\_common/CMakeFiles/common.dir/src/timed\_loop/timed\_loop.cpp, 91
- build/lib/UAV\_common/CMakeFiles/Controller\_test.dir/src/controller\_test.cpp.o.d, 91
- build/lib/UAV\_common/CMakeFiles/ODE\_test.dir/src/ode\_test.cpp.o.d, 91
- C0
  - AeroCoefficients, 11
- C\_VERSION
  - CMakeCCompilerId.c, 84
- Cab
  - AeroCoefficients, 11
- calc
  - Controller, 19, 20
  - controllers::BangBang, 16
  - controllers::DoubleSetpoint, 25
  - controllers::PID, 56
  - controllers::PIFF, 59
- calcImpulseForce
  - Simulation, 65
- Cargo, 17
  - Cargo, 18
- cargo
  - UAVparams, 78
- checkInputErrors
  - projectile\_parser.hpp, 117
- CLAMPING
  - controllers::PID, 56
  - controllers::PIFF, 58
- clc
  - controller\_plots.m, 93
- clear
  - Controller, 20
  - controllers::BangBang, 17
  - controllers::DoubleSetpoint, 25
  - controllers::PID, 57
  - controllers::PIFF, 60
- clone
  - Controller, 20
  - controllers::BangBang, 17
  - controllers::DoubleSetpoint, 25
  - controllers::PID, 57
  - controllers::PIFF, 60
- CMakeCCompilerId.c
  - \_\_has\_include, 83
  - ARCHITECTURE\_ID, 84
  - C\_VERSION, 84
  - COMPILER\_ID, 84
  - DEC, 84
  - HEX, 84
  - info\_arch, 85
  - info\_compiler, 85
  - info\_language\_extensions\_default, 85
  - info\_language\_standard\_default, 86
  - info\_platform, 86
  - main, 85
  - PLATFORM\_ID, 84
  - STRINGIFY, 85
  - STRINGIFY\_HELPER, 85
  - CMakeCXXCompilerId.cpp, 85
  - has\_include, 85
  - st.cpp.o.d, 85
  - ARCHITECTURE\_ID, 87
  - COMPILER\_ID, 87
  - CXX\_STD, 87
  - DEC, 87
  - HEX, 87
  - info\_arch, 88
  - info\_compiler, 89
  - info\_language\_extensions\_default, 89
  - info\_language\_standard\_default, 89
  - info\_platform, 89
  - main, 88
  - PLATFORM\_ID, 88
  - STRINGIFY, 88
  - STRINGIFY\_HELPER, 88
- collectSample
  - DropTest, 27
- communicationFolder
  - integration\_test.cpp, 117
- COMPILER\_ID
  - CMakeCCompilerId.c, 84
  - CMakeCXXCompilerId.cpp, 87
- Controller, 18
  - \_dt, 21
  - ~Controller, 19
  - calc, 19, 20
  - clear, 20
  - clone, 20
  - Controller, 19
  - ControllerFactory, 20
  - set\_dt, 21
- controller\_plots.m
  - clc, 93
  - csvFiles, 93
  - data, 93
  - figure, 94
  - folderPath, 94
  - i, 94
  - legend, 92
  - off, 94
  - on, 94
  - plot, 92
  - title, 93
  - x, 94
  - xlabel, 93
  - y, 94
  - ylabel, 93
- controller\_test.cpp
  - getMethodsToTest, 98
  - INstantiate\_TEST\_SUITE\_P, 99



- main, 99
- plot, 99
- plot\_directory\_name, 99
- TEST\_P, 99
- ControllerFactory
  - Controller, 20
- controllers, 9
  - UAVparams, 78
- controllers::BangBang, 15
  - BangBang, 16
  - calc, 16
  - clear, 17
  - clone, 17
- controllers::DoubleSetpoint, 24
