

PyReMoto

Generated by Doxygen 1.8.11

Contents

1	ReMoto in Python	1
2	projectPR	3
3	Namespace Index	5
3.1	Packages	5
4	Hierarchical Index	7
4.1	Class Hierarchy	7
5	Class Index	9
5.1	Class List	9
6	File Index	11
6.1	File List	11
7	Namespace Documentation	13
7.1	AxonDelay Namespace Reference	13
7.2	ChannelConductance Namespace Reference	13
7.3	Compartment Namespace Reference	13
7.3.1	Function Documentation	14
7.3.1.1	calcGLEak(area, specificRes)	14
7.4	Configuration Namespace Reference	14
7.5	Interneuron Namespace Reference	14
7.5.1	Function Documentation	15
7.5.1.1	runge_kutta(derivativeFunction, t, x, timeStep, timeStepByTwo, timeStepBySix)	15

7.6	InterneuronPool Namespace Reference	15
7.7	jointAnkleForceTask Namespace Reference	16
7.8	jointAnklePositionTask Namespace Reference	16
7.9	MotorUnit Namespace Reference	16
7.9.1	Function Documentation	16
7.9.1.1	calcGCoupling(cytR, lComp1, lComp2, dComp1, dComp2)	16
7.9.1.2	compGCouplingMatrix(gc)	17
7.9.1.3	runge_kutta(derivativeFunction, t, x, timeStep, timeStepByTwo, timeStepBySix)	17
7.10	MotorUnitPool Namespace Reference	18
7.11	MuscleHill Namespace Reference	18
7.12	MuscleNoHill Namespace Reference	18
7.13	MuscularActivation Namespace Reference	18
7.13.1	Function Documentation	19
7.13.1.1	twitchSaturation(activationsat, b)	19
7.14	NeuralTract Namespace Reference	19
7.15	NeuralTractUnit Namespace Reference	19
7.16	PointProcessGenerator Namespace Reference	20
7.16.1	Function Documentation	20
7.16.1.1	gammaPoint(GammaOrder, GammaOrderInv)	20
7.17	PulseConductanceState Namespace Reference	21
7.17.1	Function Documentation	21
7.17.1.1	compValOff(v0, alpha, beta, t, t0)	21
7.17.1.2	compValOn(v0, alpha, beta, t, t0)	21
7.18	simulation Namespace Reference	22
7.18.1	Function Documentation	22
7.18.1.1	simulator()	22
7.19	Synapse Namespace Reference	22
7.19.1	Function Documentation	23
7.19.1.1	compDynamicGmax(t, gmax, lastPulse, tau, dynamicGmax, var)	23
7.19.1.2	compRiStart(ri, t, ti, tPeak, tauOff)	23
7.19.1.3	compRiStop(rInf, ri, expFinish)	23
7.19.1.4	compRoff(Roff, t0, t, tauOff)	24
7.19.1.5	compRoffStart(Roff, ri, synContrib)	24
7.19.1.6	compRoffStop(Roff, ri, synContrib)	25
7.19.1.7	compRon(Non, rInf, Ron, t0, t, tauOn)	25
7.19.1.8	compRonStart(Ron, ri, synContrib)	26
7.19.1.9	compRonStop(Ron, ri, synContrib)	26
7.19.1.10	compSynapCond(Gmax, Ron, Roff)	27
7.20	SynapsesFactory Namespace Reference	27
7.21	SynapticNoise Namespace Reference	27

8 Class Documentation	29
8.1 AxonDelay.AxonDelay Class Reference	29
8.1.1 Detailed Description	30
8.1.2 Constructor & Destructor Documentation	30
8.1.2.1 <code>__init__(self, conf, nerve, pool, index)</code>	30
8.1.3 Member Function Documentation	30
8.1.3.1 <code>addSpinalSpike(self, t)</code>	30
8.1.3.2 <code>addTerminalSpike(self, t)</code>	31
8.1.4 Member Data Documentation	31
8.1.4.1 <code>index</code>	31
8.1.4.2 <code>latencySpinalTerminal_ms</code>	31
8.1.4.3 <code>latencyStimulusSpinal_ms</code>	31
8.1.4.4 <code>latencyStimulusTerminal_ms</code>	31
8.1.4.5 <code>length_m</code>	32
8.1.4.6 <code>stimulusPositiontoTerminal</code>	32
8.1.4.7 <code>terminalSpikeTrain</code>	32
8.1.4.8 <code>velocity_m_s</code>	32
8.2 ChannelConductance.ChannelConductance Class Reference	32
8.2.1 Detailed Description	33
8.2.2 Constructor & Destructor Documentation	33
8.2.2.1 <code>__init__(self, kind, conf, compArea, pool, neuronKind, index)</code>	33
8.2.3 Member Function Documentation	34
8.2.3.1 <code>compCondKf(self, V_mV)</code>	34
8.2.3.2 <code>compCondKs(self, V_mV)</code>	34
8.2.3.3 <code>compCondNa(self, V_mV)</code>	35
8.2.3.4 <code>computeCurrent(self, t, V_mV)</code>	35
8.2.4 Member Data Documentation	35
8.2.4.1 <code>compCond</code>	35
8.2.4.2 <code>condState</code>	35
8.2.4.3 <code>EqPot_mV</code>	36

8.2.4.4	<code>gmax_muS</code>	36
8.2.4.5	<code>kind</code>	36
8.2.4.6	<code>lenStates</code>	36
8.2.4.7	<code>stateType</code>	36
8.3	Compartment.Compartment Class Reference	36
8.3.1	Detailed Description	37
8.3.2	Constructor & Destructor Documentation	37
8.3.2.1	<code>__init__(self, kind, conf, pool, index, neuronKind)</code>	37
8.3.3	Member Function Documentation	38
8.3.3.1	<code>computeCurrent(self, t, V_mV)</code>	38
8.3.4	Member Data Documentation	38
8.3.4.1	<code>capacitance_nF</code>	38
8.3.4.2	<code>Channels</code>	38
8.3.4.3	<code>diameter_mum</code>	38
8.3.4.4	<code>gLeak</code>	38
8.3.4.5	<code>index</code>	38
8.3.4.6	<code>kind</code>	39
8.3.4.7	<code>length_mum</code>	39
8.3.4.8	<code>neuronKind</code>	39
8.3.4.9	<code>numberChannels</code>	39
8.3.4.10	<code>SynapsesIn</code>	39
8.3.4.11	<code>SynapsesOut</code>	39
8.4	Configuration.Configuration Class Reference	39
8.4.1	Detailed Description	40
8.4.2	Constructor & Destructor Documentation	40
8.4.2.1	<code>__init__(self, filename)</code>	40
8.4.3	Member Function Documentation	41
8.4.3.1	<code>determineSynapses(self, neuralSource)</code>	41
8.4.3.2	<code>inputFunctionGet(self, function)</code>	41
8.4.3.3	<code>parameterSet(self, paramTag, pool, index)</code>	41

8.4.4	Member Data Documentation	42
8.4.4.1	confArray	42
8.4.4.2	simDuration_ms	42
8.4.4.3	timeStep_ms	42
8.4.4.4	timeStepBySix_ms	42
8.4.4.5	timeStepByTwo_ms	42
8.5	Interneuron.Interneuron Class Reference	42
8.5.1	Detailed Description	44
8.5.2	Constructor & Destructor Documentation	44
8.5.2.1	__init__(self, conf, pool, index)	44
8.5.3	Member Function Documentation	44
8.5.3.1	addSomaSpike(self, t)	44
8.5.3.2	atualizeCompartments(self, t)	45
8.5.3.3	atualizeInterneuron(self, t)	45
8.5.3.4	dVdt(self, t, V)	46
8.5.3.5	transmitSpikes(self, t)	46
8.5.4	Member Data Documentation	46
8.5.4.1	capacitanceInv	46
8.5.4.2	compartment	47
8.5.4.3	compNumber	47
8.5.4.4	conf	47
8.5.4.5	G	47
8.5.4.6	iInjected	47
8.5.4.7	ilonic	47
8.5.4.8	index	47
8.5.4.9	indicesOfSynapsesOnTarget	48
8.5.4.10	kind	48
8.5.4.11	pool	48
8.5.4.12	position_mm	48
8.5.4.13	RefPer_ms	48

8.5.4.14	somaIndex	48
8.5.4.15	somaSpikeTrain	48
8.5.4.16	SynapsesOut	48
8.5.4.17	terminalSpikeTrain	49
8.5.4.18	threshold_mV	49
8.5.4.19	transmitSpikesThroughSynapses	49
8.5.4.20	tSomaSpike	49
8.5.4.21	v_mV	49
8.6	InterneuronPool.InterneuronPool Class Reference	49
8.6.1	Detailed Description	50
8.6.2	Constructor & Destructor Documentation	50
8.6.2.1	__init__(self, conf, pool)	50
8.6.3	Member Function Documentation	50
8.6.3.1	atualizeInterneuronPool(self, t)	50
8.6.3.2	listSpikes(self)	51
8.6.4	Member Data Documentation	51
8.6.4.1	conf	51
8.6.4.2	kind	51
8.6.4.3	Nnumber	51
8.6.4.4	pool	51
8.6.4.5	poolSomaSpikes	51
8.6.4.6	unit	51
8.7	jointAnkleForceTask.jointAnkleForceTask Class Reference	52
8.7.1	Detailed Description	52
8.7.2	Constructor & Destructor Documentation	52
8.7.2.1	__init__(self, conf, pools)	52
8.7.3	Member Function Documentation	52
8.7.3.1	atualizeAngle(self, t, ankleAngle)	52
8.7.3.2	atualizeAnkle(self, t, ankleAngle)	53
8.7.4	Member Data Documentation	53

8.7.4.1	ankleAngle_rad	53
8.7.4.2	conf	53
8.7.4.3	muscles	53
8.8	jointAnklePositionTask.jointAnklePositionTask Class Reference	53
8.8.1	Detailed Description	55
8.8.2	Constructor & Destructor Documentation	55
8.8.2.1	__init__(self, conf, pool, MUnumber, MUpelnumber, musculotendonLength, unit)	55
8.8.3	Member Function Documentation	55
8.8.3.1	atualizeActivation(self, activation_Sat)	55
8.8.3.2	atualizeForce(self, activation_Sat, musculoTendonLength)	56
8.8.3.3	atualizeLenghtsAndVelocity(self)	56
8.8.3.4	atualizeMuscleForce(self)	57
8.8.3.5	atualizeTendonForce(self)	57
8.8.3.6	computeAcceleration(self)	58
8.8.3.7	computeElasticElementForce(self)	58
8.8.3.8	computeForceLengthTypeI(self)	58
8.8.3.9	computeForceLengthTypeII(self)	58
8.8.3.10	computeForceVelocityTypeI(self)	59
8.8.3.11	computeForceVelocityTypeII(self)	59
8.8.3.12	computePennationAngle(self)	59
8.8.3.13	computeTypeIActiveForce(self)	60
8.8.3.14	computeTypeIIActiveForce(self)	60
8.8.3.15	computeViscousElementForce(self)	61
8.8.3.16	dLdt(self)	61
8.8.4	Member Data Documentation	61
8.8.4.1	a0_TypeI	61
8.8.4.2	a0_TypeII	61
8.8.4.3	a1_TypeI	62
8.8.4.4	a1_TypeII	62
8.8.4.5	a2_TypeI	62

8.8.4.6	a2_TypeII	62
8.8.4.7	activationTypeI	62
8.8.4.8	activationTypeII	62
8.8.4.9	b_TypeI	62
8.8.4.10	b_TypeII	62
8.8.4.11	c0_TypeI	62
8.8.4.12	c0_TypeII	62
8.8.4.13	c1_TypeI	63
8.8.4.14	c1_TypeII	63
8.8.4.15	conf	63
8.8.4.16	contractileForce_N	63
8.8.4.17	d_TypeI	63
8.8.4.18	d_TypeII	63
8.8.4.19	elasticForce_N	63
8.8.4.20	elasticity	63
8.8.4.21	force	63
8.8.4.22	forceNorm	63
8.8.4.23	length_m	64
8.8.4.24	lengthNorm	64
8.8.4.25	mass	64
8.8.4.26	maximumActivationForce	64
8.8.4.27	maximumForce_N	64
8.8.4.28	MUnumber	64
8.8.4.29	MUtypeInumber	64
8.8.4.30	optimalLength_m	64
8.8.4.31	optimalTendonLength	64
8.8.4.32	p_TypeI	65
8.8.4.33	p_TypeII	65
8.8.4.34	pennationAngle_rad	65
8.8.4.35	pennationAngleAtOptimalLengthSin	65

8.8.4.36	pool	65
8.8.4.37	strain	65
8.8.4.38	tendonCurvatureConstant	65
8.8.4.39	tendonElasticity	65
8.8.4.40	tendonForce_N	65
8.8.4.41	tendonForceNorm	65
8.8.4.42	tendonLength_m	66
8.8.4.43	tendonLengthNorm	66
8.8.4.44	tendonLinearOnsetLength	66
8.8.4.45	timeIndex	66
8.8.4.46	twitchAmp_N	66
8.8.4.47	twTet	66
8.8.4.48	velocity_m_ms	66
8.8.4.49	velocityNorm	66
8.8.4.50	viscosity	66
8.8.4.51	viscousForce_N	67
8.8.4.52	Vmax_TypeI	67
8.8.4.53	Vmax_TypeII	67
8.8.4.54	w_TypeI	67
8.8.4.55	w_TypeII	67
8.9	MotorUnit.MotorUnit Class Reference	67
8.9.1	Detailed Description	69
8.9.2	Constructor & Destructor Documentation	69
8.9.2.1	__init__(self, conf, pool, index, kind)	69
8.9.3	Member Function Documentation	69
8.9.3.1	addSomaSpike(self, t)	69
8.9.3.2	atualizeCompartments(self, t)	70
8.9.3.3	atualizeDelay(self, t)	71
8.9.3.4	atualizeMotorUnit(self, t)	71
8.9.3.5	dVdt(self, t, V)	72

8.9.3.6	transmitSpikes(self, t)	72
8.9.4	Member Data Documentation	72
8.9.4.1	bSat	72
8.9.4.2	capacitanceInv	73
8.9.4.3	compartment	73
8.9.4.4	compNumber	73
8.9.4.5	conf	73
8.9.4.6	Delay	73
8.9.4.7	G	73
8.9.4.8	iInjected	73
8.9.4.9	iIonic	74
8.9.4.10	index	74
8.9.4.11	indicesOfSynapsesOnTarget	74
8.9.4.12	kind	74
8.9.4.13	MNRefPer_ms	74
8.9.4.14	nerve	74
8.9.4.15	position_mm	74
8.9.4.16	somaIndex	75
8.9.4.17	somaSpikeTrain	75
8.9.4.18	SynapsesOut	75
8.9.4.19	terminalSpikeTrain	75
8.9.4.20	threshold_mV	75
8.9.4.21	transmitSpikesThroughSynapses	75
8.9.4.22	tSomaSpike	75
8.9.4.23	TwitchAmp_N	75
8.9.4.24	TwitchTc_ms	76
8.9.4.25	twTet	76
8.9.4.26	v_mV	76
8.10	MotorUnitPool.MotorUnitPool Class Reference	76
8.10.1	Detailed Description	77

8.10.2	Constructor & Destructor Documentation	77
8.10.2.1	__init__(self, conf, pool)	77
8.10.3	Member Function Documentation	77
8.10.3.1	atualizeMotorUnitPool(self, t)	77
8.10.3.2	listSpikes(self)	77
8.10.4	Member Data Documentation	77
8.10.4.1	Activation	77
8.10.4.2	conf	78
8.10.4.3	hillModel	78
8.10.4.4	kind	78
8.10.4.5	MUnumber	78
8.10.4.6	Muscle	78
8.10.4.7	pool	78
8.10.4.8	poolSomaSpikes	78
8.10.4.9	poolTerminalSpikes	78
8.10.4.10	unit	79
8.11	MuscleHill.MuscleHill Class Reference	79
8.11.1	Detailed Description	81
8.11.2	Constructor & Destructor Documentation	81
8.11.2.1	__init__(self, conf, pool, MUnumber, MUtypeInumber, unit)	81
8.11.3	Member Function Documentation	81
8.11.3.1	atualizeActivation(self, activation_Sat)	81
8.11.3.2	atualizeForce(self, activation_Sat)	81
8.11.3.3	atualizeLenghtsAndVelocity(self)	82
8.11.3.4	atualizeMomentArm(self, ankleAngle)	82
8.11.3.5	atualizeMuscleForce(self)	83
8.11.3.6	atualizeMusculoTendonLength(self, ankleAngle)	83
8.11.3.7	atualizeTendonForce(self)	84
8.11.3.8	computeAcceleration(self)	84
8.11.3.9	computeElasticElementForce(self)	84

8.11.3.10 computeForceLengthTypeI(self)	84
8.11.3.11 computeForceLengthTypeII(self)	85
8.11.3.12 computeForceVelocityTypeI(self)	85
8.11.3.13 computeForceVelocityTypeII(self)	85
8.11.3.14 computePennationAngle(self)	85
8.11.3.15 computeTypeIActiveForce(self)	86
8.11.3.16 computeTypeIIActiveForce(self)	86
8.11.3.17 computeViscousElementForce(self)	87
8.11.3.18 dLdt(self)	87
8.11.4 Member Data Documentation	88
8.11.4.1 a0_TypeI	88
8.11.4.2 a0_TypeII	88
8.11.4.3 a1_TypeI	88
8.11.4.4 a1_TypeII	88
8.11.4.5 a2_TypeI	88
8.11.4.6 a2_TypeII	88
8.11.4.7 activationTypeI	88
8.11.4.8 activationTypeII	88
8.11.4.9 b_TypeI	88
8.11.4.10 b_TypeII	88
8.11.4.11 c0_TypeI	89
8.11.4.12 c0_TypeII	89
8.11.4.13 c1_TypeI	89
8.11.4.14 c1_TypeII	89
8.11.4.15 conf	89
8.11.4.16 contractileForce_N	89
8.11.4.17 d_TypeI	89
8.11.4.18 d_TypeII	89
8.11.4.19 elasticForce_N	89
8.11.4.20 elasticity	89

8.11.4.21 force	90
8.11.4.22 forceNorm	90
8.11.4.23 length_m	90
8.11.4.24 lengthNorm	90
8.11.4.25 m0	90
8.11.4.26 m1	90
8.11.4.27 m2	90
8.11.4.28 m3	90
8.11.4.29 m4	90
8.11.4.30 mass	90
8.11.4.31 maximumActivationForce	91
8.11.4.32 maximumForce_N	91
8.11.4.33 momentArm_m	91
8.11.4.34 MUnumber	91
8.11.4.35 musculoTendonLength_m	91
8.11.4.36 MUpelnumber	91
8.11.4.37 n0	91
8.11.4.38 n1	91
8.11.4.39 n2	91
8.11.4.40 n3	92
8.11.4.41 n4	92
8.11.4.42 optimalLength_m	92
8.11.4.43 optimalTendonLength	92
8.11.4.44 p_TypeI	92
8.11.4.45 p_TypeII	92
8.11.4.46 pennationAngle_rad	92
8.11.4.47 pennationAngleAtOptimalLengthSin	92
8.11.4.48 pool	92
8.11.4.49 strain	92
8.11.4.50 tendonCurvatureConstant	93

8.11.4.51 tendonElasticity	93
8.11.4.52 tendonForce_N	93
8.11.4.53 tendonForceNorm	93
8.11.4.54 tendonLength_m	93
8.11.4.55 tendonLengthNorm	93
8.11.4.56 tendonLinearOnsetLength	93
8.11.4.57 timeIndex	93
8.11.4.58 twitchAmp_N	93
8.11.4.59 twTet	93
8.11.4.60 velocity_m_ms	94
8.11.4.61 velocityNorm	94
8.11.4.62 viscosity	94
8.11.4.63 viscousForce_N	94
8.11.4.64 Vmax_TypeI	94
8.11.4.65 Vmax_TypeII	94
8.11.4.66 w_TypeI	94
8.11.4.67 w_TypeII	94
8.12 MuscleNoHill.MuscleNoHill Class Reference	94
8.12.1 Detailed Description	95
8.12.2 Constructor & Destructor Documentation	95
8.12.2.1 __init__(self, conf, pool, MUnumber, MUtypeName, unit)	95
8.12.3 Member Function Documentation	95
8.12.3.1 atualizeForce(self, activation_Sat)	95
8.12.4 Member Data Documentation	95
8.12.4.1 conf	95
8.12.4.2 force	95
8.12.4.3 maximumActivationForce	96
8.12.4.4 MUnumber	96
8.12.4.5 MUtypeName	96
8.12.4.6 pool	96

8.12.4.7	timeIndex	96
8.12.4.8	twitchAmp_N	96
8.12.4.9	twTet	96
8.13	MuscularActivation.MuscularActivation Class Reference	96
8.13.1	Detailed Description	97
8.13.2	Constructor & Destructor Documentation	97
8.13.2.1	__init__(self, conf, pool, MUnumber, unit)	97
8.13.3	Member Function Documentation	97
8.13.3.1	atualizeActivationSignal(self, t, unit)	97
8.13.4	Member Data Documentation	98
8.13.4.1	activation_nonSat	98
8.13.4.2	activation_Sat	98
8.13.4.3	activationModel	98
8.13.4.4	ActMatrix	98
8.13.4.5	an	99
8.13.4.6	bSat	99
8.13.4.7	conf	99
8.13.4.8	diracDeltaValue	99
8.13.4.9	MUnumber	99
8.13.4.10	pool	99
8.14	NeuralTract.NeuralTract Class Reference	99
8.14.1	Detailed Description	100
8.14.2	Constructor & Destructor Documentation	100
8.14.2.1	__init__(self, conf, pool)	100
8.14.3	Member Function Documentation	101
8.14.3.1	atualizePool(self, t)	101
8.14.3.2	listSpikes(self)	101
8.14.4	Member Data Documentation	101
8.14.4.1	FR	101
8.14.4.2	GammaOrder	101

8.14.4.3	kind	101
8.14.4.4	Number	101
8.14.4.5	pool	102
8.14.4.6	poolTerminalSpikes	102
8.14.4.7	target	102
8.14.4.8	timeIndex	102
8.14.4.9	unit	102
8.15	NeuralTractUnit.NeuralTractUnit Class Reference	102
8.15.1	Detailed Description	103
8.15.2	Constructor & Destructor Documentation	103
8.15.2.1	__init__(self, conf, pool, GammaOrder, index)	103
8.15.3	Member Function Documentation	104
8.15.3.1	atualizeNeuralTractUnit(self, t, FR)	104
8.15.3.2	transmitSpikes(self, t)	104
8.15.4	Member Data Documentation	105
8.15.4.1	GammaOrder	105
8.15.4.2	index	105
8.15.4.3	indicesOfSynapsesOnTarget	105
8.15.4.4	kind	105
8.15.4.5	spikesGenerator	105
8.15.4.6	SynapsesOut	105
8.15.4.7	terminalSpikeTrain	105
8.15.4.8	transmitSpikesThroughSynapses	105
8.16	object Class Reference	106
8.17	PointProcessGenerator.PointProcessGenerator Class Reference	106
8.17.1	Detailed Description	106
8.17.2	Constructor & Destructor Documentation	106
8.17.2.1	__init__(self, GammaOrder, index)	106
8.17.3	Member Function Documentation	107
8.17.3.1	atualizeGenerator(self, t, firingRate)	107