  - calc, 25
  - clear, 25
  - clone, 25
  - DoubleSetpoint, 24
- controllers::PID, 55
  - AntiWindUpMode, 55
  - calc, 56
  - CLAMPING, 56
  - clear, 57
  - clone, 57
  - NONE, 56
  - PID, 56
- controllers::PIFF, 57
  - AntiWindUpMode, 58
  - calc, 59
  - CLAMPING, 58
  - clear, 60
  - clone, 60
  - NONE, 58
  - PIFF, 58, 59
- ControllerTest, 21
  - SetUp, 21
  - TearDown, 22
- ControlSurfaces, 22
  - ControlSurfaces, 22
  - getCoefficients, 23
  - getNoOfSurface, 23
  - getValues, 23
  - restoreTrim, 23
  - setValues, 23
- Cpqr
  - AeroCoefficients, 12
- CS\_coff
  - ObjParams, 41
- csvFiles
  - controller\_plots.m, 93
- ctx
  - integration\_test.cpp, 117
- CXX\_STD
  - CMakeCXXCompilerId.cpp, 87
- d
  - AeroCoefficients, 12
- data
  - controller\_plots.m, 93
- DEC
  - CMakeCCompilerId.c, 84
  - CMakeCXXCompilerId.cpp, 87
- def, 9
  - DEFAULT\_AIR\_DENSITY, 9
  - FRICTION\_EPS, 10
  - GENTLY\_PUSH, 10
  - GRAVITY\_CONST, 10
- DEFAULT\_AIR\_DENSITY
  - def, 9
- direction
  - Rotor, 61
- DoubleSetpoint
  - controllers::DoubleSetpoint, 24
- Drive, 26
  - axis, 26
  - hinges, 26
  - noOfHinges, 27
  - position, 27
- DropTest, 27
  - collectSample, 27
  - getParsedState, 27
  - recvState, 28
  - sendControlMessage, 28
  - SetUp, 28
  - TearDown, 28
- eAR
  - AeroCoefficients, 12
- ekf
  - UAVparams, 78
- EKFScalers, 28
  - baroScaler, 29
  - predictScaler, 29
  - updateScaler, 29
  - zScaler, 29
- Euler
  - ODE, 42
- exiting
  - status.hpp, 109
- factory
  - ODE, 43
- figure
  - controller\_plots.m, 94
- findIndex
  - State, 70
- folderPath
  - controller\_plots.m, 94
- forceCoff
  - Rotor, 62
- FRICTION\_EPS
  - def, 10
- fromString
  - ODE, 43
- GENTLY\_PUSH
  - def, 10
- getCoefficients

- ControlSurfaces, 23
- getForce
  - ObjParams, 39
- getLastThrust
  - Jet, 32
- getMass
  - Load, 35
- getMethodsToTest
  - controller\_test.cpp, 98
  - ode\_test.cpp, 104
- getMicrosteps
  - ODE, 44
- getNoObj
  - State, 70
- getNoOfSurface
  - ControlSurfaces, 23
- getOffset
  - Load, 35
- getParams
  - State, 70
- getParsedState
  - DropTest, 27
- getPos
  - State, 71
- getRot
  - Hinge, 30
- getRotorHoverSpeeds
  - UAVparams, 77
- getRotorMaxSpeeds
  - UAVparams, 77
- getRotorTimeConstants
  - UAVparams, 77
- getSingleton
  - Params, 54
  - UAVparams, 77
- getState
  - State, 71
- getThrust
  - Jet, 32
- getV0
  - Ammo, 14
- getValues
  - ControlSurfaces, 23
- getVel
  - State, 71
- getWind
  - ObjParams, 40
- go
  - TimedLoop, 75
- GRAVITY\_CONST
  - def, 10
- Heun
  - ODE, 42
- HEX
  - CMakeCCompilerId.c, 84
  - CMakeCXXCompilerId.cpp, 87
- Hinge, 29
  - getRot, 30
  - Hinge, 30
    - operator=, 31
    - updateValue, 31
- hinge.cpp
  - asSkewMatrix, 96
- hinges
  - Drive, 26
- hoverSpeed
  - Rotor, 62
- i
  - controller\_plots.m, 94
- id
  - ObjParams, 41
  - Projectile, 60
- idle
  - status.hpp, 109
- info\_arch
  - CMakeCCompilerId.c, 85
  - CMakeCXXCompilerId.cpp, 88
- info\_compiler
  - CMakeCCompilerId.c, 85
  - CMakeCXXCompilerId.cpp, 89
- info\_language\_extensions\_default
  - CMakeCCompilerId.c, 85
  - CMakeCXXCompilerId.cpp, 89
- info\_language\_standard\_default
  - CMakeCCompilerId.c, 86
  - CMakeCXXCompilerId.cpp, 89
- info\_platform
  - CMakeCCompilerId.c, 86
  - CMakeCXXCompilerId.cpp, 89
- initialMode
  - UAVparams, 78
- initialOrientation
  - UAVparams, 79
- initialPosition
  - UAVparams, 79
- initialVelocity
  - UAVparams, 79
- INstantiate\_TEST\_SUITE\_P
  - controller\_test.cpp, 99
  - ode\_test.cpp, 104
- instantRun
  - UAVparams, 79
- integration\_test.cpp
  - communicationFolder, 117
  - ctx, 117
  - main, 115
  - programPath, 117
  - TEST, 115
  - TEST\_F, 115, 116
- isNormal
  - simulation.cpp, 112
- lx
  - UAVparams, 79
- lxy
  - UAVparams, 79
- lxz

- UAVparams, 79
- ly
  - UAVparams, 79
- lyz
  - UAVparams, 80
- lz
  - UAVparams, 80
- Jet, 31
  - getLastThrust, 32
  - getThrust, 32
  - phases, 33
  - start, 33
  - thrust, 33
  - time, 33
- jets
  - UAVparams, 80
- legend
  - controller\_plots.m, 92
- lib/UAV\_common/header/common.hpp, 91
- lib/UAV\_common/scripts/controller\_plots.m, 92
- lib/UAV\_common/src/components/aero\_coefficients.hpp, 95
- lib/UAV\_common/src/components/components.hpp, 95
- lib/UAV\_common/src/components/control\_surfaces.cpp, 95
- lib/UAV\_common/src/components/control\_surfaces.hpp, 95
- lib/UAV\_common/src/components/drive.cpp, 96
- lib/UAV\_common/src/components/drive.hpp, 96
- lib/UAV\_common/src/components/hinge.cpp, 96
- lib/UAV\_common/src/components/hinge.hpp, 96
- lib/UAV\_common/src/components/loads.cpp, 97
- lib/UAV\_common/src/components/loads.hpp, 97
- lib/UAV\_common/src/components/navi.hpp, 97
- lib/UAV\_common/src/controllers/controller.cpp, 97
- lib/UAV\_common/src/controllers/controller.hpp, 98
- lib/UAV\_common/src/controllers/controller\_test.cpp, 98
- lib/UAV\_common/src/controllers/impl/bang\_bang.cpp, 100
- lib/UAV\_common/src/controllers/impl/bang\_bang.hpp, 100
- lib/UAV\_common/src/controllers/impl/double\_setpoint.cpp, 100
- lib/UAV\_common/src/controllers/impl/double\_setpoint.hpp, 100
- lib/UAV\_common/src/controllers/impl/PID.cpp, 101
- lib/UAV\_common/src/controllers/impl/PID.hpp, 101
- lib/UAV\_common/src/controllers/impl/PIFF.cpp, 101
- lib/UAV\_common/src/controllers/impl/PIFF.hpp, 101