8.17.4 Member Data Documentation	107
8.17.4.1 GammaOrder	107
8.17.4.2 GammaOrderInv	107
8.17.4.3 index	107
8.17.4.4 points	107
8.17.4.5 threshold	108
8.17.4.6 y	108
8.18 PulseConductanceState.PulseConductanceState Class Reference	108
8.18.1 Detailed Description	109
8.18.2 Constructor & Destructor Documentation	109
8.18.2.1 __init__(self, kind, conf, pool, neuronKind, index)	109
8.18.3 Member Function Documentation	109
8.18.3.1 changeState(self, t)	109
8.18.3.2 computeStateValue(self, t)	110
8.18.4 Member Data Documentation	110
8.18.4.1 actType	110
8.18.4.2 alpha_ms1	110
8.18.4.3 beta_ms1	110
8.18.4.4 computeValueOff	110
8.18.4.5 computeValueOn	110
8.18.4.6 kind	110
8.18.4.7 PulseDur_ms	111
8.18.4.8 state	111
8.18.4.9 t0	111
8.18.4.10 v0	111
8.18.4.11 value	111
8.19 Synapse.Synapse Class Reference	111
8.19.1 Detailed Description	112
8.19.2 Constructor & Destructor Documentation	113
8.19.2.1 __init__(self, conf, pool, index, compartment, kind, neuronKind)	113

8.19.3 Member Data Documentation	113
8.19.3.1 alpha_ms1	113
8.19.3.2 beta_ms1	113
8.19.3.3 conductanceState	113
8.19.3.4 delay_ms	113
8.19.3.5 dynamicGmax	113
8.19.3.6 dynamics	113
8.19.3.7 EqPot_mV	114
8.19.3.8 expFinish	114
8.19.3.9 gmax_muS	114
8.19.3.10 gMaxTot_muS	114
8.19.3.11 kind	114
8.19.3.12 neuronKind	114
8.19.3.13 Non	114
8.19.3.14 numberOfIncomingSynapses	114
8.19.3.15 pool	115
8.19.3.16 ri	115
8.19.3.17 rInf	115
8.19.3.18 Roff	115
8.19.3.19 Ron	115
8.19.3.20 t0	115
8.19.3.21 tauOff	115
8.19.3.22 tauOn	116
8.19.3.23 tBeginOfPulse	116
8.19.3.24 tEndOfPulse	116
8.19.3.25 ti	116
8.19.3.26 timeConstant_ms	116
8.19.3.27 tLastPulse	116
8.19.3.28 Tmax_mM	116
8.19.3.29 tPeak_ms	116

8.19.3.30 variation	117
8.20 SynapsesFactory.SynapsesFactory Class Reference	117
8.20.1 Detailed Description	117
8.20.2 Constructor & Destructor Documentation	117
8.20.2.1 __init__(self, conf, pools)	117
8.20.3 Member Data Documentation	118
8.20.3.1 numberOfSynapses	118
8.21 SynapticNoise.SynapticNoise Class Reference	118
8.21.1 Detailed Description	119
8.21.2 Constructor & Destructor Documentation	119
8.21.2.1 __init__(self, conf, pool)	119
8.21.3 Member Function Documentation	119
8.21.3.1 atualizePool(self, t)	119
8.21.3.2 listSpikes(self)	119
8.21.4 Member Data Documentation	119
8.21.4.1 FR	119
8.21.4.2 GammaOrder	120
8.21.4.3 kind	120
8.21.4.4 Number	120
8.21.4.5 pool	120
8.21.4.6 poolTerminalSpikes	120
8.21.4.7 target	120
8.21.4.8 timeIndex	120
8.21.4.9 unit	120

9 File Documentation	121
9.1 AxonDelay.py File Reference	121
9.2 ChannelConductance.py File Reference	121
9.3 Compartment.py File Reference	121
9.4 Configuration.py File Reference	122
9.5 Interneuron.py File Reference	122
9.6 InterneuronPool.py File Reference	122
9.7 jointAnkleForceTask.py File Reference	123
9.8 jointAnklePositionTask.py File Reference	123
9.9 MotorUnit.py File Reference	123
9.10 MotorUnitPool.py File Reference	124
9.11 MuscleHill.py File Reference	124
9.12 MuscleNoHill.py File Reference	124
9.13 MuscularActivation.py File Reference	124
9.14 NeuralTract.py File Reference	125
9.15 NeuralTractUnit.py File Reference	125
9.16 PointProcessGenerator.py File Reference	125
9.17 PulseConductanceState.py File Reference	126
9.18 README.md File Reference	126
9.19 simulation.py File Reference	126
9.20 Synapse.py File Reference	126
9.21 SynapsesFactory.py File Reference	127
9.22 SynapticNoise.py File Reference	127

Chapter 1

ReMoto in Python

This program is a neuronal simulation system, intended for studying spinal cord neuronal networks responsible for muscle control. These networks are affected by descending drive, afferent drive, and electrical nerve stimulation. The simulator may be used to investigate phenomena at several levels of organization, e.g., at the neuronal membrane level or at the whole muscle behavior level (e.g., muscle force generation). This versatility is due to the fact that each element (neurons, synapses, muscle fibers) has its own specific mathematical model, usually involving the action of voltage- or neurotransmitter-dependent ionic channels. The simulator should be helpful in activities such as interpretation of results obtained from neurophysiological experiments in humans or mammals, proposal of hypothesis or testing models or theories on neuronal dynamics or neuronal network processing, validation of experimental protocols, and teaching neurophysiology.

The elements that take part in the system belong to the following classes: motoneurons, muscle fibers (electrical activity and force generation), Renshaw cells, Ia inhibitory interneurons, Ib inhibitory interneurons, Ia and Ib afferents. The neurons are interconnected by chemical synapses, which can exhibit depression or facilitation.

The system simulates the following nuclei involved in flexion and extension of the human or cat ankle: Medial Gastrocnemius (MG), Lateral Gastrocnemius (LG), Soleus (SOL), and Tibialis Anterior (TA).

A web-based version can be found in remoto.leb.usp.br. The version to which this documentation refers is from a Python program that can be found in github.com/rnwatanabe/projectPR.

Chapter 2

projectPR

Neuromuscular simulator in Python.

Under progress....

To run an example, execute the file [simulation.py](#).

Chapter 3

Namespace Index

3.1 Packages

Here are the packages with brief descriptions (if available):

AxonDelay	13
ChannelConductance	13
Compartment	13
Configuration	14
Interneuron	14
InterneuronPool	15
jointAnkleForceTask	16
jointAnklePositionTask	16
MotorUnit	16
MotorUnitPool	18
MuscleHill	18
MuscleNoHill	18
MuscularActivation	18
NeuralTract	19
NeuralTractUnit	19
PointProcessGenerator	20
PulseConductanceState	21
simulation	22
Synapse	22
SynapsesFactory	27
SynapticNoise	27

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

object	106
AxonDelay.AxonDelay	29
ChannelConductance.ChannelConductance	32
Compartment.Compartment	36
Configuration.Configuration	39
Interneuron.Interneuron	42
InterneuronPool.InterneuronPool	49
jointAnkleForceTask.jointAnkleForceTask	52
jointAnklePositionTask.jointAnklePositionTask	53
MotorUnit.MotorUnit	67
MotorUnitPool.MotorUnitPool	76
MuscleHill.MuscleHill	79
MuscleNoHill.MuscleNoHill	94
MuscularActivation.MuscularActivation	96
NeuralTract.NeuralTract	99
NeuralTractUnit.NeuralTractUnit	102
PointProcessGenerator.PointProcessGenerator	106
PulseConductanceState.PulseConductanceState	108
Synapse.Synapse	111
SynapsesFactory.SynapsesFactory	117
SynapticNoise.SynapticNoise	118

Chapter 5

Class Index

5.1 Class List

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

AxonDelay.AxonDelay	
Class that implements a delay correspondent to the nerve	29
ChannelConductance.ChannelConductance	
Class that implements a model of the ionic Channels in a compartment	32
Compartment.Compartment	
Class that implements a neural compartment	36
Configuration.Configuration	
Class that builds an object of Configuration , based on a configuration file	39
Interneuron.Interneuron	
Class that implements a motor unit model	42
InterneuronPool.InterneuronPool	
Class that implements a motor unit pool	49
jointAnkleForceTask.jointAnkleForceTask	52
jointAnklePositionTask.jointAnklePositionTask	53
MotorUnit.MotorUnit	
Class that implements a motor unit model	67
MotorUnitPool.MotorUnitPool	
Class that implements a motor unit pool	76
MuscleHill.MuscleHill	79
MuscleNoHill.MuscleNoHill	94
MuscularActivation.MuscularActivation	96
NeuralTract.NeuralTract	
Class that implements a a neural tract, composed by the descending commands from the motor cortex	99
NeuralTractUnit.NeuralTractUnit	
Class that implements a neural tract unit	102
object	106
PointProcessGenerator.PointProcessGenerator	
Generator of point processes	106
PulseConductanceState.PulseConductanceState	
Implements the Destexhe pulse approximation of the solution of the states of the Hodgkin-Huxley neuron model	108
Synapse.Synapse	
Implements the synapse model from Destexhe (1994) using the computational method from Lytton (1996)	111

[SynapsesFactory.SynapsesFactory](#)

Class to build all the synapses in the system 117

[SynapticNoise.SynapticNoise](#)

Class that implements a synaptic noise for a pool of neurons 118

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

AxonDelay.py	121
ChannelConductance.py	121
Compartment.py	121
Configuration.py	122
Interneuron.py	122
InterneuronPool.py	122
jointAnkleForceTask.py	123
jointAnklePositionTask.py	123
MotorUnit.py	123
MotorUnitPool.py	124
MuscleHill.py	124
MuscleNoHill.py	124
MuscularActivation.py	124
NeuralTract.py	125
NeuralTractUnit.py	125
PointProcessGenerator.py	125
PulseConductanceState.py	126
simulation.py	126
Synapse.py	126
SynapsesFactory.py	127
SynapticNoise.py	127

Chapter 7

Namespace Documentation

7.1 AxonDelay Namespace Reference

Classes

- class [AxonDelay](#)
Class that implements a delay correspondent to the nerve.

7.2 ChannelConductance Namespace Reference

Classes

- class [ChannelConductance](#)
Class that implements a model of the ionic Channels in a compartment.

7.3 Compartment Namespace Reference

Classes

- class [Compartment](#)
Class that implements a neural compartment.

Functions

- def [calcGLEak](#) (area, specificRes)
Computes the leak conductance of the compartment.

7.3.1 Function Documentation

7.3.1.1 `def Compartment.calcGLeak (area, specificRes)`

Computes the leak conductance of the compartment.

- Input:
 - **area**: area of the compartment in cm^2 .
 - **specificRes**: specific resistance of the compartment in $\Omega.\text{cm}^2$.
- Output:
 - Leak conductance in MS.

It is compute according to the following formula:

$$g = 10^6 \cdot \frac{A}{\rho} \quad (7.1)$$

where A is the compartment area [cm^2], ρ is the specific resistance [$\Omega.\text{cm}^2$] and g is the compartment conductance [MS].

Definition at line 32 of file `Compartment.py`.

7.4 Configuration Namespace Reference

Classes

- class [Configuration](#)
Class that builds an object of [Configuration](#), based on a configuration file.

7.5 Interneuron Namespace Reference

Classes

- class [Interneuron](#)
Class that implements a motor unit model.

Functions

- def [runge_kutta](#) (derivativeFunction, t, x, timeStep, timeStepByTwo, timeStepBySix)
Function to implement the fourth order Runge-Kutta Method to solve numerically a differential equation.

7.5.1 Function Documentation

7.5.1.1 `def Interneuron.runge_kutta (derivativeFunction, t, x, timeStep, timeStepByTwo, timeStepBySix)`

Function to implement the fourth order Runge-Kutta Method to solve numerically a differential equation.

- Inputs:
 - **derivativeFunction**: function that corresponds to the derivative of the differential equation.
 - **t**: current instant.
 - **x**: current state value.
 - **timeStep**: time step of the solution of the differential equation, in the same unit of t.
 - **timeStepByTwo**: timeStep divided by two, for computational efficiency.
 - **timeStepBySix**: timeStep divided by six, for computational efficiency.

This method is intended to solve the following differential equation:

$$\frac{dx(t)}{dt} = f(t, x(t)) \quad (7.2)$$

First, four derivatives are computed:

$$\begin{aligned} k_1 &= f(t, x(t)) \\ k_2 &= f\left(t + \frac{\Delta t}{2}, x(t) + \frac{\Delta t}{2} \cdot k_1\right) \\ k_3 &= f\left(t + \frac{\Delta t}{2}, x(t) + \frac{\Delta t}{2} \cdot k_2\right) \\ k_4 &= f(t + \Delta t, x(t) + \Delta t \cdot k_3) \end{aligned}$$

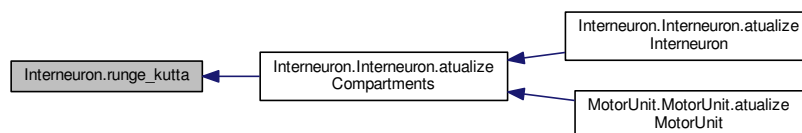
where Δt is the time step of the numerical solution of the differential equation.

Then the value of $x(t + \Delta t)$ is computed with:

$$x(t + \Delta t) = x(t) + \frac{\Delta t}{6} (k_1 + 2k_2 + 2k_3 + k_4) \quad (7.3)$$

Definition at line 51 of file Interneuron.py.

Here is the caller graph for this function:



7.6 InterneuronPool Namespace Reference

Classes

- class [InterneuronPool](#)
Class that implements a motor unit pool.

7.7 jointAnkleForceTask Namespace Reference

Classes

- class [jointAnkleForceTask](#)

7.8 jointAnklePositionTask Namespace Reference

Classes

- class [jointAnklePositionTask](#)

7.9 MotorUnit Namespace Reference

Classes

- class [MotorUnit](#)
Class that implements a motor unit model.

Functions

- def [calcGCoupling](#) (cytR, lComp1, lComp2, dComp1, dComp2)
Calculates the coupling conductance between two compartments.
- def [compGCouplingMatrix](#) (gc)
Computes the Coupling Matrix to be used in the dVdt function of the N compartments of the motor unit.
- def [runge_kutta](#) (derivativeFunction, t, x, timeStep, timeStepByTwo, timeStepBySix)
Function to implement the fourth order Runge-Kutta Method to solve numerically a differential equation.

7.9.1 Function Documentation

7.9.1.1 def MotorUnit.calcGCoupling (cytR, lComp1, lComp2, dComp1, dComp2)

Calculates the coupling conductance between two compartments.

- Inputs:
 - **cytR**: Cytoplasmatic resistivity in $\Omega\cdot\text{cm}$.
 - **lComp1, lComp2**: length of the compartments in μm .
 - **dComp1, dComp2**: diameter of the compartments in μm .
- Output:
 - coupling conductance in MS.

The coupling conductance between compartment 1 and 2 is computed by the following equation:

$$g_c = \frac{2 \cdot 10^2}{\frac{R_{cyt} l_1}{\pi r_1^2} + \frac{R_{cyt} l_2}{\pi r_2^2}} \quad (7.4)$$

where g_c is the coupling conductance [MS], R_{cyt} is the cytoplasmatic resistivity [$\Omega\cdot\text{cm}$], l_1 and l_2 are the lengths [μm] of compartments 1 and 2, respectively and r_1 and r_2 are the radius [μm] of compartments 1 and 2, respectively.

Definition at line 46 of file MotorUnit.py.

7.9.1.2 def MotorUnit.compGCCouplingMatrix (gc)

Computes the Coupling Matrix to be used in the dVdt function of the N compartments of the motor unit.

The Matrix uses the values obtained with the function calcGCCoupling.

- Inputs:
 - **gc**: the vector with N elements, with the coupling conductance of each compartment of the Motor Unit.
- Output:
 - the GC matrix

$$GC = \begin{bmatrix} -g_c[0] & g_c[0] & 0 & \dots & \dots & 0 & 0 & 0 \\ g_c[0] & -g_c[0] - g_c[1] & g_c[1] & 0 & \dots & \dots & 0 & 0 \\ \vdots & & \ddots & & & & & \\ 0 & \dots & g_c[i-1] & -g_c[i-1] - g_c[i] & g_c[i] & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & & \dots & & g_c[N-2] & -g_c[N-2] - g_c[N-1] & g_c[N-1] & 0 \\ 0 & \dots & 0 & & & 0 & g_c[N-1] & -g_c[N-1] \end{bmatrix} \quad (7.5)$$

Definition at line 78 of file MotorUnit.py.

7.9.1.3 def MotorUnit.runge_kutta (derivativeFunction, t, x, timeStep, timeStepByTwo, timeStepBySix)

Function to implement the fourth order Runge-Kutta Method to solve numerically a differential equation.

- Inputs:
 - **derivativeFunction**: function that corresponds to the derivative of the differential equation.
 - **t**: current instant.
 - **x**: current state value.
 - **timeStep**: time step of the solution of the differential equation, in the same unit of t.
 - **timeStepByTwo**: timeStep divided by two, for computational efficiency.
 - **timeStepBySix**: timeStep divided by six, for computational efficiency.

This method is intended to solve the following differential equation:

$$\frac{dx(t)}{dt} = f(t, x(t)) \quad (7.6)$$

First, four derivatives are computed:

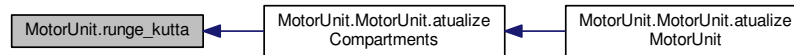
$$\begin{aligned} k_1 &= f(t, x(t)) \\ k_2 &= f\left(t + \frac{\Delta t}{2}, x(t) + \frac{\Delta t}{2} \cdot k_1\right) \\ k_3 &= f\left(t + \frac{\Delta t}{2}, x(t) + \frac{\Delta t}{2} \cdot k_2\right) \\ k_4 &= f(t + \Delta t, x(t) + \Delta t \cdot k_3) \end{aligned} \quad \text{where } \Delta t \text{ is the time step of the numerical solution of the differential equation.}$$

Then the value of $x(t + \Delta t)$ is computed with:

$$x(t + \Delta t) = x(t) + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (7.7)$$

Definition at line 133 of file MotorUnit.py.

Here is the caller graph for this function:



7.10 MotorUnitPool Namespace Reference

Classes

- class [MotorUnitPool](#)
Class that implements a motor unit pool.

7.11 MuscleHill Namespace Reference

Classes

- class [MuscleHill](#)

7.12 MuscleNoHill Namespace Reference

Classes

- class [MuscleNoHill](#)

7.13 MuscularActivation Namespace Reference

Classes

- class [MuscularActivation](#)

Functions

- def [twitchSaturation](#) (activationsat, b)
Computes the muscle unit force after the nonlinear saturation.

7.13.1 Function Documentation

7.13.1.1 `def MuscularActivation.twitchSaturation (activationsat, b)`

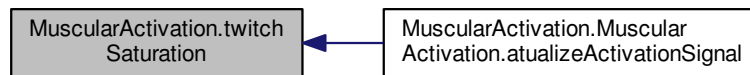
Computes the muscle unit force after the nonlinear saturation.

$$a_{sat} = \frac{1 - e^{-b \cdot a_{nSat}}}{1 + e^{-b \cdot a_{nSat}}} \quad (7.8)$$

- Inputs:
 - **activationsat**: activation signal before the saturation.
 - **b**: saturation function parameter.
- Outputs:
 - Saturated force.

Definition at line 28 of file `MuscularActivation.py`.

Here is the caller graph for this function:



7.14 NeuralTract Namespace Reference

Classes

- class [NeuralTract](#)

Class that implements a a neural tract, composed by the descending commands from the motor cortex.

7.15 NeuralTractUnit Namespace Reference

Classes

- class [NeuralTractUnit](#)

Class that implements a neural tract unit.

7.16 PointProcessGenerator Namespace Reference

Classes

- class [PointProcessGenerator](#)
Generator of point processes.

Functions

- def [gammaPoint](#) (GammaOrder, GammaOrderInv)
*Generates a number according to a Gamma Distribution with an integer order **GammaOrder**.*

7.16.1 Function Documentation

7.16.1.1 def PointProcessGenerator.gammaPoint (GammaOrder, GammaOrderInv)

Generates a number according to a Gamma Distribution with an integer order **GammaOrder**.

- Inputs:
 - **GammaOrder**: integer order of the Gamma distribution.
 - **GammaOrderInv**: inverse of the GammaOrder. This is necessary for computational efficiency.
- Outputs:
 - The number generated from the Gamma distribution.

The number is generated according to:

$$\Gamma = -\frac{1}{\lambda} \ln\left(\prod_{i=1}^{\lambda} U(0, 1)\right) \quad (7.9)$$

where λ is the order of the Gamma distribution and $U(a,b)$ is a uniform distribution from a to b.

Definition at line 38 of file PointProcessGenerator.py.

Here is the caller graph for this function:



7.17 PulseConductanceState Namespace Reference

Classes

- class [PulseConductanceState](#)

Implements the Destexhe pulse approximation of the solution of the states of the Hodgkin-Huxley neuron model.

Functions

- def [compValOn](#) (v0, alpha, beta, t, t0)

Time course of the state during the pulse for the inactivation states and before and after the pulse for the activation states.

- def [compValOff](#) (v0, alpha, beta, t, t0)

Time course of the state during the pulse for the activation states and before and after the pulse for the inactivation states.

7.17.1 Function Documentation

7.17.1.1 def PulseConductanceState.compValOff (v0, alpha, beta, t, t0)

Time course of the state during the pulse for the *activation* states and before and after the pulse for the *inactivation* states.

The value of the state v is computed according to the following equation:

$$v(t) = 1 + (v_0 - 1) \exp[-\alpha(t - t_0)] \quad (7.10)$$

where t_0 is the time at which the pulse changed the value (on to off or off to on) and v_0 is value of the state at that time.

Definition at line 46 of file PulseConductanceState.py.

7.17.1.2 def PulseConductanceState.compValOn (v0, alpha, beta, t, t0)

Time course of the state during the pulse for the *inactivation* states and before and after the pulse for the *activation* states.

The value of the state v is computed according to the following equation:

$$v(t) = v_0 \exp[-\beta(t - t_0)] \quad (7.11)$$

where t_0 is the time at which the pulse changed the value (on to off or off to on) and v_0 is value of the state at that time.

Definition at line 28 of file PulseConductanceState.py.

7.18 simulation Namespace Reference

Functions

- def [simulator](#) ()

7.18.1 Function Documentation

7.18.1.1 def simulation.simulator ()

Definition at line 25 of file simulation.py.

7.19 Synapse Namespace Reference

Classes

- class [Synapse](#)
Implements the synapse model from Destexhe (1994) using the computational method from Lytton (1996).