- lib/UAV\_common/src/logger/logger.cpp, 102
- lib/UAV\_common/src/logger/logger.hpp, 102
- lib/UAV\_common/src/ode/ode.cpp, 103
- lib/UAV\_common/src/ode/ode.hpp, 103
- lib/UAV\_common/src/ode/ode\_impl.hpp, 103
- lib/UAV\_common/src/ode/ode\_test.cpp, 104
- lib/UAV\_common/src/parser/parser.cpp, 106
- lib/UAV\_common/src/parser/parser.hpp, 107
- lib/UAV\_common/src/parser/uav\_params.cpp, 108
- lib/UAV\_common/src/parser/uav\_params.hpp, 109
- lib/UAV\_common/src/timed\_loop/status.hpp, 109
- lib/UAV\_common/src/timed\_loop/timed\_loop.cpp, 110
- lib/UAV\_common/src/timed\_loop/timed\_loop.hpp, 110
- Load, 34
  - getMass, 35
  - getOffset, 35
  - Load, 34
  - operator=, 35
  - release, 35
- loadConfig
  - UAVparams, 77
- log
  - Logger, 37
- Logger, 36
  - ~Logger, 37
  - log, 37
  - Logger, 36
  - setFmt, 38
  - setLogDirectory, 38
- logger.cpp
  - shouldLog, 102
- logger.hpp
  - LOGGER\_MASK, 103
- LOGGER\_MASK
  - logger.hpp, 103
- m
  - UAVparams, 80
- main
  - CMakeCCompilerId.c, 85
  - CMakeCXXCompilerId.cpp, 88
  - controller\_test.cpp, 99
  - integration\_test.cpp, 115
  - main.cpp, 111
  - ode\_test.cpp, 104
- main.cpp
  - main, 111
  - parseArgs, 111
- mass
  - ObjParams, 41
- maxSpeed
  - Rotor, 62
- name
  - SensorParams, 63
  - UAVparams, 80
- NONE
  - controllers::PID, 56
  - controllers::PIFF, 58
  - ODE, 42
- noOfAmmo
  - UAVparams, 80
- noOfCargo
  - UAVparams, 80
- noOfHinges
  - Drive, 27
- noOfJets

- UAVparams, 80
- noOfRotors
  - UAVparams, 81
- ObjParams, 38
  - CS\_coff, 41
  - getForce, 39
  - getWind, 40
  - id, 41
  - mass, 41
  - ObjParams, 39
  - setForce, 40
  - setWind, 40
- ODE, 41
  - ~ODE, 43
  - Euler, 42
  - factory, 43
  - fromString, 43
  - getMicrosteps, 44
  - Heun, 42
  - NONE, 42
  - ODE, 43
  - ODEMethod, 42
  - PC2, 42
  - PC4, 42
  - RK4, 42
  - step, 44
- ODE\_Euler, 45
  - ODE\_Euler, 45
  - step, 46
- ODE\_Heun, 46
  - ODE\_Heun, 47
  - step, 47
- ODE\_METHOD
  - Params, 54
- ODE\_PC2, 48
  - ODE\_PC2, 48
  - step, 48
- ODE\_PC4, 49
  - ODE\_PC4, 49
  - step, 50
- ODE\_RK4, 50
  - ODE\_RK4, 51
  - step, 51
- ode\_test.cpp
  - getMethodsToTest, 104
  - INstantiate\_TEST\_SUITE\_P, 104
  - main, 104
  - TEST\_F, 104, 105
  - TEST\_P, 105
- ODEMethod
  - ODE, 42
- ODETest, 52
  - SetUp, 52
  - TearDown, 52
- off
  - controller\_plots.m, 94
- on
  - controller\_plots.m, 94
- operator=
  - Ammo, 15
  - Hinge, 31
  - Load, 35
  - Params, 54
- Params, 52
  - ~Params, 54
  - getSingleton, 54
  - ODE\_METHOD, 54
  - operator=, 54
  - Params, 53
  - STEP\_TIME, 54
- parseArgs
  - main.cpp, 111
- parseHinge
  - uav\_params.cpp, 108
- parseInput
  - projectile\_parser.hpp, 117