Functions

- def [compSynapCond](#) (Gmax, Ron, Roff)
Computes the synaptic conductance.
- def [compRon](#) (Non, rInf, Ron, t0, t, tauOn)
Computes the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that have neurotransmitters being released (during the pulse).
- def [compRoff](#) (Roff, t0, t, tauOff)
Computes the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that do not have neurotransmitters being released (before and after the pulse).
- def [compRiStart](#) (ri, t, ti, tPeak, tauOff)
Computes the fraction of bound postsynaptic receptors to neurotransmitters in individual synapses when the neurotransmitter begin (begin of the pulse).
- def [compRiStop](#) (rInf, ri, expFinish)
Computes the fraction of bound postsynaptic receptors to neurotransmitters in individual synapses when the neurotransmitter release stops (the pulse ends).
- def [compRonStart](#) (Ron, ri, synContrib)
Incorporates a new conductance to the set of conductances during a pulse.
- def [compRoffStart](#) (Roff, ri, synContrib)
Incorporates a new conductance to the set of conductances that are not during a pulse.
- def [compRonStop](#) (Ron, ri, synContrib)
Removes a conductance from the set of conductances during a pulse.
- def [compRoffStop](#) (Roff, ri, synContrib)
Removes a conductance from the set of conductances that are not during a pulse.
- def [compDynamicGmax](#) (t, gmax, lastPulse, tau, dynamicGmax, var)

7.19.1 Function Documentation

7.19.1.1 `def Synapse.compDynamicGmax(t, gmax, lastPulse, tau, dynamicGmax, var)`

Definition at line 313 of file Synapse.py.

7.19.1.2 `def Synapse.compRiStart(ri, t, ti, tPeak, tauOff)`

Computes the fraction of bound postsynaptic receptors to neurotransmitters in individual synapses when the neurotransmitter begin (begin of the pulse).

- Inputs:
 - **ri**: the fraction of postsynaptic receptors that were bound to neurotransmitters at the last state change.
 - **t**: current instant, in ms.
 - **ti**: The instant that the last pulse began.
 - **tPeak**: The duration of the pulse.
 - **tauOff**: Time constant after a pulse, in ms.
- Output:
 - individual synapse state value.

It is computed by the following equation:

$$r_{i_{newValue}} = r_{i_{oldValue}} \exp\left(\frac{t_i + T_{dur} - t}{\tau_{off}}\right) \quad (7.12)$$

Definition at line 148 of file Synapse.py.

7.19.1.3 `def Synapse.compRiStop(rInf, ri, expFinish)`

Computes the fraction of bound postsynaptic receptors to neurotransmitters in individual synapses when the neurotransmitter release stops (the pulse ends).

- Inputs:
 - **rInf**: the fraction of postsynaptic receptors that would be bound to neurotransmitters after an infinite amount of time with neurotransmitter being released.
 - **ri**: the fraction of postsynaptic receptors that were bound to neurotransmitters at the last state change.
 - **expFinish**: Is the value of the exponential at the end of the pulse ($\exp(T_{dur}/\tau_{on})$). It is computed before for computational efficiency.
- Output:
 - individual synapse state value.

It is computed by the following equation:

$$r_{i_{newValue}} = r_{\infty} + (r_{i_{oldValue}} - r_{\infty}) \exp\left(\frac{T_{dur}}{\tau_{on}}\right) \quad (7.13)$$

Definition at line 180 of file Synapse.py.

7.19.1.4 `def Synapse.compRoff (Roff, t0, t, tauOff)`

Computes the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that do not have neurotransmitters being released (before and after the pulse).

- Inputs:
 - **Roff**: sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that do not have neurotransmitters being released (before and after the pulse).
 - **t0**: instant that the last spike arrived to the compartment.
 - **t**: current instant, in ms.
 - **tauOff**: time constant after a pulse, in ms.
- Output:
 - The fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that do not have neurotransmitters being released.

It is computed by the following formula:

$$R_{off_{newValue}} = R_{off_{oldValue}} \exp \left(-\frac{t - t0}{\tau_{off}} \right) \quad (7.14)$$

Definition at line 117 of file Synapse.py.

7.19.1.5 `def Synapse.compRoffStart (Roff, ri, synContrib)`

Incorporates a new conductance to the set of conductances that are not during a pulse.

- Inputs:
 - **Roff**: sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that do not have neurotransmitters being released (before and after the pulse).
 - **ri**: fraction of postsynaptic receptors that are bound to neurotransmitters of the individual synapses.
 - **synContrib**: individual conductance contribution to the global synaptic conductance.
- Output:
 - The new value of the sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that do not have neurotransmitters being released (before and after the pulse).

It is computed as:

$$R_{off_{newValue}} = R_{off_{oldValue}} - r_i S_{indCont} \quad (7.15)$$

Definition at line 244 of file Synapse.py.

7.19.1.6 def Synapse.compRoffStop (Roff, ri, synContrib)

Removes a conductance from the set of conductances that are not during a pulse.

- Inputs:
 - **Roff**: sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that do not have neurotransmitters being released (before and after the pulse).
 - **ri**: fraction of postsynaptic receptors that are bound to neurotransmitters of the individual synapses.
 - **synContrib**: individual conductance contribution to the global synaptic conductance.
- Output:
 - The new value of the sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that do not have neurotransmitters being released (before and after the pulse).

It is computed as:

$$R_{off_{newValue}} = R_{off_{oldValue}} + r_i S_{indCont} \quad (7.16)$$

Definition at line 309 of file Synapse.py.

7.19.1.7 def Synapse.compRon (Non, rlnf, Ron, t0, t, tauOn)

Computes the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that have neurotransmitters being released (during the pulse).

- Inputs:
 - **Non**: sum of the fractions of the individual conductances that are receiving neurotransmitter (during pulse) relative to the G_{max} ($N_{on} = \sum_{i=1} g_{ion} / G_{max}$).
 - **rlnf**: the fraction of postsynaptic receptors that would be bound to neurotransmitters after an infinite amount of time with neurotransmitter being released.
 - **Ron**: sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that have neurotransmitters being released (during the pulse).
 - **t0**: instant that the last spike arrived to the compartment.
 - **t**: current instant, in ms.
 - **tauOn**: Time constant during a pulse, in ms. $\tau_{on} = \frac{1}{\alpha \cdot T_{max} + \beta}$.
- Outputs:
 - The fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that have neurotransmitters being released

It is computed by the following equation:

$$R_{on_{newValue}} = N_{on} r_{\infty} \left[1 - \exp \left(-\frac{t - t_0}{\tau_{on}} \right) \right] + R_{on_{oldValue}} \exp \left(-\frac{t - t_0}{\tau_{on}} \right) \quad (7.17)$$

Definition at line 81 of file Synapse.py.

7.19.1.8 def Synapse.compRonStart (*Ron*, *ri*, *synContrib*)

Incorporates a new conductance to the set of conductances during a pulse.

- Inputs:
 - **Ron**: sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that have neurotransmitters being released (during the pulse).
 - **ri**: fraction of postsynaptic receptors that are bound to neurotransmitters of the individual synapses.
 - **synContrib**: individual conductance contribution to the global synaptic conductance.
- Output:
 - The new value of the sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that have neurotransmitters being released (during the pulse).

It is computed as:

$$R_{on_{newValue}} = R_{on_{oldValue}} + r_i S_{indCont} \quad (7.18)$$

Definition at line 211 of file Synapse.py.

7.19.1.9 def Synapse.compRonStop (*Ron*, *ri*, *synContrib*)

Removes a conductance from the set of conductances during a pulse.

- Inputs:
 - **Ron**: sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that have neurotransmitters being released (during the pulse).
 - **ri**: fraction of postsynaptic receptors that are bound to neurotransmitters of the individual synapses.
 - **synContrib**: individual conductance contribution to the global synaptic conductance.
- Output:
 - The new value of the sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that have neurotransmitters being released (during the pulse).

It is computed as:

$$R_{on_{newValue}} = R_{on_{oldValue}} - r_i S_{indCont} \quad (7.19)$$

Definition at line 275 of file Synapse.py.

7.19.1.10 def Synapse.compSynapCond (*Gmax*, *Ron*, *Roff*)

Computes the synaptic conductance.

- Input:
 - **Gmax**: the sum of individual conductances of all synapses in the compartment, in μS .
 - **Ron**: sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that have neurotransmitters being released (during the pulse).
 - **Roff**: sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that do not have neurotransmitters being released (before and after the pulse).
- Output:
 - the synaptic conductance of all synapses in the compartment, in μS .

It is computed by the following formula:

$$G = G_{max}(R_{on} + R_{off}) \quad (7.20)$$

where G is the synaptic conductance of all synapses in the compartment.

Definition at line 41 of file Synapse.py.

7.20 SynapsesFactory Namespace Reference

Classes

- class [SynapsesFactory](#)
Class to build all the synapses in the system.

7.21 SynapticNoise Namespace Reference

Classes

- class [SynapticNoise](#)
Class that implements a synaptic noise for a pool of neurons.

Chapter 8

Class Documentation

8.1 AxonDelay.AxonDelay Class Reference

Class that implements a delay correspondent to the nerve.

Public Member Functions

- `def __init__` (self, conf, nerve, pool, [index](#))
Constructor.
- `def addTerminalSpike` (self, t)
Indicates to the [AxonDelay](#) object that a spike has occurred in the Terminal.
- `def addSpinalSpike` (self, t)
Indicates to the [AxonDelay](#) object that a spike has occurred in the soma.

Public Attributes

- [index](#)
Integer corresponding to the motor unit order in the pool, according to the Henneman's principle (size principle).
- [length_m](#)
Length, in m, of the part of the nerve that is not modelled as a delay.
- [velocity_m_s](#)
Velocity of conduction, in m/s, of the part of the nerve that is not modelled as a delay.
- [stimulusPositiontoTerminal](#)
Distance, in m, of the stimulus position to the terminal.
- [latencyStimulusSpinal_ms](#)
time, in ms, that the signal takes to travel between the stimulus and the spinal cord.
- [latencySpinalTerminal_ms](#)
time, in ms, that the signal takes to travel between the spinal cord and the terminal.
- [latencyStimulusTerminal_ms](#)
time, in ms, tat the signal takes to travel between the stimulus and the terminal.
- [terminalSpikeTrain](#)
Float with instant, in ms, of the last spike in the terminal.

8.1.1 Detailed Description

Class that implements a delay correspondent to the nerve.

This class corresponds to the part of the axon that is modeled with no dynamics. Ideally this class would not exist and all the axon would be modelled in the motor unit or sensory class with the proper dynamics.

Definition at line 16 of file AxonDelay.py.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 `def AxonDelay.AxonDelay.__init__(self, conf, nerve, pool, index)`

Constructor.

- Inputs:
 - **conf**: [Configuration](#) object with the simulation parameters.
 - **nerve**: string with type of the nerve. It can be *PTN* (posterior tibial nerve) or *CPN* (common peroneal nerve).
 - **pool**: string with Motor unit pool to which the motor unit belongs.
 - **index**: integer corresponding to the motor unit order in the pool, according to the Henneman's principle (size principle).

Definition at line 35 of file AxonDelay.py.

8.1.3 Member Function Documentation

8.1.3.1 `def AxonDelay.AxonDelay.addSpinalSpike(self, t)`

Indicates to the [AxonDelay](#) object that a spike has occurred in the soma.

- Inputs:
 - **t**: current instant, in ms.

Definition at line 76 of file AxonDelay.py.

Here is the call graph for this function:



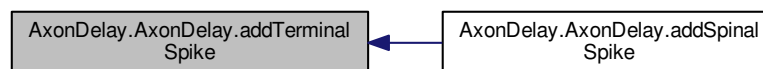
8.1.3.2 `def AxonDelay.AxonDelay.addTerminalSpike (self, t)`

Indicates to the [AxonDelay](#) object that a spike has occurred in the Terminal.

- Inputs:
 - `t`: current instant, in ms.

Definition at line 65 of file `AxonDelay.py`.

Here is the caller graph for this function:



8.1.4 Member Data Documentation

8.1.4.1 `AxonDelay.AxonDelay.index`

Integer corresponding to the motor unit order in the pool, according to the Henneman's principle (size principle).

Definition at line 39 of file `AxonDelay.py`.

8.1.4.2 `AxonDelay.AxonDelay.latencySpinalTerminal_ms`

time, in ms, that the signal takes to travel between the spinal cord and the terminal.

Definition at line 50 of file `AxonDelay.py`.

8.1.4.3 `AxonDelay.AxonDelay.latencyStimulusSpinal_ms`

time, in ms, that the signal takes to travel between the stimulus and the spinal cord.

Definition at line 48 of file `AxonDelay.py`.

8.1.4.4 `AxonDelay.AxonDelay.latencyStimulusTerminal_ms`

time, in ms, that the signal takes to travel between the stimulus and the terminal.

Definition at line 52 of file `AxonDelay.py`.

8.1.4.5 AxonDelay.AxonDelay.length_m

Length, in m, of the part of the nerve that is not modelled as a delay.

Definition at line 42 of file AxonDelay.py.

8.1.4.6 AxonDelay.AxonDelay.stimulusPositiontoTerminal

Distance, in m, of the stimulus position to the terminal.

Definition at line 46 of file AxonDelay.py.

8.1.4.7 AxonDelay.AxonDelay.terminalSpikeTrain

Float with instant, in ms, of the last spike in the terminal.

Definition at line 55 of file AxonDelay.py.

8.1.4.8 AxonDelay.AxonDelay.velocity_m_s

Velocity of conduction, in m/s, of the part of the nerve that is not modelled as a delay.

Definition at line 44 of file AxonDelay.py.

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

- [AxonDelay.py](#)

8.2 ChannelConductance.ChannelConductance Class Reference

Class that implements a model of the ionic Channels in a compartment.

Public Member Functions

- def `__init__` (self, `kind`, `conf`, `compArea`, `pool`, `neuronKind`, `index`)
Constructor.
- def `computeCurrent` (self, `t`, `V_mV`)
Computes the current generated by the ionic Channel.
- def `compCondKf` (self, `V_mV`)
Computes the conductance of a Kf Channel.
- def `compCondKs` (self, `V_mV`)
Computes the conductance of a slow potassium Channel.
- def `compCondNa` (self, `V_mV`)
Computes the conductance of a Na Channel.

Public Attributes

- [kind](#)
string with the type of the ionic channel.
- [condState](#)
List of ConductanceState objects, representing each state of the ionic channel.
- [EqPot_mV](#)
Equilibrium Potential of the ionic channel, mV.
- [gmax_muS](#)
Maximal conductance, in μS , of the ionic channel.
- [stateType](#)
String with type of dynamics of the states.
- [compCond](#)
Function that computes the conductance dynamics.
- [lenStates](#)
Integer with the number of states in the ionic channel.

8.2.1 Detailed Description

Class that implements a model of the ionic Channels in a compartment.

Definition at line 16 of file ChannelConductance.py.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 `def ChannelConductance.ChannelConductance.__init__(self, kind, conf, compArea, pool, neuronKind, index)`

Constructor.

Builds an ionic channel conductance.

-Inputs:

- **kind**: string with the type of the ionic channel. For now it can be *Na* (Sodium), *Ks* (slow Potassium), *Kf* (fast Potassium) or *Ca* (Calcium).
- **conf**: instance of the [Configuration](#) class (see [Configuration](#) file).
- **compArea**: float with the area of the compartment that the Channel belongs, in cm^2 .
- **pool**: the pool that this state belongs.
- **index**: the index of the unit that this state belongs.

Definition at line 38 of file ChannelConductance.py.

8.2.3 Member Function Documentation

8.2.3.1 `def ChannelConductance.ChannelConductance.compCondKf (self, V_mV)`

Computes the conductance of a Kf Channel.

This function is assigned as `self.compCond` to a Kf Channel at the class constructor.

- Input:
 - **V_mV**: membrane potential of the compartment in mV.

Output:

- Conductance in μS .

It is computed as:

$$g = g_{max} n^4 (E_0 - V) \quad (8.1)$$

where E_0 is the equilibrium potential of the compartment, V is the membrane potential and n is the state of a fast potassium channel..

Definition at line 114 of file `ChannelConductance.py`.

8.2.3.2 `def ChannelConductance.ChannelConductance.compCondKs (self, V_mV)`

Computes the conductance of a slow potassium Channel.

This function is assigned as `self.compCond` to a Ks Channel at the class constructor.

- Input:
 - **V_mV**: membrane potential of the compartment in mV.
- Output:
 - Conductance in μS .

It is computed as:

$$g = g_{max} q^2 (E_0 - V) \quad (8.2)$$

where E_0 is the equilibrium potential of the compartment, V is the membrane potential and q is the state of a slow potassium channel.

Definition at line 137 of file `ChannelConductance.py`.

8.2.3.3 `def ChannelConductance.ChannelConductance.compCondNa (self, V_mV)`

Computes the conductance of a Na Channel.

This function is assigned as `self.compCond` to a Na Channel at the class constructor. -Input:

- **V_mV**: membrane potential of the compartment in mV.

Output:

- Conductance in μS .

It is computed as:

$$g = g_{max} m^3 h (E_0 - V) \quad (8.3)$$

where E_0 is the equilibrium potential of the compartment, V is the membrane potential and m and h are the states of a sodium channel..

Definition at line 158 of file ChannelConductance.py.

8.2.3.4 `def ChannelConductance.ChannelConductance.computeCurrent (self, t, V_mV)`

Computes the current generated by the ionic Channel.

- Inputs:
 - **t**: instant in ms.
 - **V_mV**: membrane potential of the compartment in mV.
- Outputs:
 - Ionic current, in nA

Definition at line 90 of file ChannelConductance.py.

8.2.4 Member Data Documentation

8.2.4.1 `ChannelConductance.ChannelConductance.compCond`

Function that computes the conductance dynamics.

Definition at line 59 of file ChannelConductance.py.

8.2.4.2 `ChannelConductance.ChannelConductance.condState`

List of ConductanceState objects, representing each state of the ionic channel.

Definition at line 44 of file ChannelConductance.py.

8.2.4.3 ChannelConductance.ChannelConductance.EqPot_mV

Equilibrium Potential of the ionic channel, mV.

Definition at line 47 of file ChannelConductance.py.

8.2.4.4 ChannelConductance.ChannelConductance.gmax_muS

Maximal conductance, in μS , of the ionic channel.

Definition at line 49 of file ChannelConductance.py.

8.2.4.5 ChannelConductance.ChannelConductance.kind

string with the type of the ionic channel.

For now it can be *Na* (Sodium), *Ks* (slow Potassium), *Kf* (fast Potassium) or *Ca* (Calcium).

Definition at line 42 of file ChannelConductance.py.

8.2.4.6 ChannelConductance.ChannelConductance.lenStates

Integer with the number of states in the ionic channel.

Definition at line 73 of file ChannelConductance.py.

8.2.4.7 ChannelConductance.ChannelConductance.stateType

String with type of dynamics of the states.

For now it accepts the string pulse.

Definition at line 51 of file ChannelConductance.py.

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

- [ChannelConductance.py](#)

8.3 Compartment.Compartment Class Reference

Class that implements a neural compartment.

Public Member Functions

- def `__init__` (self, [kind](#), conf, pool, [index](#), [neuronKind](#))
Constructor.
- def `computeCurrent` (self, t, [V_mV](#))
Computes the active currents of the compartment.

Public Attributes

- [Channels](#)
List of [ChannelConductance](#) objects in the [Compartment](#).
- [neuronKind](#)
String with the type of the motor unit.
- [SynapsesOut](#)
List of summed synapses (see Lytton, 1996) that the [Compartment](#) do with other neural components.
- [SynapsesIn](#)
List of summed synapses (see Lytton, 1996) that the [Compartment](#) receive from other neural components.
- [kind](#)
The kind of compartment.
- [index](#)
Integer corresponding to the motor unit order in the pool, according to the Henneman's principle (size principle).
- [length_mum](#)
Length of the compartment, in μm .
- [diameter_mum](#)
Diameter of the compartment, in μm .
- [capacitance_nF](#)
Capacitance of the compartment, in nF.
- [gLeak](#)
Leak conductance of the compartment, in MS.
- [numberChannels](#)
Integer with the number of ionic channels.

8.3.1 Detailed Description

Class that implements a neural compartment.

For now it is implemented *dendrite* and *soma*.

Definition at line 40 of file Compartment.py.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 `def Compartment.Compartment.__init__(self, kind, conf, pool, index, neuronKind)`

Constructor.

- Inputs:
 - **kind**: The kind of compartment. For now, it can be *soma* or *dendrite*.
 - **conf**: [Configuration](#) object with the simulation parameters.
 - **pool**: string with Motor unit pool to which the motor unit belongs.
 - **index**: integer corresponding to the motor unit order in the pool, according to the Henneman's principle (size principle).
 - **neuronKind**: string with the type of the motor unit. It can be *S* (slow), *FR* (fast and resistant), and *FF* (fast and fatigable).

Definition at line 60 of file Compartment.py.

8.3.3 Member Function Documentation

8.3.3.1 `def Compartment.Compartment.computeCurrent (self, t, V_mV)`

Computes the active currents of the compartment.

Active currents are the currents from the ionic channels and from the synapses.

- Inputs:
 - **t**: current instant, in ms.
 - **V_mV**: membrane potential, in mV.

Definition at line 114 of file `Compartment.py`.

8.3.4 Member Data Documentation

8.3.4.1 `Compartment.Compartment.capacitance_nF`

Capacitance of the compartment, in nF.

Definition at line 89 of file `Compartment.py`.

8.3.4.2 `Compartment.Compartment.Channels`

List of [ChannelConductance](#) objects in the [Compartment](#).

Definition at line 63 of file `Compartment.py`.

8.3.4.3 `Compartment.Compartment.diameter_mum`

Diameter of the compartment, in μm .

Definition at line 85 of file `Compartment.py`.

8.3.4.4 `Compartment.Compartment.gLeak`

Leak conductance of the compartment, in MS.

Definition at line 92 of file `Compartment.py`.

8.3.4.5 `Compartment.Compartment.index`

Integer corresponding to the motor unit order in the pool, according to the Henneman's principle (size principle).

Definition at line 80 of file `Compartment.py`.

8.3.4.6 `Compartment.Compartment.kind`

The kind of compartment.

For now, it can be *soma* or *dendrite*.

Definition at line 76 of file `Compartment.py`.

8.3.4.7 `Compartment.Compartment.length_mum`

Length of the compartment, in μm .

Definition at line 83 of file `Compartment.py`.

8.3.4.8 `Compartment.Compartment.neuronKind`

String with the type of the motor unit.

It can be *S* (slow), *FR* (fast and resistant), and *FF* (fast and fatigable).

Definition at line 66 of file `Compartment.py`.

8.3.4.9 `Compartment.Compartment.numberChannels`

Integer with the number of ionic channels.

Definition at line 102 of file `Compartment.py`.

8.3.4.10 `Compartment.Compartment.SynapsesIn`

List of summed synapses (see Lytton, 1996) that the [Compartment](#) receive from other neural components.

Definition at line 71 of file `Compartment.py`.

8.3.4.11 `Compartment.Compartment.SynapsesOut`

List of summed synapses (see Lytton, 1996) that the [Compartment](#) do with other neural components.

Definition at line 68 of file `Compartment.py`.

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

- [Compartment.py](#)

8.4 Configuration.Configuration Class Reference

Class that builds an object of [Configuration](#), based on a configuration file.

Public Member Functions

- `def __init__ (self, filename)`
Constructor.
- `def parameterSet (self, paramTag, pool, index)`
Function that returns the value of wished parameter specified in the paramTag variable.
- `def inputFunctionGet (self, function)`
Returns a numpy array with the values of the function for the whole simulation.
- `def determineSynapses (self, neuralSource)`
Function used to determine all the synapses that a given pool makes.

Public Attributes

- `confArray`
An array with all the simulation parameters.
- `timeStep_ms`
Time step of the numerical solution of the differential equation.
- `simDuration_ms`
Total length of the simulation in ms.
- `timeStepByTwo_ms`
The variable timeStep divided by two, for computational efficiency.
- `timeStepBySix_ms`
The variable timeStep divided by six, for computational efficiency.

8.4.1 Detailed Description

Class that builds an object of [Configuration](#), based on a configuration file.

Definition at line 38 of file Configuration.py.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 `def Configuration.Configuration.__init__ (self, filename)`

Constructor.

Builds the [Configuration](#) object. A [Configuration](#) object is responsible to set the variables that are used in the whole system, such as `timeStep` and `simDuration`.

- Inputs:
 - **filename**: name of the file with the parameter values. The extension of the file should be `.rmto`.

Definition at line 52 of file Configuration.py.

8.4.3 Member Function Documentation

8.4.3.1 `def Configuration.Configuration.determineSynapses (self, neuralSource)`

Function used to determine all the synapses that a given pool makes.

It is used in the [SynapsesFactory](#) class.

- Inputs:
 - **neuralSource** - string with the pool name from which is desired to know what synapses it will make.
- Outputs:
 - array of strings with all the synapses target that the neuralSource will make.

Definition at line 164 of file Configuration.py.

8.4.3.2 `def Configuration.Configuration.inputFunctionGet (self, function)`

Returns a numpy array with the values of the function for the whole simulation.

It is used to obtain before the simulation run all the values of the inputs.

- Inputs:
 - **function**: function from which is desired to obtain its values during the simulation duration.
- Output:
 - ndarray with the function values for each instant.

Definition at line 148 of file Configuration.py.

8.4.3.3 `def Configuration.Configuration.parameterSet (self, paramTag, pool, index)`

Function that returns the value of wished parameter specified in the paramTag variable.

In the case of min/max parameters, the value returned is the specific to the index of the unit that called the function.

- Inputs:
 - **paramTag**: string with the name of the wished parameter as in the first column of the rmto file.
 - **pool**: pool from which the unit that will receive the parameter value belongs. For example SOL. It is used only in the parameters that have a range.
 - **index**: index of the unit. It is an integer.
- Outputs:
 - required parameter value

Definition at line 92 of file Configuration.py.

8.4.4 Member Data Documentation

8.4.4.1 Configuration.Configuration.confArray

An array with all the simulation parameters.

Definition at line 55 of file Configuration.py.

8.4.4.2 Configuration.Configuration.simDuration_ms

Total length of the simulation in ms.

Definition at line 65 of file Configuration.py.

8.4.4.3 Configuration.Configuration.timeStep_ms

Time step of the numerical solution of the differential equation.

Definition at line 62 of file Configuration.py.

8.4.4.4 Configuration.Configuration.timeStepBySix_ms

The variable timeStep divided by six, for computational efficiency.

Definition at line 69 of file Configuration.py.

8.4.4.5 Configuration.Configuration.timeStepByTwo_ms

The variable timeStep divided by two, for computational efficiency.

Definition at line 67 of file Configuration.py.

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

- [Configuration.py](#)

8.5 Interneuron.Interneuron Class Reference

Class that implements a motor unit model.

Public Member Functions

- `def __init__ (self, conf, pool, index)`
- Constructor.*
- `def atualizeInterneuron (self, t)`
- Atualize the dynamical and nondynamical (delay) parts of the motor unit.*
- `def atualizeCompartments (self, t)`
- Atualize all neural compartments.*
- `def dVdt (self, t, V)`
- Compute the potential derivative of all compartments of the motor unit.*
- `def addSomaSpike (self, t)`
- When the soma potential is above the threshold a spike is added to the soma.*
- `def transmitSpikes (self, t)`

Public Attributes

- [conf](#)
- Configuration object with the simulation parameters.*
- [pool](#)
- [kind](#)
- [tSomaSpike](#)
- The instant of the last spike of the Motor unit at the Soma compartment.*
- [somaSpikeTrain](#)
- Vector with the instants of spikes at the soma.*
- [index](#)
- Integer corresponding to the [Interneuron](#) order in the pool.*
- [compartment](#)
- Vector of [Compartment](#) of the Motor Unit.*
- [threshold_mV](#)
- Value of the membrane potential, in mV, that is considered a spike.*
- [position_mm](#)
- Anatomical position of the neuron, in mm.*
- [compNumber](#)
- Number of compartments.*
- [v_mV](#)
- Vector with membrane potential, in mV, of all compartments.*
- [capacitanceInv](#)
- Vector with the inverse of the capacitance of all compartments.*
- [ilonic](#)
- Vector with current, in nA, of each compartment coming from other elements of the model.*
- [iInjected](#)
- Vector with the current, in nA, injected in each compartment.*
- [G](#)
- Matrix of the conductance of the motoneuron.*
- [somaIndex](#)
- index of the soma compartment.*
- [RefPer_ms](#)
- Refractory period, in ms, of the motoneuron.*
- [terminalSpikeTrain](#)
- Vector with the instants of spikes at the terminal.*
- [SynapsesOut](#)
- Build synapses.*
- [transmitSpikesThroughSynapses](#)
- [indicesOfSynapsesOnTarget](#)

8.5.1 Detailed Description

Class that implements a motor unit model.

Encompasses a motoneuron and a muscle unit.

Definition at line 66 of file Interneuron.py.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 `def Interneuron.Interneuron.__init__(self, conf, pool, index)`

Constructor.

- Inputs:
 - **conf**: [Configuration](#) object with the simulation parameters.
 - **pool**: string with [Interneuron](#) pool to which the motor unit belongs. It can be *RC* (Renshaw cell), *laIn* (la [Interneuron](#)), *lbIn* (lb [Interneuron](#)) and *gII*.
 - **index**: integer corresponding to the motor unit order in the pool, according to the Henneman's principle (size principle).

Definition at line 83 of file Interneuron.py.

8.5.3 Member Function Documentation

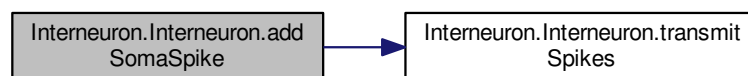
8.5.3.1 `def Interneuron.Interneuron.addSomaSpike (self, t)`

When the soma potential is above the threshold a spike is added to the soma.

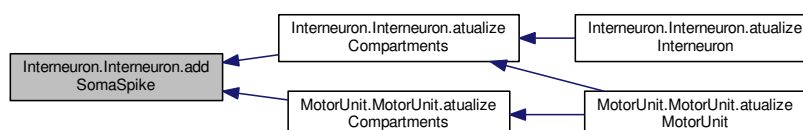
- Inputs:
 - **t**: current instant, in ms.

Definition at line 217 of file Interneuron.py.

Here is the call graph for this function:



Here is the caller graph for this function:



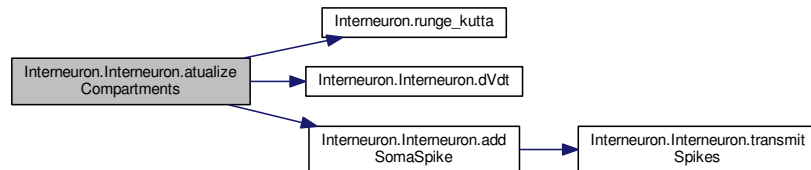
8.5.3.2 def Interneuron.Interneuron.atualizeCompartments (self, t)

Atualize all neural compartments.

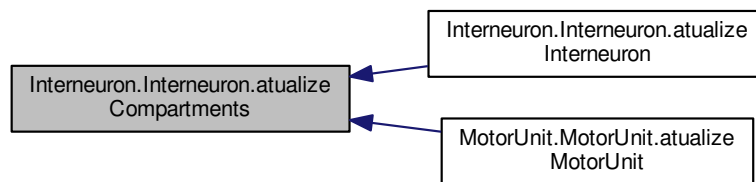
- Inputs:
 - t: current instant, in ms.

Definition at line 177 of file Interneuron.py.

Here is the call graph for this function:



Here is the caller graph for this function:



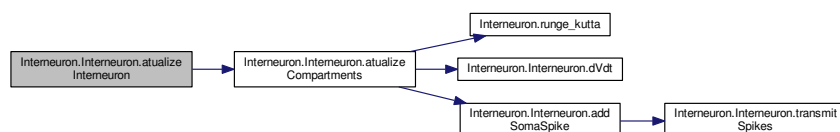
8.5.3.3 def Interneuron.Interneuron.atualizeInterneuron (self, t)

Atualize the dynamical and nondynamical (delay) parts of the motor unit.

- Inputs:
 - t: current instant, in ms.

Definition at line 167 of file Interneuron.py.

Here is the call graph for this function:



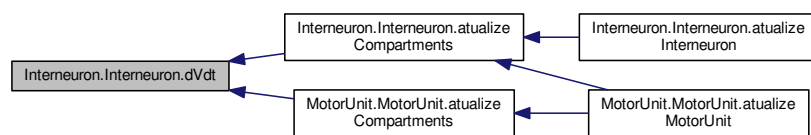
8.5.3.4 `def Interneuron.Interneuron.dVdt (self, t, V)`

Compute the potential derivative of all compartments of the motor unit.

- Inputs:
 - **t**: current instant, in ms.
 - **V**: Vector with the current potential value of all neural compartments of the motor unit.

Definition at line 202 of file Interneuron.py.

Here is the caller graph for this function:

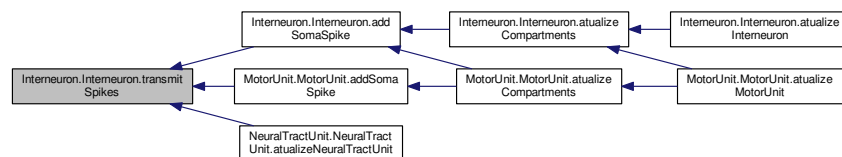


8.5.3.5 `def Interneuron.Interneuron.transmitSpikes (self, t)`

- Inputs:
 - **t**: current instant, in ms.

Definition at line 231 of file Interneuron.py.

Here is the caller graph for this function:



8.5.4 Member Data Documentation

8.5.4.1 `Interneuron.Interneuron.capacitanceInv`

Vector with the inverse of the capacitance of all compartments.

Definition at line 126 of file Interneuron.py.

8.5.4.2 Interneuron.Interneuron.compartment

Vector of [Compartment](#) of the Motor Unit.

Definition at line 101 of file Interneuron.py.

8.5.4.3 Interneuron.Interneuron.compNumber

Number of compartments.

Definition at line 112 of file Interneuron.py.

8.5.4.4 Interneuron.Interneuron.conf

[Configuration](#) object with the simulation parameters.

Definition at line 86 of file Interneuron.py.

8.5.4.5 Interneuron.Interneuron.G

Matrix of the conductance of the motoneuron.

Multiplied by the vector `self.v_mV`, results in the passive currents of each compartment.

Definition at line 140 of file Interneuron.py.

8.5.4.6 Interneuron.Interneuron.ilnjected

Vector with the current, in nA, injected in each compartment.

Definition at line 132 of file Interneuron.py.

8.5.4.7 Interneuron.Interneuron.ilonic

Vector with current, in nA, of each compartment coming from other elements of the model.

For example from ionic channels and synapses.

Definition at line 130 of file Interneuron.py.

8.5.4.8 Interneuron.Interneuron.index

Integer corresponding to the [Interneuron](#) order in the pool.

Definition at line 99 of file Interneuron.py.

8.5.4.9 Interneuron.Interneuron.indicesOfSynapsesOnTarget

Definition at line 157 of file Interneuron.py.

8.5.4.10 Interneuron.Interneuron.kind

Definition at line 90 of file Interneuron.py.

8.5.4.11 Interneuron.Interneuron.pool

Definition at line 88 of file Interneuron.py.

8.5.4.12 Interneuron.Interneuron.position_mm

Anatomical position of the neuron, in mm.

Definition at line 106 of file Interneuron.py.

8.5.4.13 Interneuron.Interneuron.RefPer_ms

Refractory period, in ms, of the motoneuron.

Definition at line 147 of file Interneuron.py.

8.5.4.14 Interneuron.Interneuron.somaIndex

index of the soma compartment.

Definition at line 144 of file Interneuron.py.

8.5.4.15 Interneuron.Interneuron.somaSpikeTrain

Vector with the instants of spikes at the soma.

Definition at line 97 of file Interneuron.py.

8.5.4.16 Interneuron.Interneuron.SynapsesOut

Build synapses.

Definition at line 155 of file Interneuron.py.

8.5.4.17 Interneuron.Interneuron.terminalSpikeTrain

Vector with the instants of spikes at the terminal.

Definition at line 150 of file Interneuron.py.

8.5.4.18 Interneuron.Interneuron.threshold_mV

Value of the membrane potential, in mV, that is considered a spike.

Definition at line 103 of file Interneuron.py.

8.5.4.19 Interneuron.Interneuron.transmitSpikesThroughSynapses

Definition at line 156 of file Interneuron.py.

8.5.4.20 Interneuron.Interneuron.tSomaSpike

The instant of the last spike of the Motor unit at the Soma compartment.

Definition at line 94 of file Interneuron.py.

8.5.4.21 Interneuron.Interneuron.v_mV

Vector with membrane potential,in mV, of all compartments.

Definition at line 114 of file Interneuron.py.

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

- [Interneuron.py](#)

8.6 InterneuronPool.InterneuronPool Class Reference

Class that implements a motor unit pool.

Public Member Functions

- `def __init__ (self, conf, pool)`
Constructor.
- `def atualizeInterneuronPool (self, t)`
Update all parts of the Motor Unit pool.
- `def listSpikes (self)`
List the spikes that occurred in the soma and in the terminal of the different motor units.

Public Attributes

- [kind](#)
Indicates that is Motor Unit pool.
- [conf](#)
Configuration object with the simulation parameters.
- [pool](#)
String with Motor unit pool to which the motor unit belongs.
- [Nnumber](#)
Number of Neurons.
- [unit](#)
List of [Interneuron](#) objects.
- [poolSomaSpikes](#)
Vector with the instants of spikes in the soma compartment, in ms.

8.6.1 Detailed Description

Class that implements a motor unit pool.

Encompasses a set of motor units that controls a single muscle.

Definition at line 19 of file InterneuronPool.py.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 `def InterneuronPool.InterneuronPool.__init__(self, conf, pool)`

Constructor.

- Inputs:
 - **conf**: [Configuration](#) object with the simulation parameters.
 - **pool**: string with [Interneuron](#) pool to which the motor unit belongs.

Definition at line 31 of file InterneuronPool.py.

8.6.3 Member Function Documentation

8.6.3.1 `def InterneuronPool.InterneuronPool.atualizeInterneuronPool(self, t)`

Update all parts of the Motor Unit pool.

It consists to update all motor units, the activation signal and the muscle force.

- Inputs:
 - **t**: current instant, in ms.

Definition at line 64 of file InterneuronPool.py.

8.6.3.2 `def InterneuronPool.InterneuronPool.listSpikes (self)`

List the spikes that occurred in the soma and in the terminal of the different motor units.

Definition at line 73 of file `InterneuronPool.py`.

8.6.4 Member Data Documentation

8.6.4.1 `InterneuronPool.InterneuronPool.conf`

[Configuration](#) object with the simulation parameters.

Definition at line 37 of file `InterneuronPool.py`.

8.6.4.2 `InterneuronPool.InterneuronPool.kind`

Indicates that is Motor Unit pool.

Definition at line 34 of file `InterneuronPool.py`.

8.6.4.3 `InterneuronPool.InterneuronPool.Nnumber`

Number of Neurons.

Definition at line 41 of file `InterneuronPool.py`.

8.6.4.4 `InterneuronPool.InterneuronPool.pool`

String with Motor unit pool to which the motor unit belongs.

Definition at line 39 of file `InterneuronPool.py`.

8.6.4.5 `InterneuronPool.InterneuronPool.poolSomaSpikes`

Vector with the instants of spikes in the soma compartment, in ms.

Definition at line 50 of file `InterneuronPool.py`.

8.6.4.6 `InterneuronPool.InterneuronPool.unit`

List of [Interneuron](#) objects.

Definition at line 44 of file `InterneuronPool.py`.

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

- [InterneuronPool.py](#)

8.7 jointAnkleForceTask.jointAnkleForceTask Class Reference

Public Member Functions

- `def __init__ (self, conf, pools)`
- `def atualizeAnkle (self, t, ankleAngle)`
- `def atualizeAngle (self, t, ankleAngle)`

Public Attributes

- [conf](#)
- [muscles](#)
- [ankleAngle_rad](#)

8.7.1 Detailed Description

Definition at line 12 of file jointAnkleForceTask.py.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 `def jointAnkleForceTask.jointAnkleForceTask.__init__ (self, conf, pools)`

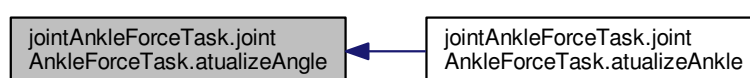
Definition at line 14 of file jointAnkleForceTask.py.

8.7.3 Member Function Documentation

8.7.3.1 `def jointAnkleForceTask.jointAnkleForceTask.atualizeAngle (self, t, ankleAngle)`

Definition at line 37 of file jointAnkleForceTask.py.

Here is the caller graph for this function:



8.7.3.2 `def jointAnkleForceTask.jointAnkleForceTask.atualizeAnkle (self, t, ankleAngle)`

Definition at line 28 of file jointAnkleForceTask.py.

Here is the call graph for this function:



8.7.4 Member Data Documentation

8.7.4.1 `jointAnkleForceTask.jointAnkleForceTask.ankleAngle_rad`

Definition at line 25 of file jointAnkleForceTask.py.

8.7.4.2 `jointAnkleForceTask.jointAnkleForceTask.conf`

Definition at line 16 of file jointAnkleForceTask.py.

8.7.4.3 `jointAnkleForceTask.jointAnkleForceTask.muscles`

Definition at line 17 of file jointAnkleForceTask.py.

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

- [jointAnkleForceTask.py](#)

8.8 jointAnklePositionTask.jointAnklePositionTask Class Reference

Public Member Functions

- `def __init__ (self, conf, pool, MUnumber, MUsynthesize, musculotendonLength, unit)`
- `def atualizeForce (self, activation_Sat, musculoTendonLength)`
Compute the muscle force when no muscle dynamics (Hill model) is used.
- `def atualizeActivation (self, activation_Sat)`
- `def computePennationAngle (self)`
- `def computeForceLengthTypeI (self)`
- `def computeForceLengthTypeII (self)`
- `def computeForceVelocityTypeI (self)`
- `def computeForceVelocityTypeII (self)`
- `def computeAcceleration (self)`
- `def dLdt (self)`
- `def atualizeMuscleForce (self)`
- `def atualizeTendonForce (self)`
- `def computeElasticElementForce (self)`
- `def computeViscousElementForce (self)`
- `def computeTypeIActiveForce (self)`
- `def computeTypeIIActiveForce (self)`
- `def atualizeLenghtsAndVelocity (self)`

Public Attributes

- [conf](#)
- [pool](#)
- [MUnumber](#)
- [MUtypeInumber](#)
- [timeIndex](#)
- [twTet](#)

Twitch-tetanus relationship (see `atualizeForce` function explanation)
- [twitchAmp_N](#)

Amplitude of the muscle unit twitch, in N (see `atualizeForce` function explanation).
- [maximumActivationForce](#)

This is used for normalization purposes.
- [force](#)

Muscle force along time, in N.
- [tendonForce_N](#)
- [contractileForce_N](#)
- [elasticForce_N](#)
- [viscousForce_N](#)
- [length_m](#)
- [velocity_m_ms](#)
- [tendonLength_m](#)
- [pennationAngle_rad](#)
- [activationTypeI](#)
- [activationTypeII](#)
- [optimalLength_m](#)
- [pennationAngleAtOptimalLengthSin](#)
- [maximumForce_N](#)

Maximum force of the Hill model, in N.
- [elasticity](#)
- [strain](#)
- [viscosity](#)
- [mass](#)
- [tendonElasticity](#)
- [tendonLinearOnsetLength](#)
- [tendonCurvatureConstant](#)
- [optimalTendonLength](#)
- [lengthNorm](#)
- [velocityNorm](#)
- [tendonLengthNorm](#)
- [forceNorm](#)
- [tendonForceNorm](#)
- [b_TypeI](#)
- [b_TypeII](#)
- [p_TypeI](#)
- [p_TypeII](#)
- [w_TypeI](#)
- [w_TypeII](#)
- [d_TypeI](#)
- [d_TypeII](#)
- [a0_TypeI](#)
- [a0_TypeII](#)
- [a1_TypeI](#)
- [a1_TypeII](#)

- [a2_TypeI](#)
- [a2_TypeII](#)
- [c0_TypeI](#)
- [c0_TypeII](#)
- [c1_TypeI](#)
- [c1_TypeII](#)
- [Vmax_TypeI](#)
- [Vmax_TypeII](#)

8.8.1 Detailed Description

Definition at line 12 of file jointAnklePositionTask.py.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 `def jointAnklePositionTask.jointAnklePositionTask.__init__(self, conf, pool, MUnumber, MUtypeInumber, musculotendonLength, unit)`

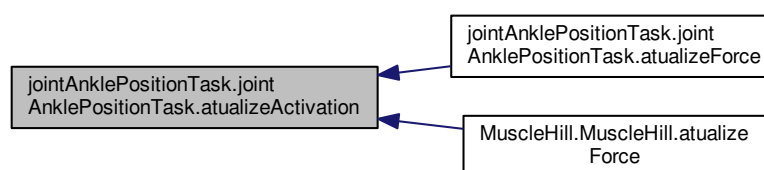
Definition at line 14 of file jointAnklePositionTask.py.

8.8.3 Member Function Documentation

8.8.3.1 `def jointAnklePositionTask.jointAnklePositionTask.atualizeActivation (self, activation_Sat)`

Definition at line 178 of file jointAnklePositionTask.py.

Here is the caller graph for this function:



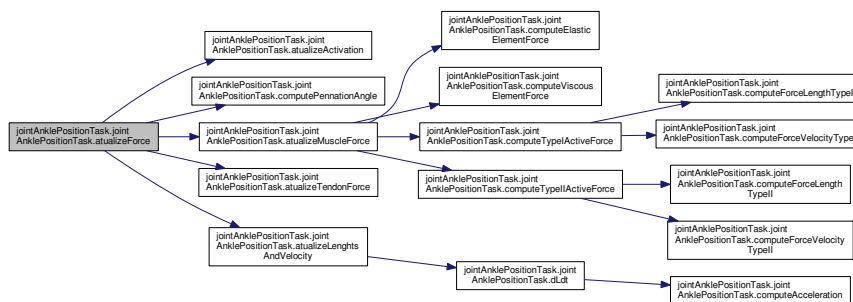
8.8.3.2 `def jointAnklePositionTask.jointAnklePositionTask.atualizeForce (self, activation_Sat, musculoTendonLength)`

Compute the muscle force when no muscle dynamics (Hill model) is used.

This operation is vectorized. Each element of the vectors correspond to one motor unit. For each motor unit, the force is computed by the following formula:

Definition at line 160 of file `jointAnklePositionTask.py`.