- parseMatrixXd
  - parser.cpp, 106
  - parser.hpp, 107
- parser.cpp
  - parseMatrixXd, 106
  - parseVectorXd, 106
- parser.hpp
  - parseMatrixXd, 107
  - parseVectorXd, 108
- parseVectorXd
  - parser.cpp, 106
  - parser.hpp, 108
- PC2
  - ODE, 42
- PC4
  - ODE, 42
- phases
  - Jet, 33
- PID
  - controllers::PID, 56
- PIFF
  - controllers::PIFF, 58, 59
- PLATFORM\_ID
  - CMakeCCompilerId.c, 84
  - CMakeCXXCompilerId.cpp, 88
- plot
  - controller\_plots.m, 92
  - controller\_test.cpp, 99
- plot\_directory\_name
  - controller\_test.cpp, 99
- position
  - Drive, 27
  - Projectile, 61
- predictScaler
  - EKFScalers, 29
- programPath
  - integration\_test.cpp, 117
- Projectile, 60
  - id, 60
  - position, 61

- velocity, [61](#)
- projectile\_parser.hpp
  - checkInputErrors, [117](#)
  - parseInput, [117](#)
- Q
  - AHRSPParams, [13](#)
- R
  - AHRSPParams, [13](#)
- real\_time
  - State, [73](#)
- recvState
  - DropTest, [28](#)
- refreshTime
  - SensorParams, [63](#)
- release
  - Load, [35](#)
- reload
  - status.hpp, [109](#)
- removeObj
  - Simulation, [67](#)
  - State, [72](#)
- restoreTrim
  - ControlSurfaces, [23](#)
- RK4
  - ODE, [42](#)
- Rotor, [61](#)
  - direction, [61](#)
  - forceCoff, [62](#)
  - hoverSpeed, [62](#)
  - maxSpeed, [62](#)
  - timeConstant, [62](#)
  - torqueCoff, [62](#)
- rotorMixer
  - UAVparams, [81](#)
- rotors
  - UAVparams, [81](#)
- run
  - Simulation, [67](#)
- running
  - status.hpp, [109](#)
- S
  - AeroCoefficients, [12](#)
- sd
  - SensorParams, [63](#)
- sendControlMessage
  - DropTest, [28](#)
- SensorParams, [62](#)
  - bias, [63](#)
  - name, [63](#)
  - refreshTime, [63](#)
  - sd, [63](#)
- sensors
  - UAVparams, [81](#)
- set\_dt
  - Controller, [21](#)
- setFmt
  - Logger, [38](#)
- setForce
  - ObjParams, [40](#)
- setLogDirectory
  - Logger, [38](#)
- SetUp
  - ControllerTest, [21](#)
  - DropTest, [28](#)
  - ODETest, [52](#)
- setValues
  - ControlSurfaces, [23](#)
- setVel
  - State, [72](#)
- setWind
  - ObjParams, [40](#)
- shouldLog
  - logger.cpp, [102](#)
- Simulation, [63](#)
  - ~Simulation, [64](#)
  - addCommand, [65](#)
  - addObj, [65](#)
  - calcImpulseForce, [65](#)
  - removeObj, [67](#)
  - run, [67](#)
  - Simulation, [64](#)
  - solidSurfCollision, [67](#)
  - updateForce, [67](#)
  - updateWind, [68](#)
- simulation.cpp
  - isNormal, [112](#)
- solidSurfCollision
  - Simulation, [67](#)
- src/defines.hpp, [110](#)
- src/main.cpp, [111](#)
- src/params.cpp, [111](#)
- src/params.hpp, [112](#)
- src/simulation.cpp, [112](#)
- src/simulation.hpp, [113](#)
- src/state.cpp, [113](#)
- src/state.hpp, [113](#)
- stallLimit