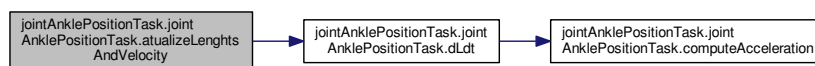
Here is the call graph for this function:



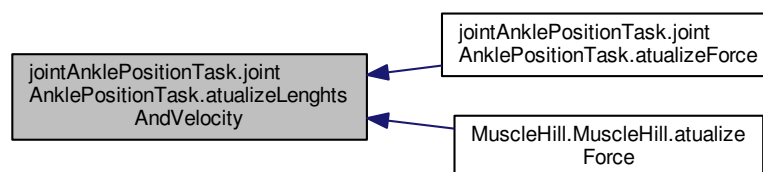
8.8.3.3 `def jointAnklePositionTask.jointAnklePositionTask.atualizeLengthsAndVelocity (self)`

Definition at line 237 of file `jointAnklePositionTask.py`.

Here is the call graph for this function:



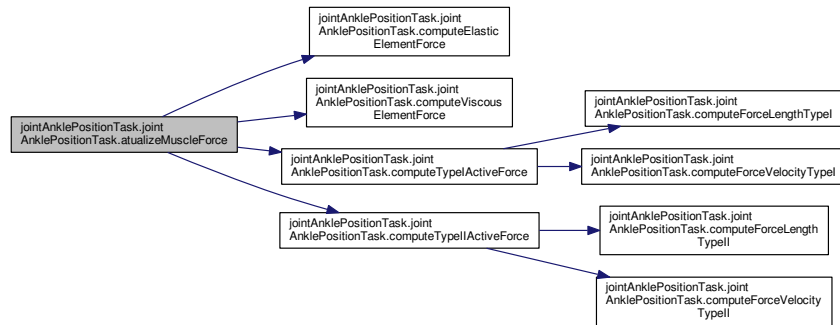
Here is the caller graph for this function:



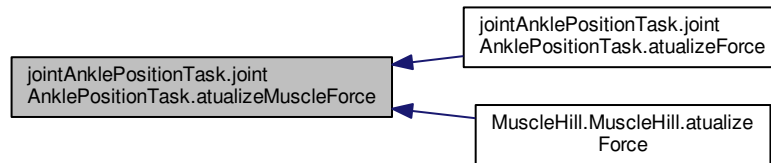
8.8.3.4 `def jointAnklePositionTask.jointAnklePositionTask.atualizeMuscleForce (self)`

Definition at line 217 of file jointAnklePositionTask.py.

Here is the call graph for this function:

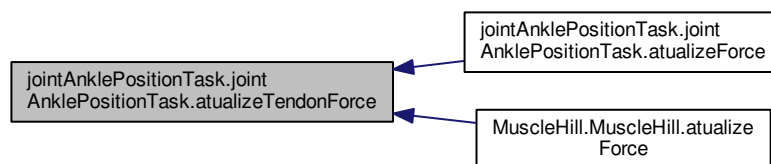


Here is the caller graph for this function:

8.8.3.5 `def jointAnklePositionTask.jointAnklePositionTask.atualizeTendonForce (self)`

Definition at line 221 of file jointAnklePositionTask.py.

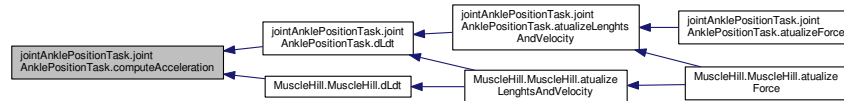
Here is the caller graph for this function:



8.8.3.6 def jointAnklePositionTask.jointAnklePositionTask.computeAcceleration (self)

Definition at line 209 of file jointAnklePositionTask.py.

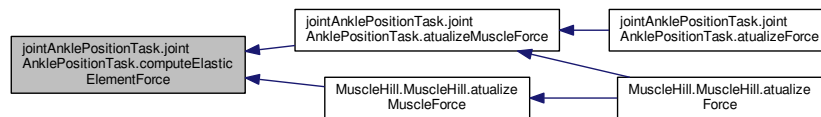
Here is the caller graph for this function:



8.8.3.7 def jointAnklePositionTask.jointAnklePositionTask.computeElasticElementForce (self)

Definition at line 225 of file jointAnklePositionTask.py.

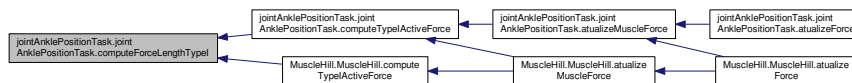
Here is the caller graph for this function:



8.8.3.8 def jointAnklePositionTask.jointAnklePositionTask.computeForceLengthTypel (self)

Definition at line 189 of file jointAnklePositionTask.py.

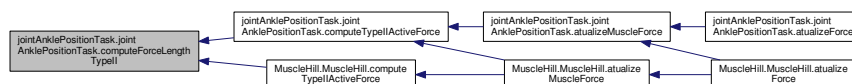
Here is the caller graph for this function:



8.8.3.9 def jointAnklePositionTask.jointAnklePositionTask.computeForceLengthTypell (self)

Definition at line 192 of file jointAnklePositionTask.py.

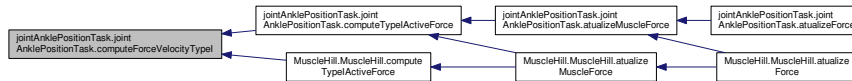
Here is the caller graph for this function:



8.8.3.10 `def jointAnklePositionTask.jointAnklePositionTask.computeForceVelocityTypel (self)`

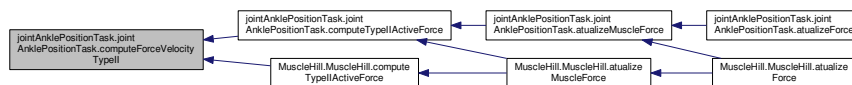
Definition at line 195 of file jointAnklePositionTask.py.

Here is the caller graph for this function:

8.8.3.11 `def jointAnklePositionTask.jointAnklePositionTask.computeForceVelocityTypeell (self)`

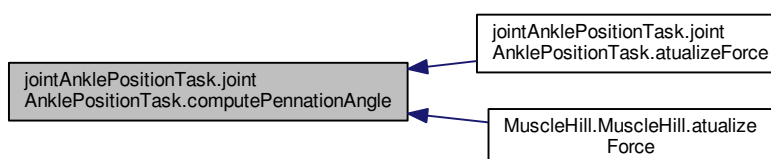
Definition at line 202 of file jointAnklePositionTask.py.

Here is the caller graph for this function:

8.8.3.12 `def jointAnklePositionTask.jointAnklePositionTask.computePennationAngle (self)`

Definition at line 186 of file jointAnklePositionTask.py.

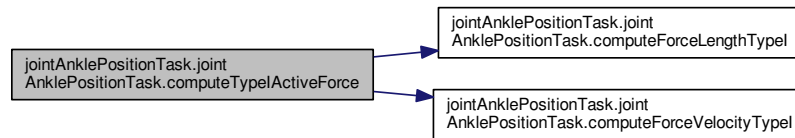
Here is the caller graph for this function:



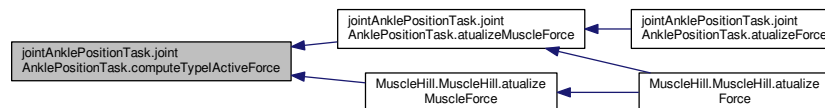
8.8.3.13 `def jointAnklePositionTask.jointAnklePositionTask.computeTypeIActiveForce (self)`

Definition at line 231 of file jointAnklePositionTask.py.

Here is the call graph for this function:



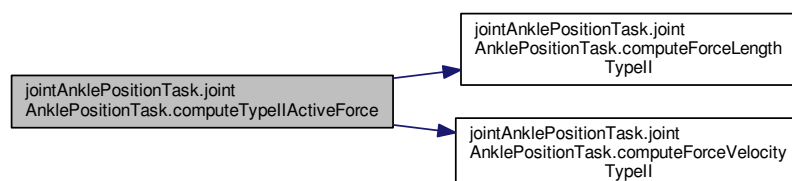
Here is the caller graph for this function:



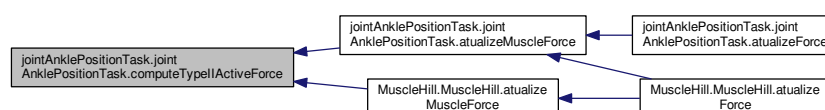
8.8.3.14 `def jointAnklePositionTask.jointAnklePositionTask.computeTypeIIActiveForce (self)`

Definition at line 234 of file jointAnklePositionTask.py.

Here is the call graph for this function:



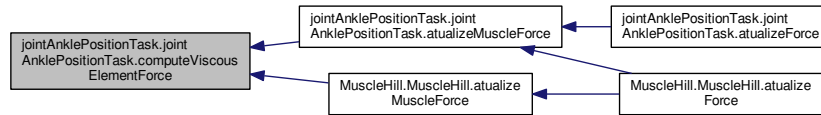
Here is the caller graph for this function:



8.8.3.15 `def jointAnklePositionTask.jointAnklePositionTask.computeViscousElementForce (self)`

Definition at line 228 of file jointAnklePositionTask.py.

Here is the caller graph for this function:

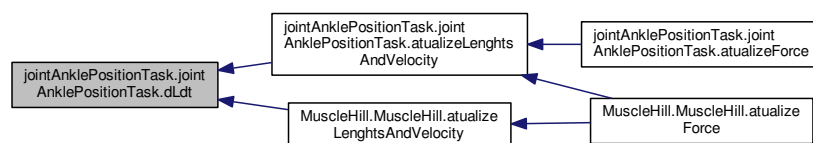
8.8.3.16 `def jointAnklePositionTask.jointAnklePositionTask.dLdt (self)`

Definition at line 214 of file jointAnklePositionTask.py.

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.4 Member Data Documentation

8.8.4.1 `jointAnklePositionTask.jointAnklePositionTask.a0_TypeI`

Definition at line 91 of file jointAnklePositionTask.py.

8.8.4.2 `jointAnklePositionTask.jointAnklePositionTask.a0_TypeII`

Definition at line 93 of file jointAnklePositionTask.py.

8.8.4.3 `jointAnklePositionTask.jointAnklePositionTask.a1_TypeI`

Definition at line 94 of file `jointAnklePositionTask.py`.

8.8.4.4 `jointAnklePositionTask.jointAnklePositionTask.a1_TypeII`

Definition at line 96 of file `jointAnklePositionTask.py`.

8.8.4.5 `jointAnklePositionTask.jointAnklePositionTask.a2_TypeI`

Definition at line 97 of file `jointAnklePositionTask.py`.

8.8.4.6 `jointAnklePositionTask.jointAnklePositionTask.a2_TypeII`

Definition at line 99 of file `jointAnklePositionTask.py`.

8.8.4.7 `jointAnklePositionTask.jointAnklePositionTask.activationTypeI`

Definition at line 45 of file `jointAnklePositionTask.py`.

8.8.4.8 `jointAnklePositionTask.jointAnklePositionTask.activationTypeII`

Definition at line 46 of file `jointAnklePositionTask.py`.

8.8.4.9 `jointAnklePositionTask.jointAnklePositionTask.b_TypeI`

Definition at line 77 of file `jointAnklePositionTask.py`.

8.8.4.10 `jointAnklePositionTask.jointAnklePositionTask.b_TypeII`

Definition at line 79 of file `jointAnklePositionTask.py`.

8.8.4.11 `jointAnklePositionTask.jointAnklePositionTask.c0_TypeI`

Definition at line 100 of file `jointAnklePositionTask.py`.

8.8.4.12 `jointAnklePositionTask.jointAnklePositionTask.c0_TypeII`

Definition at line 102 of file `jointAnklePositionTask.py`.

8.8.4.13 jointAnklePositionTask.jointAnklePositionTask.c1_Typel

Definition at line 103 of file jointAnklePositionTask.py.

8.8.4.14 jointAnklePositionTask.jointAnklePositionTask.c1_Typell

Definition at line 105 of file jointAnklePositionTask.py.

8.8.4.15 jointAnklePositionTask.jointAnklePositionTask.conf

Definition at line 16 of file jointAnklePositionTask.py.

8.8.4.16 jointAnklePositionTask.jointAnklePositionTask.contractileForce_N

Definition at line 38 of file jointAnklePositionTask.py.

8.8.4.17 jointAnklePositionTask.jointAnklePositionTask.d_Typel

Definition at line 88 of file jointAnklePositionTask.py.

8.8.4.18 jointAnklePositionTask.jointAnklePositionTask.d_Typell

Definition at line 90 of file jointAnklePositionTask.py.

8.8.4.19 jointAnklePositionTask.jointAnklePositionTask.elasticForce_N

Definition at line 39 of file jointAnklePositionTask.py.

8.8.4.20 jointAnklePositionTask.jointAnklePositionTask.elasticity

Definition at line 53 of file jointAnklePositionTask.py.

8.8.4.21 jointAnklePositionTask.jointAnklePositionTask.force

Muscle force along time, in N.

Definition at line 36 of file jointAnklePositionTask.py.

8.8.4.22 jointAnklePositionTask.jointAnklePositionTask.forceNorm

Definition at line 73 of file jointAnklePositionTask.py.

8.8.4.23 jointAnklePositionTask.jointAnklePositionTask.length_m

Definition at line 41 of file jointAnklePositionTask.py.

8.8.4.24 jointAnklePositionTask.jointAnklePositionTask.lengthNorm

Definition at line 68 of file jointAnklePositionTask.py.

8.8.4.25 jointAnklePositionTask.jointAnklePositionTask.mass

Definition at line 59 of file jointAnklePositionTask.py.

8.8.4.26 jointAnklePositionTask.jointAnklePositionTask.maximumActivationForce

This is used for normalization purposes.

It is the maximum force that the muscle reach when the Hill model is not used.

Definition at line 34 of file jointAnklePositionTask.py.

8.8.4.27 jointAnklePositionTask.jointAnklePositionTask.maximumForce_N

Maximum force of the Hill model, in N.

Definition at line 50 of file jointAnklePositionTask.py.

8.8.4.28 jointAnklePositionTask.jointAnklePositionTask.MUnumber

Definition at line 18 of file jointAnklePositionTask.py.

8.8.4.29 jointAnklePositionTask.jointAnklePositionTask.MUtypeInumber

Definition at line 19 of file jointAnklePositionTask.py.

8.8.4.30 jointAnklePositionTask.jointAnklePositionTask.optimalLength_m

Definition at line 47 of file jointAnklePositionTask.py.

8.8.4.31 jointAnklePositionTask.jointAnklePositionTask.optimalTendonLength

Definition at line 65 of file jointAnklePositionTask.py.

8.8.4.32 jointAnklePositionTask.jointAnklePositionTask.p_TypeI

Definition at line 81 of file jointAnklePositionTask.py.

8.8.4.33 jointAnklePositionTask.jointAnklePositionTask.p_TypeII

Definition at line 83 of file jointAnklePositionTask.py.

8.8.4.34 jointAnklePositionTask.jointAnklePositionTask.pennationAngle_rad

Definition at line 44 of file jointAnklePositionTask.py.

8.8.4.35 jointAnklePositionTask.jointAnklePositionTask.pennationAngleAtOptimalLengthSin

Definition at line 48 of file jointAnklePositionTask.py.

8.8.4.36 jointAnklePositionTask.jointAnklePositionTask.pool

Definition at line 17 of file jointAnklePositionTask.py.

8.8.4.37 jointAnklePositionTask.jointAnklePositionTask.strain

Definition at line 55 of file jointAnklePositionTask.py.

8.8.4.38 jointAnklePositionTask.jointAnklePositionTask.tendonCurvatureConstant

Definition at line 64 of file jointAnklePositionTask.py.

8.8.4.39 jointAnklePositionTask.jointAnklePositionTask.tendonElasticity

Definition at line 62 of file jointAnklePositionTask.py.

8.8.4.40 jointAnklePositionTask.jointAnklePositionTask.tendonForce_N

Definition at line 37 of file jointAnklePositionTask.py.

8.8.4.41 jointAnklePositionTask.jointAnklePositionTask.tendonForceNorm

Definition at line 74 of file jointAnklePositionTask.py.

8.8.4.42 jointAnklePositionTask.jointAnklePositionTask.tendonLength_m

Definition at line 43 of file jointAnklePositionTask.py.

8.8.4.43 jointAnklePositionTask.jointAnklePositionTask.tendonLengthNorm

Definition at line 72 of file jointAnklePositionTask.py.

8.8.4.44 jointAnklePositionTask.jointAnklePositionTask.tendonLinearOnsetLength

Definition at line 63 of file jointAnklePositionTask.py.

8.8.4.45 jointAnklePositionTask.jointAnklePositionTask.timeIndex

Definition at line 21 of file jointAnklePositionTask.py.

8.8.4.46 jointAnklePositionTask.jointAnklePositionTask.twitchAmp_N

Amplitude of the muscle unit twitch, in N (see atualizeForce function explanation).

Definition at line 26 of file jointAnklePositionTask.py.

8.8.4.47 jointAnklePositionTask.jointAnklePositionTask.twTet

Twitch-tetanus relationship (see atualizeForce function explanation)

Definition at line 24 of file jointAnklePositionTask.py.

8.8.4.48 jointAnklePositionTask.jointAnklePositionTask.velocity_m_ms

Definition at line 42 of file jointAnklePositionTask.py.

8.8.4.49 jointAnklePositionTask.jointAnklePositionTask.velocityNorm

Definition at line 70 of file jointAnklePositionTask.py.

8.8.4.50 jointAnklePositionTask.jointAnklePositionTask.viscosity

Definition at line 57 of file jointAnklePositionTask.py.

8.8.4.51 jointAnklePositionTask.jointAnklePositionTask.viscousForce_N

Definition at line 40 of file jointAnklePositionTask.py.

8.8.4.52 jointAnklePositionTask.jointAnklePositionTask.Vmax_Typel

Definition at line 106 of file jointAnklePositionTask.py.

8.8.4.53 jointAnklePositionTask.jointAnklePositionTask.Vmax_Typell

Definition at line 108 of file jointAnklePositionTask.py.

8.8.4.54 jointAnklePositionTask.jointAnklePositionTask.w_Typel

Definition at line 85 of file jointAnklePositionTask.py.

8.8.4.55 jointAnklePositionTask.jointAnklePositionTask.w_Typell

Definition at line 87 of file jointAnklePositionTask.py.

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

- [jointAnklePositionTask.py](#)

8.9 MotorUnit.MotorUnit Class Reference

Class that implements a motor unit model.

Public Member Functions

- def `__init__` (self, [conf](#), pool, [index](#), [kind](#))
Constructor.
- def [atualizeMotorUnit](#) (self, t)
Atualize the dynamical and nondynamical (delay) parts of the motor unit.
- def [atualizeCompartments](#) (self, t)
Atualize all neural compartments.
- def [dVdt](#) (self, t, V)
Compute the potential derivative of all compartments of the motor unit.
- def [addSomaSpike](#) (self, t)
When the soma potential is above the threshold a spike is added tom the soma.
- def [atualizeDelay](#) (self, t)
Atualize the terminal spike train, by considering the Delay of the nerve.
- def [transmitSpikes](#) (self, t)

Public Attributes

- [conf](#)
Configuration object with the simulation parameters.
- [kind](#)
String with the type of the motor unit.
- [tSomaSpike](#)
The instant of the last spike of the Motor unit at the Soma compartment.
- [somaSpikeTrain](#)
Vector with the instants of spikes at the soma.
- [index](#)
Integer corresponding to the motor unit order in the pool, according to the Henneman's principle (size principle).
- [compartment](#)
Vector of [Compartment](#) of the Motor Unit.
- [threshold_mV](#)
Value of the membrane potential, in mV, that is considered a spike.
- [position_mm](#)
Anatomical position of the neuron, in mm.
- [compNumber](#)
Number of compartments.
- [v_mV](#)
Vector with membrane potential, in mV, of all compartments.
- [capacitanceInv](#)
Vector with the inverse of the capacitance of all compartments.
- [ilonic](#)
Vector with current, in nA, of each compartment coming from other elements of the model.
- [iInjected](#)
Vector with the current, in nA, injected in each compartment.
- [G](#)
Matrix of the conductance of the motoneuron.
- [somalIndex](#)
index of the soma compartment.
- [MNRefPer_ms](#)
Refractory period, in ms, of the motoneuron.
- [nerve](#)
String with type of the nerve.
- [Delay](#)
[AxonDelay](#) object of the motor unit.
- [terminalSpikeTrain](#)
Vector with the instants of spikes at the terminal.
- [TwitchTc_ms](#)
Contraction time of the twitch muscle unit, in ms.
- [TwitchAmp_N](#)
Amplitude of the muscle unit twitch, in N.
- [bSat](#)
Parameter of the saturation.
- [twTet](#)
Twitch- tetanus relationship.
- [SynapsesOut](#)
EMG data.
- [transmitSpikesThroughSynapses](#)
- [indicesOfSynapsesOnTarget](#)

8.9.1 Detailed Description

Class that implements a motor unit model.

Encompasses a motoneuron and a muscle unit.

Definition at line 148 of file MotorUnit.py.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 `def MotorUnit.MotorUnit.__init__(self, conf, pool, index, kind)`

Constructor.

- Inputs:
 - **conf**: [Configuration](#) object with the simulation parameters.

cyto + **pool**: string with Motor unit pool to which the motor unit belongs.

- **index**: integer corresponding to the motor unit order in the pool, according to the Henneman's principle (size principle).
- **kind**: string with the type of the motor unit. It can be *S* (slow), *FR* (fast and resistant), and *FF* (fast and fatigable).

Definition at line 167 of file MotorUnit.py.

8.9.3 Member Function Documentation

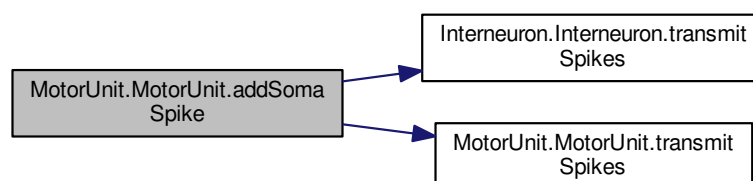
8.9.3.1 `def MotorUnit.MotorUnit.addSomaSpike(self, t)`

When the soma potential is above the threshold a spike is added tom the soma.

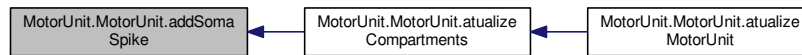
- Inputs:
 - **t**: current instant, in ms.

Definition at line 335 of file MotorUnit.py.

Here is the call graph for this function:



Here is the caller graph for this function:



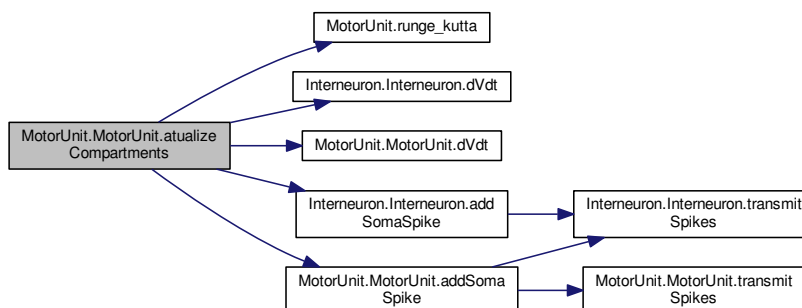
8.9.3.2 `def MotorUnit.MotorUnit.atualizeCompartments (self, t)`

Atualize all neural compartments.

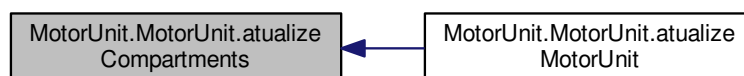
- Inputs:
 - `t`: current instant, in ms.

Definition at line 298 of file `MotorUnit.py`.

Here is the call graph for this function:



Here is the caller graph for this function:



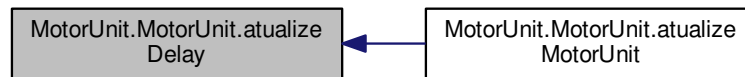
8.9.3.3 `def MotorUnit.MotorUnit.atualizeDelay (self, t)`

Atualize the terminal spike train, by considering the Delay of the nerve.

- Inputs:
 - `t`: current instant, in ms.

Definition at line 352 of file MotorUnit.py.

Here is the caller graph for this function:



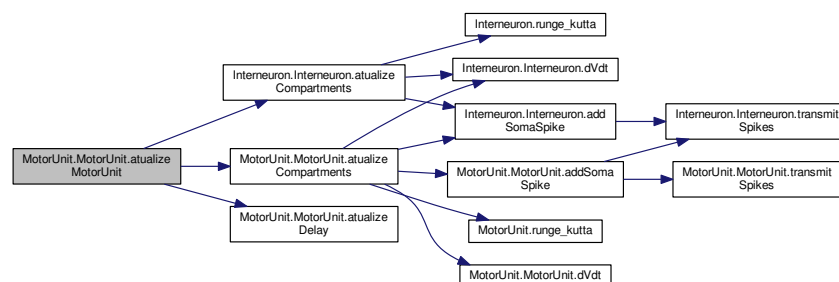
8.9.3.4 `def MotorUnit.MotorUnit.atualizeMotorUnit (self, t)`

Atualize the dynamical and nondynamical (delay) parts of the motor unit.

- Inputs:
 - `t`: current instant, in ms.

Definition at line 286 of file MotorUnit.py.

Here is the call graph for this function:



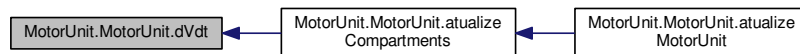
8.9.3.5 `def MotorUnit.MotorUnit.dVdt (self, t, V)`

Compute the potential derivative of all compartments of the motor unit.

- Inputs:
 - **t**: current instant, in ms.
 - **V**: Vector with the current potential value of all neural compartments of the motor unit.

Definition at line 321 of file MotorUnit.py.

Here is the caller graph for this function:

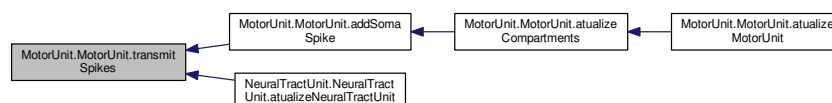


8.9.3.6 `def MotorUnit.MotorUnit.transmitSpikes (self, t)`

- Inputs:
 - **t**: current instant, in ms.

Definition at line 362 of file MotorUnit.py.

Here is the caller graph for this function:



8.9.4 Member Data Documentation

8.9.4.1 `MotorUnit.MotorUnit.bSat`

Parameter of the saturation.

Definition at line 267 of file MotorUnit.py.

8.9.4.2 MotorUnit.MotorUnit.capacitanceInv

Vector with the inverse of the capacitance of all compartments.

Definition at line 219 of file MotorUnit.py.

8.9.4.3 MotorUnit.MotorUnit.compartment

Vector of [Compartment](#) of the Motor Unit.

Definition at line 187 of file MotorUnit.py.

8.9.4.4 MotorUnit.MotorUnit.compNumber

Number of compartments.

Definition at line 197 of file MotorUnit.py.

8.9.4.5 MotorUnit.MotorUnit.conf

[Configuration](#) object with the simulation parameters.

Definition at line 170 of file MotorUnit.py.

8.9.4.6 MotorUnit.MotorUnit.Delay

[AxonDelay](#) object of the motor unit.

Definition at line 252 of file MotorUnit.py.

8.9.4.7 MotorUnit.MotorUnit.G

Matrix of the conductance of the motoneuron.

Multiplied by the vector `self.v_mV`, results in the passive currents of each compartment.

Definition at line 234 of file MotorUnit.py.

8.9.4.8 MotorUnit.MotorUnit.injected

Vector with the current, in nA, injected in each compartment.

Definition at line 225 of file MotorUnit.py.

8.9.4.9 MotorUnit.MotorUnit.ionic

Vector with current, in nA, of each compartment coming from other elements of the model.

For example from ionic channels and synapses.

Definition at line 223 of file MotorUnit.py.

8.9.4.10 MotorUnit.MotorUnit.index

Integer corresponding to the motor unit order in the pool, according to the Henneman's principle (size principle).

Definition at line 185 of file MotorUnit.py.

8.9.4.11 MotorUnit.MotorUnit.indicesOfSynapsesOnTarget

Definition at line 277 of file MotorUnit.py.

8.9.4.12 MotorUnit.MotorUnit.kind

String with the type of the motor unit.

It can be *S* (slow), *FR* (fast and resistant) and **FF** (fast and fatigable).

Definition at line 175 of file MotorUnit.py.

8.9.4.13 MotorUnit.MotorUnit.MNRefPer_ms

Refractory period, in ms, of the motoneuron.

Definition at line 241 of file MotorUnit.py.

8.9.4.14 MotorUnit.MotorUnit.nerve

String with type of the nerve.

It can be PTN (posterior tibial nerve) or CPN (common peroneal nerve).

Definition at line 247 of file MotorUnit.py.

8.9.4.15 MotorUnit.MotorUnit.position_mm

Anatomical position of the neuron, in mm.

Definition at line 192 of file MotorUnit.py.

8.9.4.16 MotorUnit.MotorUnit.somaIndex

index of the soma compartment.

Definition at line 238 of file MotorUnit.py.

8.9.4.17 MotorUnit.MotorUnit.somaSpikeTrain

Vector with the instants of spikes at the soma.

Definition at line 183 of file MotorUnit.py.

8.9.4.18 MotorUnit.MotorUnit.SynapsesOut

EMG data.

Build synapses

Definition at line 275 of file MotorUnit.py.

8.9.4.19 MotorUnit.MotorUnit.terminalSpikeTrain

Vector with the instants of spikes at the terminal.

Definition at line 256 of file MotorUnit.py.

8.9.4.20 MotorUnit.MotorUnit.threshold_mV

Value of the membrane potential, in mV, that is considered a spike.

Definition at line 189 of file MotorUnit.py.

8.9.4.21 MotorUnit.MotorUnit.transmitSpikesThroughSynapses

Definition at line 276 of file MotorUnit.py.

8.9.4.22 MotorUnit.MotorUnit.tSomaSpike

The instant of the last spike of the Motor unit at the Soma compartment.

Definition at line 180 of file MotorUnit.py.

8.9.4.23 MotorUnit.MotorUnit.TwitchAmp_N

Amplitude of the muscle unit twitch, in N.

Definition at line 265 of file MotorUnit.py.

8.9.4.24 MotorUnit.MotorUnit.TwitchTc_ms

Contraction time of the twitch muscle unit, in ms.

Definition at line 263 of file MotorUnit.py.

8.9.4.25 MotorUnit.MotorUnit.twTet

Twitch- tetanus relationship.

Definition at line 269 of file MotorUnit.py.

8.9.4.26 MotorUnit.MotorUnit.v_mV

Vector with membrane potential,in mV, of all compartments.

Definition at line 199 of file MotorUnit.py.

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

- [MotorUnit.py](#)

8.10 MotorUnitPool.MotorUnitPool Class Reference

Class that implements a motor unit pool.

Public Member Functions

- `def __init__(self, conf, pool)`
Constructor.
- `def atualizeMotorUnitPool(self, t)`
Update all parts of the Motor Unit pool.
- `def listSpikes(self)`
List the spikes that occurred in the soma and in the terminal of the different motor units.

Public Attributes

- `kind`
Indicates that is Motor Unit pool.
- `conf`
Configuration object with the simulation parameters.
- `pool`
String with Motor unit pool to which the motor unit belongs.
- `MUnumber`
Number of motor units.
- `unit`
List of MotorUnit objects.
- `poolSomaSpikes`
Vector with the instants of spikes in the soma compartment, in ms.
- `poolTerminalSpikes`
Vector with the instants of spikes in the terminal, in ms.
- `Activation`
- `hillModel`
String indicating whther a Hill model is used or not.
- `Muscle`

8.10.1 Detailed Description

Class that implements a motor unit pool.

Encompasses a set of motor units that controls a single muscle.

Definition at line 26 of file MotorUnitPool.py.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 `def MotorUnitPool.MotorUnitPool.__init__(self, conf, pool)`

Constructor.

- Inputs:
 - **conf**: [Configuration](#) object with the simulation parameters.
 - **pool**: string with Motor unit pool to which the motor unit belongs.

Definition at line 38 of file MotorUnitPool.py.

8.10.3 Member Function Documentation

8.10.3.1 `def MotorUnitPool.MotorUnitPool.atualizeMotorUnitPool(self, t)`

Update all parts of the Motor Unit pool.

It consists to update all motor units, the activation signal and the muscle force.

- Inputs:
 - **t**: current instant, in ms.

Definition at line 93 of file MotorUnitPool.py.

8.10.3.2 `def MotorUnitPool.MotorUnitPool.listSpikes(self)`

List the spikes that occurred in the soma and in the terminal of the different motor units.

Definition at line 104 of file MotorUnitPool.py.

8.10.4 Member Data Documentation

8.10.4.1 `MotorUnitPool.MotorUnitPool.Activation`

Definition at line 71 of file MotorUnitPool.py.

8.10.4.2 `MotorUnitPool.MotorUnitPool.conf`

[Configuration](#) object with the simulation parameters.

Definition at line 44 of file `MotorUnitPool.py`.

8.10.4.3 `MotorUnitPool.MotorUnitPool.hillModel`

String indicating whether a Hill model is used or not.

For now, it can be *No*.

Definition at line 75 of file `MotorUnitPool.py`.

8.10.4.4 `MotorUnitPool.MotorUnitPool.kind`

Indicates that is Motor Unit pool.

Definition at line 41 of file `MotorUnitPool.py`.

8.10.4.5 `MotorUnitPool.MotorUnitPool.MUnumber`

Number of motor units.

Definition at line 51 of file `MotorUnitPool.py`.

8.10.4.6 `MotorUnitPool.MotorUnitPool.Muscle`

Definition at line 77 of file `MotorUnitPool.py`.

8.10.4.7 `MotorUnitPool.MotorUnitPool.pool`

String with Motor unit pool to which the motor unit belongs.

Definition at line 46 of file `MotorUnitPool.py`.

8.10.4.8 `MotorUnitPool.MotorUnitPool.poolSomaSpikes`

Vector with the instants of spikes in the soma compartment, in ms.

Definition at line 66 of file `MotorUnitPool.py`.

8.10.4.9 `MotorUnitPool.MotorUnitPool.poolTerminalSpikes`

Vector with the instants of spikes in the terminal, in ms.

Definition at line 68 of file `MotorUnitPool.py`.

8.10.4.10 MotorUnitPool.MotorUnitPool.unit

List of [MotorUnit](#) objects.

Definition at line 54 of file MotorUnitPool.py.

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

- [MotorUnitPool.py](#)

8.11 MuscleHill.MuscleHill Class Reference

Public Member Functions

- def [__init__](#) (self, [conf](#), [pool](#), [MUnumber](#), [MUtypelnumber](#), unit)
- def [atualizeForce](#) (self, activation_Sat)
Compute the muscle force when no muscle dynamics (Hill model) is used.
- def [atualizeActivation](#) (self, activation_Sat)
- def [computePennationAngle](#) (self)
- def [computeForceLengthTypeI](#) (self)
- def [computeForceLengthTypeII](#) (self)
- def [computeForceVelocityTypeI](#) (self)
- def [computeForceVelocityTypeII](#) (self)
- def [computeAcceleration](#) (self)
- def [dLdt](#) (self)
- def [atualizeMuscleForce](#) (self)
- def [atualizeTendonForce](#) (self)
- def [computeElasticElementForce](#) (self)
- def [computeViscousElementForce](#) (self)
- def [computeTypeIActiveForce](#) (self)
- def [computeTypeIIActiveForce](#) (self)
- def [atualizeLenghtsAndVelocity](#) (self)
- def [atualizeMusculoTendonLength](#) (self, ankleAngle)
- def [atualizeMomentArm](#) (self, ankleAngle)

Public Attributes

- [conf](#)
- [pool](#)
- [MUnumber](#)
- [MUtypelnumber](#)
- [timeIndex](#)
- [twTet](#)
Twitch-tetanus relationship (see [atualizeForce](#) function explanation)
- [twitchAmp_N](#)
Amplitude of the muscle unit twitch, in N (see [atualizeForce](#) function explanation).
- [maximumActivationForce](#)
This is used for normalization purposes.
- [force](#)
Muscle force along time, in N.

- [tendonForce_N](#)
- [contractileForce_N](#)
- [elasticForce_N](#)
- [viscousForce_N](#)
- [length_m](#)
- [velocity_m_ms](#)
- [tendonLength_m](#)
- [pennationAngle_rad](#)
- [activationTypeI](#)
- [activationTypeII](#)
- [musculoTendonLength_m](#)
- [momentArm_m](#)
- [optimalLength_m](#)
- [pennationAngleAtOptimalLengthSin](#)
- [maximumForce_N](#)

Maximum force of the Hill model, in N.

- [elasticity](#)
- [strain](#)
- [viscosity](#)
- [mass](#)
- [tendonElasticity](#)
- [tendonLinearOnsetLength](#)
- [tendonCurvatureConstant](#)
- [optimalTendonLength](#)
- [lengthNorm](#)
- [velocityNorm](#)
- [tendonLengthNorm](#)
- [forceNorm](#)
- [tendonForceNorm](#)
- [b_TypeI](#)
- [b_TypeII](#)
- [p_TypeI](#)
- [p_TypeII](#)
- [w_TypeI](#)
- [w_TypeII](#)
- [d_TypeI](#)
- [d_TypeII](#)
- [a0_TypeI](#)
- [a0_TypeII](#)
- [a1_TypeI](#)
- [a1_TypeII](#)
- [a2_TypeI](#)
- [a2_TypeII](#)
- [c0_TypeI](#)
- [c0_TypeII](#)
- [c1_TypeI](#)
- [c1_TypeII](#)
- [Vmax_TypeI](#)
- [Vmax_TypeII](#)
- [m0](#)
- [m1](#)
- [m2](#)
- [m3](#)
- [m4](#)

- [n0](#)
- [n1](#)
- [n2](#)
- [n3](#)
- [n4](#)

8.11.1 Detailed Description

Definition at line 12 of file MuscleHill.py.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 `def MuscleHill.MuscleHill.__init__(self, conf, pool, MUnumber, MUnumber, unit)`

Definition at line 14 of file MuscleHill.py.

8.11.3 Member Function Documentation

8.11.3.1 `def MuscleHill.MuscleHill.atualizeActivation(self, activation_Sat)`

Definition at line 205 of file MuscleHill.py.

Here is the caller graph for this function:



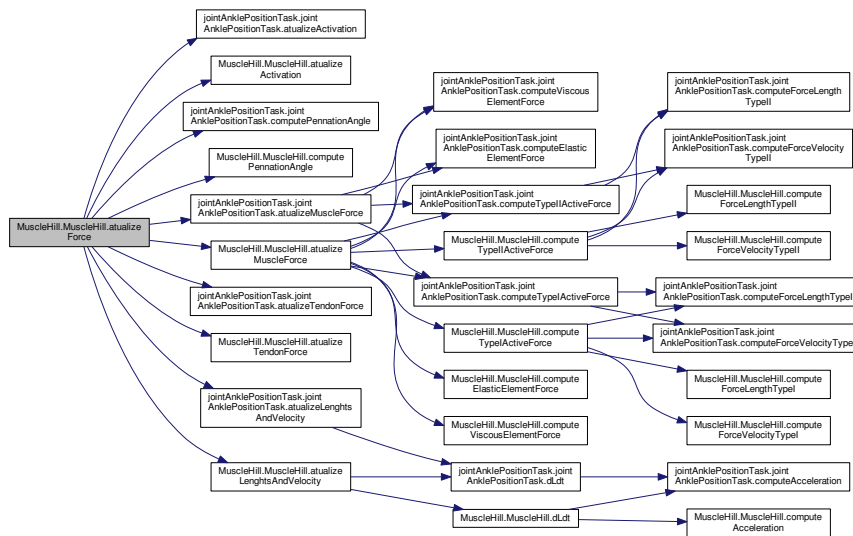
8.11.3.2 `def MuscleHill.MuscleHill.atualizeForce(self, activation_Sat)`

Compute the muscle force when no muscle dynamics (Hill model) is used.

This operation is vectorized. Each element of the vectors correspond to one motor unit. For each motor unit, the force is computed by the following formula:

Definition at line 185 of file MuscleHill.py.

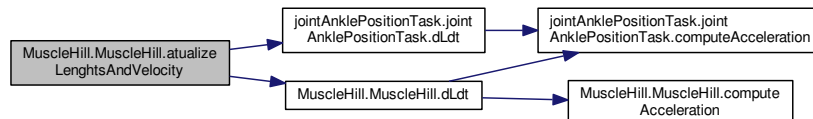
Here is the call graph for this function:



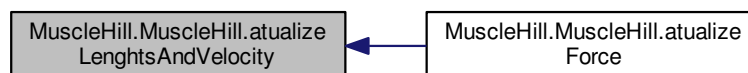
8.11.3.3 def MuscleHill.MuscleHill.atualizeLengthsAndVelocity (self)

Definition at line 266 of file MuscleHill.py.

Here is the call graph for this function:



Here is the caller graph for this function:



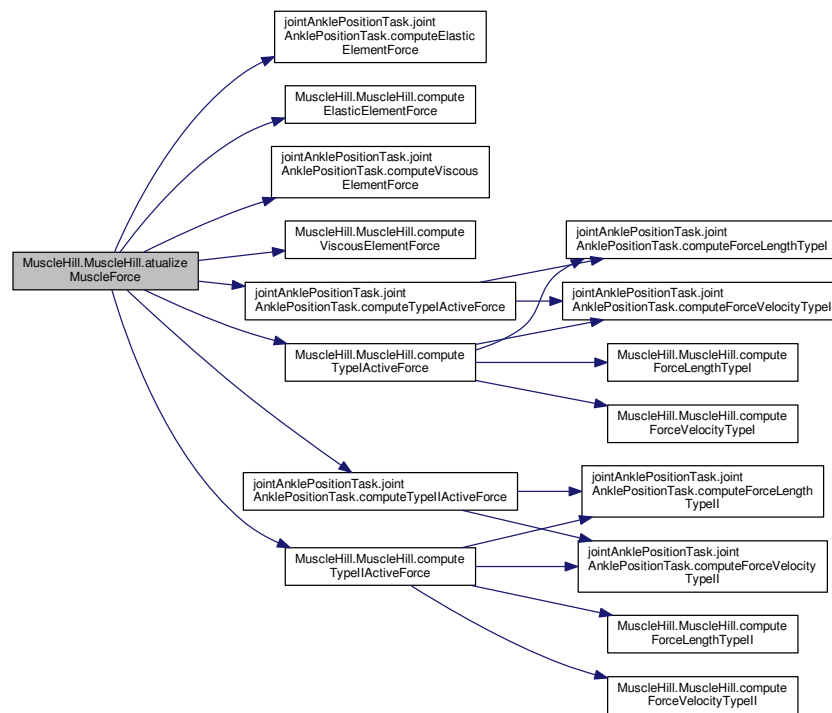
8.11.3.4 def MuscleHill.MuscleHill.atualizeMomentArm (self, ankleAngle)

Definition at line 277 of file MuscleHill.py.

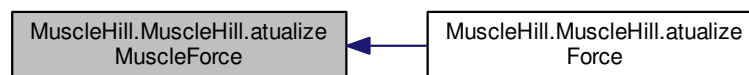
8.11.3.5 def MuscleHill.MuscleHill.atualizeMuscleForce (self)

Definition at line 244 of file MuscleHill.py.

Here is the call graph for this function:



Here is the caller graph for this function:



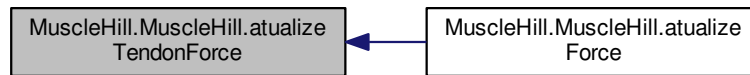
8.11.3.6 def MuscleHill.MuscleHill.atualizeMusculoTendonLength (self, ankleAngle)

Definition at line 270 of file MuscleHill.py.

8.11.3.7 `def MuscleHill.MuscleHill.atualizeTendonForce (self)`

Definition at line 248 of file MuscleHill.py.

Here is the caller graph for this function:



8.11.3.8 `def MuscleHill.MuscleHill.computeAcceleration (self)`

Definition at line 236 of file MuscleHill.py.

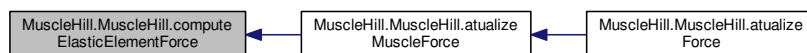
Here is the caller graph for this function:



8.11.3.9 `def MuscleHill.MuscleHill.computeElasticElementForce (self)`

Definition at line 252 of file MuscleHill.py.

Here is the caller graph for this function:



8.11.3.10 `def MuscleHill.MuscleHill.computeForceLengthTypeI (self)`

Definition at line 216 of file MuscleHill.py.

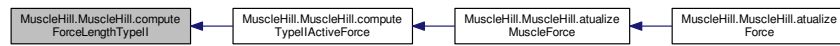
Here is the caller graph for this function:



8.11.3.11 `def MuscleHill.MuscleHill.computeForceLengthTypell (self)`

Definition at line 219 of file MuscleHill.py.

Here is the caller graph for this function:



8.11.3.12 `def MuscleHill.MuscleHill.computeForceVelocityTypel (self)`

Definition at line 222 of file MuscleHill.py.

Here is the caller graph for this function:



8.11.3.13 `def MuscleHill.MuscleHill.computeForceVelocityTypell (self)`

Definition at line 229 of file MuscleHill.py.

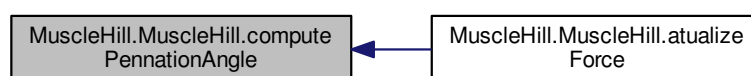
Here is the caller graph for this function:



8.11.3.14 `def MuscleHill.MuscleHill.computePennationAngle (self)`

Definition at line 213 of file MuscleHill.py.

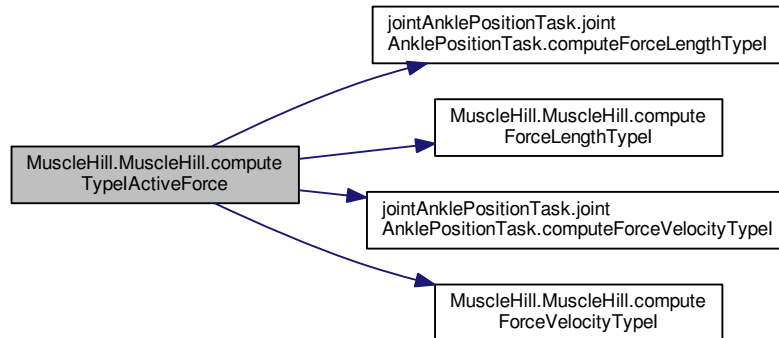
Here is the caller graph for this function:



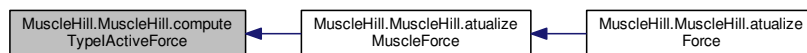
8.11.3.15 `def MuscleHill.MuscleHill.computeTypeIActiveForce (self)`

Definition at line 258 of file MuscleHill.py.