  - AeroCoefficients, [12](#)
- start
  - Jet, [33](#)
- State, [68](#)
  - addObj, [69](#)
  - findIndex, [70](#)
  - getNoObj, [70](#)
  - getParams, [70](#)
  - getPos, [71](#)
  - getState, [71](#)
  - getVel, [71](#)
  - real\_time, [73](#)
  - removeObj, [72](#)
  - setVel, [72](#)
  - State, [69](#)
  - stateMutex, [74](#)
  - status, [74](#)

- to\_string, 72
  - updateForce, 72
  - updateState, 73
  - updateWind, 73
- stateMutex
  - State, 74
- Status
  - status.hpp, 109
- status
  - State, 74
- status.hpp
  - exiting, 109
  - idle, 109
  - reload, 109
  - running, 109
  - Status, 109
- step
  - ODE, 44
  - ODE\_Euler, 46
  - ODE\_Heun, 47
  - ODE\_PC2, 48
  - ODE\_PC4, 50
  - ODE\_RK4, 51
- STEP\_TIME
  - Params, 54
- STRINGIFY
  - CMakeCCompilerId.c, 85
  - CMakeCXXCompilerId.cpp, 88
- STRINGIFY\_HELPER
  - CMakeCCompilerId.c, 85
  - CMakeCXXCompilerId.cpp, 88
- surfaceMixer
  - UAVparams, 81
- surfaces
  - UAVparams, 81
- target
  - UAVparams, 81
- TearDown
  - ControllerTest, 22
  - DropTest, 28
  - ODETest, 52
- TEST
  - integration\_test.cpp, 115
- TEST\_F
  - integration\_test.cpp, 115, 116
  - ode\_test.cpp, 104, 105
- TEST\_P
  - controller\_test.cpp, 99
  - ode\_test.cpp, 105
- tests/integration\_test.cpp, 114
- tests/projectile\_parser.hpp, 117
- thrust
  - Jet, 33
- time
  - Jet, 33
- timeConstant
  - Rotor, 62
- TimedLoop, 74
  - go, 75
  - TimedLoop, 74
- title
  - controller\_plots.m, 93
- to\_string
  - State, 72
- torqueCoff
  - Rotor, 62
- type
  - AHRSPParams, 13
- uav\_params.cpp
  - parseHinge, 108
- UAVparams, 75
  - ~UAVparams, 77
  - aero\_coffs, 78
  - ahrs, 78
  - ammo, 78
  - cargo, 78
  - controllers, 78
  - ekf, 78
  - getRotorHoverSpeeds, 77
  - getRotorMaxSpeeds, 77
  - getRotorTimeContants, 77
  - getSingleton, 77
  - initialMode, 78
  - initialOrientation, 79
  - initialPosition, 79
  - initialVelocity, 79
  - instantRun, 79
  - lx, 79
  - lxy, 79
  - lxz, 79
  - ly, 79
  - lyz, 80
  - lz, 80
  - jets, 80
  - loadConfig, 77
  - m, 80
  - name, 80
  - noOfAmmo, 80
  - noOfCargo, 80
  - noOfJets, 80
  - noOfRotors, 81
  - rotorMixer, 81
  - rotors, 81
  - sensors, 81
  - surfaceMixer, 81
  - surfaces, 81
  - target, 81
  - UAVparams, 77
- updateForce
  - Simulation, 67
  - State, 72
- updateScaler
  - EKFScalers, 29
- updateState
  - State, 73
- updateValue

---

- Hinge, [31](#)
- updateWind
  - Simulation, [68](#)
  - State, [73](#)
- velocity
  - Projectile, [61](#)
- x
  - controller\_plots.m, [94](#)
- xlabel
  - controller\_plots.m, [93](#)
- y
  - controller\_plots.m, [94](#)
- ylabel
  - controller\_plots.m, [93](#)
- zScaler
  - EKFScalers, [29](#)