Here is the call graph for this function:



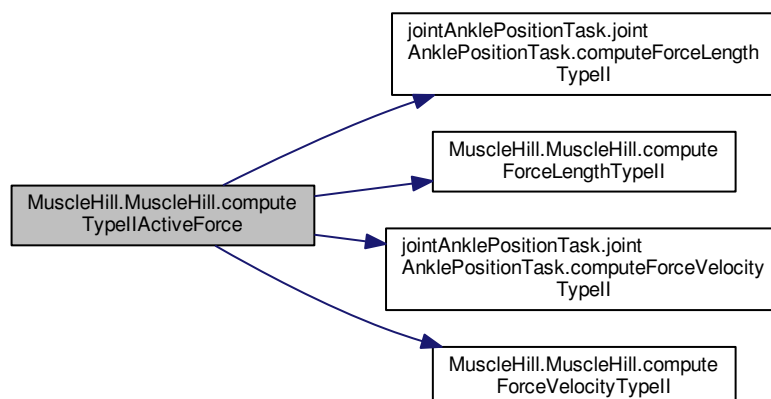
Here is the caller graph for this function:



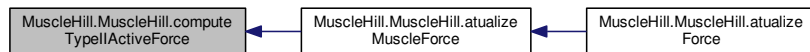
8.11.3.16 `def MuscleHill.MuscleHill.computeTypeIIActiveForce (self)`

Definition at line 262 of file MuscleHill.py.

Here is the call graph for this function:



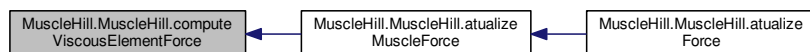
Here is the caller graph for this function:



8.11.3.17 `def MuscleHill.MuscleHill.computeViscousElementForce (self)`

Definition at line 255 of file `MuscleHill.py`.

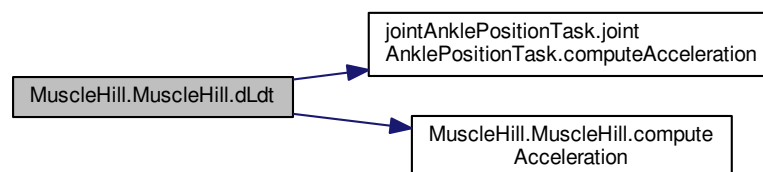
Here is the caller graph for this function:



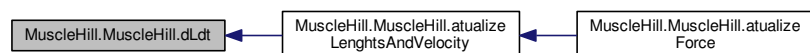
8.11.3.18 `def MuscleHill.MuscleHill.dLdt (self)`

Definition at line 241 of file `MuscleHill.py`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.11.4 Member Data Documentation

8.11.4.1 MuscleHill.MuscleHill.a0_Typel

Definition at line 93 of file MuscleHill.py.

8.11.4.2 MuscleHill.MuscleHill.a0_Typell

Definition at line 95 of file MuscleHill.py.

8.11.4.3 MuscleHill.MuscleHill.a1_Typel

Definition at line 96 of file MuscleHill.py.

8.11.4.4 MuscleHill.MuscleHill.a1_Typell

Definition at line 98 of file MuscleHill.py.

8.11.4.5 MuscleHill.MuscleHill.a2_Typel

Definition at line 99 of file MuscleHill.py.

8.11.4.6 MuscleHill.MuscleHill.a2_Typell

Definition at line 101 of file MuscleHill.py.

8.11.4.7 MuscleHill.MuscleHill.activationTypel

Definition at line 45 of file MuscleHill.py.

8.11.4.8 MuscleHill.MuscleHill.activationTypell

Definition at line 46 of file MuscleHill.py.

8.11.4.9 MuscleHill.MuscleHill.b_Typel

Definition at line 79 of file MuscleHill.py.

8.11.4.10 MuscleHill.MuscleHill.b_Typell

Definition at line 81 of file MuscleHill.py.

8.11.4.11 MuscleHill.MuscleHill.c0_TypeI

Definition at line 102 of file MuscleHill.py.

8.11.4.12 MuscleHill.MuscleHill.c0_TypeII

Definition at line 104 of file MuscleHill.py.

8.11.4.13 MuscleHill.MuscleHill.c1_TypeI

Definition at line 105 of file MuscleHill.py.

8.11.4.14 MuscleHill.MuscleHill.c1_TypeII

Definition at line 107 of file MuscleHill.py.

8.11.4.15 MuscleHill.MuscleHill.conf

Definition at line 16 of file MuscleHill.py.

8.11.4.16 MuscleHill.MuscleHill.contractileForce_N

Definition at line 38 of file MuscleHill.py.

8.11.4.17 MuscleHill.MuscleHill.d_TypeI

Definition at line 90 of file MuscleHill.py.

8.11.4.18 MuscleHill.MuscleHill.d_TypeII

Definition at line 92 of file MuscleHill.py.

8.11.4.19 MuscleHill.MuscleHill.elasticForce_N

Definition at line 39 of file MuscleHill.py.

8.11.4.20 MuscleHill.MuscleHill.elasticity

Definition at line 55 of file MuscleHill.py.

8.11.4.21 MuscleHill.MuscleHill.force

Muscle force along time, in N.

Definition at line 36 of file MuscleHill.py.

8.11.4.22 MuscleHill.MuscleHill.forceNorm

Definition at line 75 of file MuscleHill.py.

8.11.4.23 MuscleHill.MuscleHill.length_m

Definition at line 41 of file MuscleHill.py.

8.11.4.24 MuscleHill.MuscleHill.lengthNorm

Definition at line 70 of file MuscleHill.py.

8.11.4.25 MuscleHill.MuscleHill.m0

Definition at line 112 of file MuscleHill.py.

8.11.4.26 MuscleHill.MuscleHill.m1

Definition at line 114 of file MuscleHill.py.

8.11.4.27 MuscleHill.MuscleHill.m2

Definition at line 116 of file MuscleHill.py.

8.11.4.28 MuscleHill.MuscleHill.m3

Definition at line 118 of file MuscleHill.py.

8.11.4.29 MuscleHill.MuscleHill.m4

Definition at line 120 of file MuscleHill.py.

8.11.4.30 MuscleHill.MuscleHill.mass

Definition at line 61 of file MuscleHill.py.

8.11.4.31 MuscleHill.MuscleHill.maximumActivationForce

This is used for normalization purposes.

It is the maximum force that the muscle reach when the Hill model is not used.

Definition at line 34 of file MuscleHill.py.

8.11.4.32 MuscleHill.MuscleHill.maximumForce_N

Maximum force of the Hill model, in N.

Definition at line 52 of file MuscleHill.py.

8.11.4.33 MuscleHill.MuscleHill.momentArm_m

Definition at line 48 of file MuscleHill.py.

8.11.4.34 MuscleHill.MuscleHill.MUnumber

Definition at line 18 of file MuscleHill.py.

8.11.4.35 MuscleHill.MuscleHill.musculoTendonLength_m

Definition at line 47 of file MuscleHill.py.

8.11.4.36 MuscleHill.MuscleHill.MUtypeInumber

Definition at line 19 of file MuscleHill.py.

8.11.4.37 MuscleHill.MuscleHill.n0

Definition at line 123 of file MuscleHill.py.

8.11.4.38 MuscleHill.MuscleHill.n1

Definition at line 125 of file MuscleHill.py.

8.11.4.39 MuscleHill.MuscleHill.n2

Definition at line 127 of file MuscleHill.py.

8.11.4.40 MuscleHill.MuscleHill.n3

Definition at line 129 of file MuscleHill.py.

8.11.4.41 MuscleHill.MuscleHill.n4

Definition at line 131 of file MuscleHill.py.

8.11.4.42 MuscleHill.MuscleHill.optimalLength_m

Definition at line 49 of file MuscleHill.py.

8.11.4.43 MuscleHill.MuscleHill.optimalTendonLength

Definition at line 67 of file MuscleHill.py.

8.11.4.44 MuscleHill.MuscleHill.p_Type1

Definition at line 83 of file MuscleHill.py.

8.11.4.45 MuscleHill.MuscleHill.p_Type11

Definition at line 85 of file MuscleHill.py.

8.11.4.46 MuscleHill.MuscleHill.pennationAngle_rad

Definition at line 44 of file MuscleHill.py.

8.11.4.47 MuscleHill.MuscleHill.pennationAngleAtOptimalLengthSin

Definition at line 50 of file MuscleHill.py.

8.11.4.48 MuscleHill.MuscleHill.pool

Definition at line 17 of file MuscleHill.py.

8.11.4.49 MuscleHill.MuscleHill.strain

Definition at line 57 of file MuscleHill.py.

8.11.4.50 MuscleHill.MuscleHill.tendonCurvatureConstant

Definition at line 66 of file MuscleHill.py.

8.11.4.51 MuscleHill.MuscleHill.tendonElasticity

Definition at line 64 of file MuscleHill.py.

8.11.4.52 MuscleHill.MuscleHill.tendonForce_N

Definition at line 37 of file MuscleHill.py.

8.11.4.53 MuscleHill.MuscleHill.tendonForceNorm

Definition at line 76 of file MuscleHill.py.

8.11.4.54 MuscleHill.MuscleHill.tendonLength_m

Definition at line 43 of file MuscleHill.py.

8.11.4.55 MuscleHill.MuscleHill.tendonLengthNorm

Definition at line 74 of file MuscleHill.py.

8.11.4.56 MuscleHill.MuscleHill.tendonLinearOnsetLength

Definition at line 65 of file MuscleHill.py.

8.11.4.57 MuscleHill.MuscleHill.timeIndex

Definition at line 21 of file MuscleHill.py.

8.11.4.58 MuscleHill.MuscleHill.twitchAmp_N

Amplitude of the muscle unit twitch, in N (see `atualizeForce` function explanation).

Definition at line 26 of file MuscleHill.py.

8.11.4.59 MuscleHill.MuscleHill.twTet

Twitch-tetanus relationship (see `atualizeForce` function explanation)

Definition at line 24 of file MuscleHill.py.

8.11.4.60 MuscleHill.MuscleHill.velocity_m_ms

Definition at line 42 of file MuscleHill.py.

8.11.4.61 MuscleHill.MuscleHill.velocityNorm

Definition at line 72 of file MuscleHill.py.

8.11.4.62 MuscleHill.MuscleHill.viscosity

Definition at line 59 of file MuscleHill.py.

8.11.4.63 MuscleHill.MuscleHill.viscousForce_N

Definition at line 40 of file MuscleHill.py.

8.11.4.64 MuscleHill.MuscleHill.Vmax_TypeI

Definition at line 108 of file MuscleHill.py.

8.11.4.65 MuscleHill.MuscleHill.Vmax_TypeII

Definition at line 110 of file MuscleHill.py.

8.11.4.66 MuscleHill.MuscleHill.w_TypeI

Definition at line 87 of file MuscleHill.py.

8.11.4.67 MuscleHill.MuscleHill.w_TypeII

Definition at line 89 of file MuscleHill.py.

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

- [MuscleHill.py](#)

8.12 MuscleNoHill.MuscleNoHill Class Reference

Public Member Functions

- def [__init__](#) (self, [conf](#), [pool](#), [MUnumber](#), [MUTypeI](#)number, unit)
- def [atualizeForce](#) (self, activation_Sat)

Compute the muscle force when no muscle dynamics (Hill model) is used.

Public Attributes

- [conf](#)
- [pool](#)
- [MUnumber](#)
- [MUTypeInumber](#)
- [twTet](#)

Twitch- tetanus relationship (see atualizeForceNoHill function explanation)
- [twitchAmp_N](#)

Amplitude of the muscle unit twitch, in N (see atualizeForceNoHill function explanation).
- [maximumActivationForce](#)

This is used for normalization purposes.
- [force](#)

Muscle force along time, in N.
- [timeIndex](#)

8.12.1 Detailed Description

Definition at line 10 of file MuscleNoHill.py.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 `def MuscleNoHill.MuscleNoHill.__init__(self, conf, pool, MUnumber, MUTypelnumber, unit)`

Definition at line 12 of file MuscleNoHill.py.

8.12.3 Member Function Documentation

8.12.3.1 `def MuscleNoHill.MuscleNoHill.atualizeForce(self, activation_Sat)`

Compute the muscle force when no muscle dynamics (Hill model) is used.

This operation is vectorized. Each element of the vectors correspond to one motor unit. For each motor unit, the force is computed by the following formula:

Definition at line 55 of file MuscleNoHill.py.

8.12.4 Member Data Documentation

8.12.4.1 `MuscleNoHill.MuscleNoHill.conf`

Definition at line 14 of file MuscleNoHill.py.

8.12.4.2 `MuscleNoHill.MuscleNoHill.force`

Muscle force along time, in N.

Definition at line 31 of file MuscleNoHill.py.

8.12.4.3 `MuscleNoHill.MuscleNoHill.maximumActivationForce`

This is used for normalization purposes.

It is the maximum force that the muscle reach when the Hill model is not used.

Definition at line 29 of file `MuscleNoHill.py`.

8.12.4.4 `MuscleNoHill.MuscleNoHill.MUnumber`

Definition at line 16 of file `MuscleNoHill.py`.

8.12.4.5 `MuscleNoHill.MuscleNoHill.MUtypelnumber`

Definition at line 17 of file `MuscleNoHill.py`.

8.12.4.6 `MuscleNoHill.MuscleNoHill.pool`

Definition at line 15 of file `MuscleNoHill.py`.

8.12.4.7 `MuscleNoHill.MuscleNoHill.timeIndex`

Definition at line 33 of file `MuscleNoHill.py`.

8.12.4.8 `MuscleNoHill.MuscleNoHill.twitchAmp_N`

Amplitude of the muscle unit twitch, in N (see `atualizeForceNoHill` function explanation).

Definition at line 22 of file `MuscleNoHill.py`.

8.12.4.9 `MuscleNoHill.MuscleNoHill.twTet`

Twitch- tetanus relationship (see `atualizeForceNoHill` function explanation)

Definition at line 20 of file `MuscleNoHill.py`.

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

- [MuscleNoHill.py](#)

8.13 `MuscularActivation.MuscularActivation` Class Reference

Public Member Functions

- def `__init__` (self, `conf`, `pool`, `MUnumber`, unit)
- def `atualizeActivationSignal` (self, t, unit)
Update the activation signal of the motor units.

Public Attributes

- [conf](#)
- [pool](#)
- [MUnumber](#)
- [activationModel](#)

Model of the activation signal.

- [ActMatrix](#)

Matrix that multiplied by the vector formed as the formula below gives the activation signal at instant n :

$$Av(n) = \begin{bmatrix} a_1(n-1) & a_1(n-2) & e_1(n-1) & \dots & a_i(n-i) & a_i(n-2) & e_i(n-1) & \dots & a_{NMU}(n-1) & a_{NMU}(n-2) & e_{NMU}(n-1) \end{bmatrix}^T \quad (8.4)$$

where $a_i(n)$ is the activation signal of the motor unit i , $e_i(n)$ is $1/T$ (inverse of simulation time step, Dirac's delta approximation) if the motor unit i , fired at instant n .

- [an](#)

Is a vector formed as:

$$Av(n) = \begin{bmatrix} a_1(n-1) & a_1(n-2) & e_1(n-1) & \dots & a_i(n-i) & a_i(n-2) & e_i(n-1) & \dots & a_{NMU}(n-1) & a_{NMU}(n-2) & e_{NMU}(n-1) \end{bmatrix}^T \quad (8.5)$$

It is multiplied by the matrix `actMatrix` to obtain the activation signal (see `actMatrix` explanation)

- [activation_nonSat](#)

The non-saturated activation signal of all motor units (see `actMatrix` explanation).

- [bSat](#)

The parameter b (see `twitchSaturation` function explanation) of each motor unit.

- [activation_Sat](#)

The non-saturated activation signal of all motor units (see `actMatrix` explanation).

- [diracDeltaValue](#)

Dirac's delta approximation amplitude value.

8.13.1 Detailed Description

Definition at line 31 of file `MuscularActivation.py`.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 `def MuscularActivation.MuscularActivation.__init__(self, conf, pool, MUnumber, unit)`

Definition at line 34 of file `MuscularActivation.py`.

8.13.3 Member Function Documentation

8.13.3.1 `def MuscularActivation.MuscularActivation.atualizeActivationSignal(self, t, unit)`

Update the activation signal of the motor units.

- Inputs:

– **t**: current instant, in ms.

Definition at line 109 of file MuscularActivation.py.

Here is the call graph for this function:



8.13.4 Member Data Documentation

8.13.4.1 MuscularActivation.MuscularActivation.activation_nonSat

The non-saturated activation signal of all motor units (see actMatrix explanation).

Definition at line 86 of file MuscularActivation.py.

8.13.4.2 MuscularActivation.MuscularActivation.activation_Sat

The non-saturated activation signal of all motor units (see actMatrix explanation).

Definition at line 97 of file MuscularActivation.py.

8.13.4.3 MuscularActivation.MuscularActivation.activationModel

Model of the activation signal.

For now, it can be *SOCDS* (second order critically damped system).

Definition at line 41 of file MuscularActivation.py.

8.13.4.4 MuscularActivation.MuscularActivation.ActMatrix

Matrix that multiplied by the vector formed as the formula below gives the activation signal at instant n :

$$Av(n) = \begin{bmatrix} a_1(n-1) & a_1(n-2) & e_1(n-1) & \dots & a_i(n-i) & a_i(n-2) & e_i(n-1) & \dots & a_{N_{MU}}(n-1) & a_{N_{MU}}(n-2) & e_{N_{MU}}(n-1) \end{bmatrix}^T \quad (8.6)$$

where $a_i(n)$ is the activation signal of the motor unit i , $e_i(n)$ is $1/T$ (inverse of simulation time step, Dirac's delta approximation) if the motor unit i , fired at instant n .

The vector Av is updated every step at the function `atualizeActivationSignal`. The activation matrix itself is formed as:

$$A = \begin{bmatrix} 2 \exp\left(-\frac{T}{T_{c1}}\right) & -\exp\left(-2\frac{T}{T_{c1}}\right) & \frac{T^2}{T_{c1}^2} \exp\left(1-\frac{T}{T_{c1}}\right) & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & 0 & 0 & 2 \exp\left(-\frac{T}{T_{c1}}\right) & -\exp\left(-2\frac{T}{T_{c1}}\right) & \frac{T^2}{T_{c1}^2} \exp\left(1-\frac{T}{T_{c1}}\right) & 0 & \dots & \dots & 0 \\ 0 & 0 & \dots & \dots & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & \dots & 0 & 0 & 0 & 2 \exp\left(-\frac{T}{T_{cN_{MU}}}\right) & -\exp\left(-2\frac{T}{T_{cN_{MU}}}\right) & \frac{T^2}{T_{cN_{MU}}^2} \exp\left(1-\frac{T}{T_{cN_{MU}}}\right) \end{bmatrix} \quad (8.7)$$

The nonsaturated activation signal a of all the motor units is obtained with:

$$a = A.Av \quad (8.8)$$

where each element of a is the activation signal of a motor unit.

Definition at line 69 of file MuscularActivation.py.

8.13.4.5 MuscularActivation.MuscularActivation.an

Is a vector formed as:

$$Av(n) = \begin{bmatrix} a_1(n-1) & a_1(n-2) & e_1(n-1) & \dots & a_i(n-i) & a_i(n-2) & e_i(n-1) & \dots & a_{NMU}(n-1) & a_{NMU}(n-2) & e_{NMU}(n-1) \end{bmatrix}^T \quad (8.9)$$

It is multiplied by the matrix actMatrix to obtain the activation signal (see actMatrix explanation)

Definition at line 83 of file MuscularActivation.py.

8.13.4.6 MuscularActivation.MuscularActivation.bSat

The parameter b (see twitchSaturation function explanation) of each motor unit.

Definition at line 89 of file MuscularActivation.py.

8.13.4.7 MuscularActivation.MuscularActivation.conf

Definition at line 36 of file MuscularActivation.py.

8.13.4.8 MuscularActivation.MuscularActivation.diracDeltaValue

Dirac's delta approximation amplitude value.

Is the inverse of the simulation time step ($1/T$).

Definition at line 100 of file MuscularActivation.py.

8.13.4.9 MuscularActivation.MuscularActivation.MUnumber

Definition at line 38 of file MuscularActivation.py.

8.13.4.10 MuscularActivation.MuscularActivation.pool

Definition at line 37 of file MuscularActivation.py.

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

- [MuscularActivation.py](#)

8.14 NeuralTract.NeuralTract Class Reference

Class that implements a a neural tract, composed by the descending commands from the motor cortex.

Public Member Functions

- `def __init__ (self, conf, pool)`
Constructor.
- `def atualizePool (self, t)`
Update all neural tract units from the neural tract.
- `def listSpikes (self)`
List the spikes that occurred in neural tract units.

Public Attributes

- `kind`
Indicates that is a neural tract.
- `pool`
String with the name of the Neural tract.
- `Number`
The number of neural tract units.
- `unit`
List of NeuralTRactUnit objects.
- `GammaOrder`
- `poolTerminalSpikes`
Vector with the instants of spikes in the terminal, in ms.
- `target`
Indicates the measure that the TargetFunction of the spikes follows.
- `FR`
The mean firing rate of the neural tract units.
- `timeIndex`

8.14.1 Detailed Description

Class that implements a a neural tract, composed by the descending commands from the motor cortex.

Definition at line 15 of file NeuralTract.py.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 `def NeuralTract.NeuralTract.__init__ (self, conf, pool)`

Constructor.

- Inputs:
 - **conf**: `Configuration` object with the simulation parameters.
 - **pool**: string with the name of the Neural tract.

Definition at line 26 of file NeuralTract.py.

8.14.3 Member Function Documentation

8.14.3.1 `def NeuralTract.NeuralTract.atualizePool (self, t)`

Update all neural tract units from the neural tract.

- Inputs:
 - `t`: cuurent instant, in ms.

Definition at line 67 of file NeuralTract.py.

8.14.3.2 `def NeuralTract.NeuralTract.listSpikes (self)`

List the spikes that occurred in neural tract units.

Definition at line 75 of file NeuralTract.py.

8.14.4 Member Data Documentation

8.14.4.1 `NeuralTract.NeuralTract.FR`

The mean firing rate of the neural tract units.

Definition at line 53 of file NeuralTract.py.

8.14.4.2 `NeuralTract.NeuralTract.GammaOrder`

Definition at line 37 of file NeuralTract.py.

8.14.4.3 `NeuralTract.NeuralTract.kind`

Indicates that is a neural tract.

Definition at line 28 of file NeuralTract.py.

8.14.4.4 `NeuralTract.NeuralTract.Number`

The number of neural tract units.

Definition at line 32 of file NeuralTract.py.

8.14.4.5 NeuralTract.NeuralTract.pool

String with the name of the Neural tract.

Definition at line 30 of file NeuralTract.py.

8.14.4.6 NeuralTract.NeuralTract.poolTerminalSpikes

Vector with the instants of spikes in the terminal, in ms.

Definition at line 42 of file NeuralTract.py.

8.14.4.7 NeuralTract.NeuralTract.target

Indicates the measure that the TargetFunction of the spikes follows.

For now ita can be *ISI* (interspike interval) or *FR* (firing rate).

Definition at line 46 of file NeuralTract.py.

8.14.4.8 NeuralTract.NeuralTract.timeIndex

Definition at line 56 of file NeuralTract.py.

8.14.4.9 NeuralTract.NeuralTract.unit

List of NeuralTractUnit objects.

Definition at line 35 of file NeuralTract.py.

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

- [NeuralTract.py](#)

8.15 NeuralTractUnit.NeuralTractUnit Class Reference

Class that implements a neural tract unit.

Public Member Functions

- def `__init__` (self, conf, pool, [GammaOrder](#), [index](#))
Constructor.
- def `atualizeNeuralTractUnit` (self, t, FR)
- def `transmitSpikes` (self, t)

Public Attributes

- [GammaOrder](#)

Integer order of the Gamma distribution.

- [spikesGenerator](#)

A [PointProcessGenerator](#) object, corresponding the generator of spikes of the neural tract unit.

- [terminalSpikeTrain](#)

List of the spikes of the neural tract unit.

- [kind](#)

- [SynapsesOut](#)

- [transmitSpikesThroughSynapses](#)

- [indicesOfSynapsesOnTarget](#)

- [index](#)

Integer corresponding to the neural tract unit identification.

8.15.1 Detailed Description

Class that implements a neural tract unit.

It consists of a point process generator.

Definition at line 21 of file NeuralTractUnit.py.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 `def NeuralTractUnit.NeuralTractUnit.__init__(self, conf, pool, GammaOrder, index)`

Constructor.

- Inputs:
 - **conf**: [Configuration](#) object with the simulation parameters.
 - **pool**: string with the name of the Neural tract.
 - **index**: integer corresponding to the neural tract unit identification.

Definition at line 36 of file NeuralTractUnit.py.

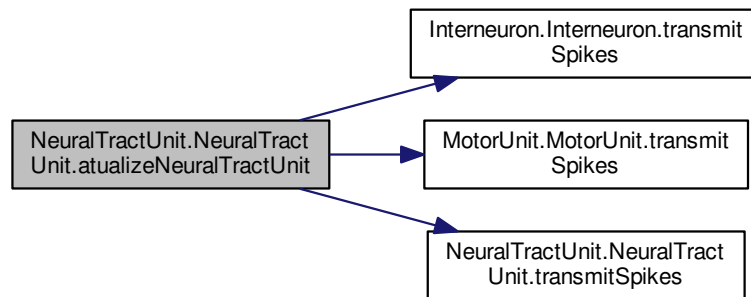
8.15.3 Member Function Documentation

8.15.3.1 `def NeuralTractUnit.NeuralTractUnit.atualizeNeuralTractUnit (self, t, FR)`

- Inputs:
 - **t**: current instant, in ms.
 - **FR**:

Definition at line 68 of file NeuralTractUnit.py.

Here is the call graph for this function:



8.15.3.2 `def NeuralTractUnit.NeuralTractUnit.transmitSpikes (self, t)`

- Inputs:
 - **t**: current instant, in ms.

Definition at line 80 of file NeuralTractUnit.py.

Here is the caller graph for this function:



8.15.4 Member Data Documentation

8.15.4.1 NeuralTractUnit.NeuralTractUnit.GammaOrder

Integer order of the Gamma distribution.

Definition at line 39 of file NeuralTractUnit.py.

8.15.4.2 NeuralTractUnit.NeuralTractUnit.index

Integer corresponding to the neural tract unit identification.

Definition at line 57 of file NeuralTractUnit.py.

8.15.4.3 NeuralTractUnit.NeuralTractUnit.indicesOfSynapsesOnTarget

Definition at line 54 of file NeuralTractUnit.py.

8.15.4.4 NeuralTractUnit.NeuralTractUnit.kind

Definition at line 47 of file NeuralTractUnit.py.

8.15.4.5 NeuralTractUnit.NeuralTractUnit.spikesGenerator

A [PointProcessGenerator](#) object, corresponding the generator of spikes of the neural tract unit.

Definition at line 43 of file NeuralTractUnit.py.

8.15.4.6 NeuralTractUnit.NeuralTractUnit.SynapsesOut

Definition at line 52 of file NeuralTractUnit.py.

8.15.4.7 NeuralTractUnit.NeuralTractUnit.terminalSpikeTrain

List of the spikes of the neural tract unit.

Definition at line 45 of file NeuralTractUnit.py.

8.15.4.8 NeuralTractUnit.NeuralTractUnit.transmitSpikesThroughSynapses

Definition at line 53 of file NeuralTractUnit.py.

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

- [NeuralTractUnit.py](#)

8.16 object Class Reference

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

- [NeuralTract.py](#)

8.17 PointProcessGenerator.PointProcessGenerator Class Reference

Generator of point processes.

Public Member Functions

- `def __init__(self, GammaOrder, index)`
Constructor.
- `def atualizeGenerator(self, t, firingRate)`

Public Attributes

- [GammaOrder](#)
Integer order of the Gamma distribution.
- [GammaOrderInv](#)
Inverse of the GammaOrder.
- [index](#)
Integer corresponding to the unit order in the pool to which this generator is associated.
- [y](#)
Auxiliary variable cumulating a value that indicates whether there will be a new spike or not.
- [threshold](#)
Spike threshold.
- [points](#)
List of spike instants of the generator.

8.17.1 Detailed Description

Generator of point processes.

Definition at line 47 of file PointProcessGenerator.py.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 `def PointProcessGenerator.PointProcessGenerator.__init__(self, GammaOrder, index)`

Constructor.

- Inputs:
 - **GammaOrder**: integer order of the Gamma distribution.
 - **index**: integer corresponding to the unit order in the pool.

Definition at line 58 of file PointProcessGenerator.py.

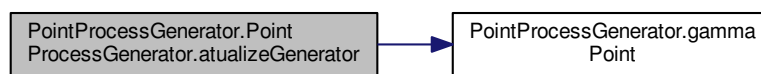
8.17.3 Member Function Documentation

8.17.3.1 `def PointProcessGenerator.PointProcessGenerator.atualizeGenerator (self, t, firingRate)`

- Inputs:
 - **t**: current instant, in ms.
 - **firingRate**: instant firing rate, in spikes/s.

Definition at line 87 of file PointProcessGenerator.py.

Here is the call graph for this function:



8.17.4 Member Data Documentation

8.17.4.1 `PointProcessGenerator.PointProcessGenerator.GammaOrder`

Integer order of the Gamma distribution.

Gamma order 1 is Poisson process and order 10 is a Gaussian process.

Definition at line 61 of file PointProcessGenerator.py.

8.17.4.2 `PointProcessGenerator.PointProcessGenerator.GammaOrderInv`

Inverse of the GammaOrder.

This is necessary for computational efficiency.

Definition at line 64 of file PointProcessGenerator.py.

8.17.4.3 `PointProcessGenerator.PointProcessGenerator.index`

Integer corresponding to the unit order in the pool to which this generator is associated.

Definition at line 67 of file PointProcessGenerator.py.

8.17.4.4 `PointProcessGenerator.PointProcessGenerator.points`

List of spike instants of the generator.

Definition at line 77 of file PointProcessGenerator.py.

8.17.4.5 `PointProcessGenerator.PointProcessGenerator.threshold`

Spike threshold.

When the auxiliary variable y reaches the value of threshold, there is a new spike.

Definition at line 75 of file `PointProcessGenerator.py`.

8.17.4.6 `PointProcessGenerator.PointProcessGenerator.y`

Auxiliary variable cumulating a value that indicates whether there will be a new spike or not.

Definition at line 71 of file `PointProcessGenerator.py`.

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

- [PointProcessGenerator.py](#)

8.18 `PulseConductanceState.PulseConductanceState` Class Reference

Implements the Destexhe pulse approximation of the solution of the states of the Hodgkin-Huxley neuron model.

Public Member Functions

- `def __init__ (self, kind, conf, pool, neuronKind, index)`
Initializes the pulse conductance state.
- `def changeState (self, t)`
Void function that modify the current situation (true/false) of the state.
- `def computeStateValue (self, t)`
Compute the state value by using the approximation of Destexhe (1997) to compute the Hodgkin-Huxley states.

Public Attributes

- `kind`
- `value`
- `v0`
- `t0`
- `state`
- `beta_ms1`
- `alpha_ms1`
- `PulseDur_ms`
- `actType`
- `computeValueOn`
- `computeValueOff`

8.18.1 Detailed Description

Implements the Destexhe pulse approximation of the solution of the states of the Hodgkin-Huxley neuron model.

Definition at line 54 of file PulseConductanceState.py.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 `def PulseConductanceState.PulseConductanceState.__init__(self, kind, conf, pool, neuronKind, index)`

Initializes the pulse conductance state.

Variables: *kind* - type of the state(m, h, n, q). *conf* - an instance of the [Configuration](#) class with the functions to correctly parameterize the model. See the [Configuration](#) class. *pool* - the pool that this state belongs. *neuronKind* - *index* - the index of the unit that this state belongs.

Definition at line 66 of file PulseConductanceState.py.

8.18.3 Member Function Documentation

8.18.3.1 `def PulseConductanceState.PulseConductanceState.changeState(self, t)`

Void function that modify the current situation (true/false) of the state.

- Inputs:
 - *t*: current instant, in ms.

Definition at line 105 of file PulseConductanceState.py.

Here is the caller graph for this function:



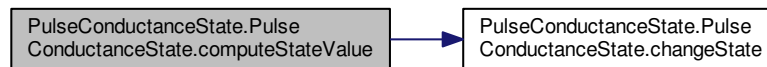
8.18.3.2 `def PulseConductanceState.PulseConductanceState.computeStateValue (self, t)`

Compute the state value by using the approximation of Destexhe (1997) to compute the Hodgkin-Huxley states.

- Input:
 - `t`: current instant, in ms.

Definition at line 117 of file `PulseConductanceState.py`.

Here is the call graph for this function:



8.18.4 Member Data Documentation

8.18.4.1 `PulseConductanceState.PulseConductanceState.actType`

Definition at line 81 of file `PulseConductanceState.py`.

8.18.4.2 `PulseConductanceState.PulseConductanceState.alpha_ms1`

Definition at line 77 of file `PulseConductanceState.py`.

8.18.4.3 `PulseConductanceState.PulseConductanceState.beta_ms1`

Definition at line 76 of file `PulseConductanceState.py`.

8.18.4.4 `PulseConductanceState.PulseConductanceState.computeValueOff`

Definition at line 91 of file `PulseConductanceState.py`.

8.18.4.5 `PulseConductanceState.PulseConductanceState.computeValueOn`

Definition at line 90 of file `PulseConductanceState.py`.

8.18.4.6 `PulseConductanceState.PulseConductanceState.kind`

Definition at line 67 of file `PulseConductanceState.py`.

8.18.4.7 `PulseConductanceState.PulseConductanceState.PulseDur_ms`

Definition at line 78 of file `PulseConductanceState.py`.

8.18.4.8 `PulseConductanceState.PulseConductanceState.state`

Definition at line 74 of file `PulseConductanceState.py`.

8.18.4.9 `PulseConductanceState.PulseConductanceState.t0`

Definition at line 72 of file `PulseConductanceState.py`.

8.18.4.10 `PulseConductanceState.PulseConductanceState.v0`

Definition at line 71 of file `PulseConductanceState.py`.

8.18.4.11 `PulseConductanceState.PulseConductanceState.value`

Definition at line 68 of file `PulseConductanceState.py`.

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

- [PulseConductanceState.py](#)

8.19 Synapse.Synapse Class Reference

Implements the synapse model from Destexhe (1994) using the computational method from Lytton (1996).

Public Member Functions

- `def __init__(self, conf, pool, index, compartment, kind, neuronKind)`
Constructor.

Public Attributes

- [pool](#)
- [kind](#)
- [neuronKind](#)
- [EqPot_mV](#)
- [alpha_ms1](#)
- [beta_ms1](#)
- [Tmax_mM](#)
- [tPeak_ms](#)

Pulse duration, in ms.
- [gmax_muS](#)
- [delay_ms](#)
- [dynamics](#)
- [variation](#)
- [timeConstant_ms](#)
- [gMaxTot_muS](#)

The sum of individual conductances of all synapses in the compartment, in μS ($G_{max} = \sum_{i=1}^N g_i$).
- [numberOfIncomingSynapses](#)
- [rInf](#)

The fraction of postsynaptic receptors that would be bound to neurotransmitters after an infinite amount of time with neurotransmitter being released.
- [tauOn](#)

Time constant during a pulse, in ms.
- [tauOff](#)

Time constant after a pulse, in ms.
- [expFinish](#)

Is the value of the exponential at the end of the pulse.
- [Non](#)

Sum of the fractions of the individual conductances that are receiving neurotransmitter (during pulse) relative to the G_{max} .
- [Ron](#)

Sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that have neurotransmitters being released (during the pulse).
- [Roff](#)

Sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that do not have neurotransmitters being released (before and after the pulse).
- [t0](#)

Instant that the last spike arrived to the compartment.
- [conductanceState](#)
- [tBeginOfPulse](#)
- [tEndOfPulse](#)
- [tLastPulse](#)
- [ri](#)

List with the fractions of postsynaptic receptors that are bound to neurotransmitters of the individual synapses.
- [ti](#)

List with the instants of spike arriving at each conductance, in ms.
- [dynamicGmax](#)

8.19.1 Detailed Description

Implements the synapse model from Destexhe (1994) using the computational method from Lytton (1996).

Definition at line 323 of file Synapse.py.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 `def Synapse.Synapse.__init__(self, conf, pool, index, compartment, kind, neuronKind)`

Constructor.

- Input:
 - **conf**: [Configuration](#) object with the simulation parameters.
 - **pool**: string with identification of the pool to which the synapse belongs.
 - **index**: integer identification of the unit in the pool.
 - **compartment**: integer identification of the compartment of the unit where the synapse is.
 - **kind**: string with the type of synapse. It can be *excitatory* or *inhibitory*.
 - **neuronKind**:

Definition at line 343 of file Synapse.py.

8.19.3 Member Data Documentation

8.19.3.1 `Synapse.Synapse.alpha_ms1`

Definition at line 349 of file Synapse.py.

8.19.3.2 `Synapse.Synapse.beta_ms1`

Definition at line 350 of file Synapse.py.

8.19.3.3 `Synapse.Synapse.conductanceState`

Definition at line 400 of file Synapse.py.

8.19.3.4 `Synapse.Synapse.delay_ms`

Definition at line 356 of file Synapse.py.

8.19.3.5 `Synapse.Synapse.dynamicGmax`

Definition at line 412 of file Synapse.py.

8.19.3.6 `Synapse.Synapse.dynamics`

Definition at line 357 of file Synapse.py.

8.19.3.7 Synapse.Synapse.EqPot_mV

Definition at line 348 of file Synapse.py.

8.19.3.8 Synapse.Synapse.expFinish

Is the value of the exponential at the end of the pulse.

It is computed as $\exp(T_{dur}/\tau_{on})$.

Definition at line 380 of file Synapse.py.

8.19.3.9 Synapse.Synapse.gmax_muS

Definition at line 355 of file Synapse.py.

8.19.3.10 Synapse.Synapse.gMaxTot_muS

The sum of individual conductances of all synapses in the compartment, in μS ($G_{max} = \sum_{i=1}^N g_i$).

Definition at line 363 of file Synapse.py.

8.19.3.11 Synapse.Synapse.kind

Definition at line 345 of file Synapse.py.

8.19.3.12 Synapse.Synapse.neuronKind

Definition at line 346 of file Synapse.py.

8.19.3.13 Synapse.Synapse.Non

Sum of the fractions of the individual conductances that are receiving neurotransmitter (during pulse) relative to the G_{max} .

(

Definition at line 387 of file Synapse.py.

8.19.3.14 Synapse.Synapse.numberOfIncomingSynapses

Definition at line 364 of file Synapse.py.

8.19.3.15 Synapse.Synapse.pool

Definition at line 344 of file Synapse.py.

8.19.3.16 Synapse.Synapse.ri

List with the fractions of postsynaptic receptors that are bound to neurotransmitters of the individual synapses.

Definition at line 407 of file Synapse.py.

8.19.3.17 Synapse.Synapse.rInf

The fraction of postsynaptic receptors that would be bound to neurotransmitters after an infinite amount of time with neurotransmitter being released.

Definition at line 370 of file Synapse.py.

8.19.3.18 Synapse.Synapse.Roff

Sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that do not have neurotransmitters being released (before and after the pulse).

Definition at line 396 of file Synapse.py.

8.19.3.19 Synapse.Synapse.Ron

Sum of the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that have neurotransmitters being released (during the pulse).

Definition at line 391 of file Synapse.py.

8.19.3.20 Synapse.Synapse.t0

Instant that the last spike arrived to the compartment.

Definition at line 398 of file Synapse.py.

8.19.3.21 Synapse.Synapse.tauOff

Time constant after a pulse, in ms.

$$\tau_{off} = \frac{1}{\beta}$$

Definition at line 376 of file Synapse.py.

8.19.3.22 Synapse.Synapse.tauOn

Time constant during a pulse, in ms.

$$\tau_{on} = \frac{1}{\alpha \cdot T_{max} + \beta}$$

Definition at line 373 of file Synapse.py.

8.19.3.23 Synapse.Synapse.tBeginOfPulse

Definition at line 401 of file Synapse.py.

8.19.3.24 Synapse.Synapse.tEndOfPulse

Definition at line 402 of file Synapse.py.

8.19.3.25 Synapse.Synapse.ti

List with the instants of spike arriving at each conductance, in ms.

Definition at line 410 of file Synapse.py.

8.19.3.26 Synapse.Synapse.timeConstant_ms

Definition at line 359 of file Synapse.py.

8.19.3.27 Synapse.Synapse.tLastPulse

Definition at line 403 of file Synapse.py.

8.19.3.28 Synapse.Synapse.Tmax_mM

Definition at line 351 of file Synapse.py.

8.19.3.29 Synapse.Synapse.tPeak_ms

Pulse duration, in ms.

Definition at line 353 of file Synapse.py.

8.19.3.30 Synapse.Synapse.variation

Definition at line 358 of file Synapse.py.

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

- [Synapse.py](#)

8.20 SynapsesFactory.SynapsesFactory Class Reference

Class to build all the synapses in the system.

Public Member Functions

- `def __init__(self, conf, pools)`
Constructor.

Public Attributes

- `numberOfSynapses`
Total number of synapses in the system.

8.20.1 Detailed Description

Class to build all the synapses in the system.

Definition at line 18 of file SynapsesFactory.py.

8.20.2 Constructor & Destructor Documentation

8.20.2.1 `def SynapsesFactory.SynapsesFactory.__init__(self, conf, pools)`

Constructor.

- Inputs:
 - **conf**: [Configuration](#) object with the simulation parameters.
 - **pools**: list of all the pools in the system.

Definition at line 32 of file SynapsesFactory.py.

8.20.3 Member Data Documentation

8.20.3.1 SynapsesFactory.SynapsesFactory.numberOfSynapses

Total number of synapses in the system.

Definition at line 34 of file SynapsesFactory.py.

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

- [SynapsesFactory.py](#)

8.21 SynapticNoise.SynapticNoise Class Reference

Class that implements a synaptic noise for a pool of neurons.

Public Member Functions

- `def __init__ (self, conf, pool)`
Constructor.
- `def atualizePool (self, t)`
Update all neural tract units from the neural tract.
- `def listSpikes (self)`
List the spikes that occurred in neural tract units.

Public Attributes

- [kind](#)
Indicates that is a neural tract.
- [pool](#)
String with the name of the pool.
- [Number](#)
The number of neural tract units.
- [unit](#)
List of [NeuralTractUnit](#) objects.
- [GammaOrder](#)
- [poolTerminalSpikes](#)
Vector with the instants of spikes in the terminal, in ms.
- [target](#)
Indicates the measure that the TargetFunction of the spikes follows.
- [FR](#)
The mean firing rate of the neural tract units.
- [timeIndex](#)

8.21.1 Detailed Description

Class that implements a synaptic noise for a pool of neurons.

Definition at line 14 of file SynapticNoise.py.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 `def SynapticNoise.SynapticNoise.__init__(self, conf, pool)`

Constructor.

- Inputs:
 - **conf**: [Configuration](#) object with the simulation parameters.
 - **pool**: string with the name of the pool.

Definition at line 25 of file SynapticNoise.py.

8.21.3 Member Function Documentation

8.21.3.1 `def SynapticNoise.SynapticNoise.atualizePool(self, t)`

Update all neural tract units from the neural tract.

- Inputs:
 - **t**: current instant, in ms.

Definition at line 67 of file SynapticNoise.py.

8.21.3.2 `def SynapticNoise.SynapticNoise.listSpikes(self)`

List the spikes that occurred in neural tract units.

Definition at line 77 of file SynapticNoise.py.

8.21.4 Member Data Documentation

8.21.4.1 `SynapticNoise.SynapticNoise.FR`

The mean firing rate of the neural tract units.

Definition at line 52 of file SynapticNoise.py.

8.21.4.2 `SynapticNoise.SynapticNoise.GammaOrder`

Definition at line 36 of file `SynapticNoise.py`.

8.21.4.3 `SynapticNoise.SynapticNoise.kind`

Indicates that is a neural tract.

Definition at line 27 of file `SynapticNoise.py`.

8.21.4.4 `SynapticNoise.SynapticNoise.Number`

The number of neural tract units.

Definition at line 31 of file `SynapticNoise.py`.

8.21.4.5 `SynapticNoise.SynapticNoise.pool`

String with the name of the pool.

Definition at line 29 of file `SynapticNoise.py`.

8.21.4.6 `SynapticNoise.SynapticNoise.poolTerminalSpikes`

Vector with the instants of spikes in the terminal, in ms.

Definition at line 41 of file `SynapticNoise.py`.

8.21.4.7 `SynapticNoise.SynapticNoise.target`

Indicates the measure that the `TargetFunction` of the spikes follows.

For now it can be *ISI* (interspike interval) or *FR* (firing rate).

Definition at line 45 of file `SynapticNoise.py`.

8.21.4.8 `SynapticNoise.SynapticNoise.timeIndex`

Definition at line 55 of file `SynapticNoise.py`.

8.21.4.9 `SynapticNoise.SynapticNoise.unit`

List of [NeuralTractUnit](#) objects.

Definition at line 34 of file `SynapticNoise.py`.

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

- [SynapticNoise.py](#)

Chapter 9

File Documentation

9.1 AxonDelay.py File Reference

Classes

- class [AxonDelay.AxonDelay](#)
Class that implements a delay correspondent to the nerve.

Namespaces

- [AxonDelay](#)

9.2 ChannelConductance.py File Reference

Classes

- class [ChannelConductance.ChannelConductance](#)
Class that implements a model of the ionic Channels in a compartment.

Namespaces

- [ChannelConductance](#)

9.3 Compartment.py File Reference

Classes

- class [Compartment.Compartment](#)
Class that implements a neural compartment.

Namespaces

- [Compartment](#)

Functions

- def [Compartment.calcGLEak](#) (area, specificRes)
Computes the leak conductance of the compartment.

9.4 Configuration.py File Reference

Classes

- class [Configuration.Configuration](#)
Class that builds an object of [Configuration](#), based on a configuration file.

Namespaces

- [Configuration](#)

9.5 Interneuron.py File Reference

Classes

- class [Interneuron.Interneuron](#)
Class that implements a motor unit model.

Namespaces

- [Interneuron](#)

Functions

- def [Interneuron.runge_kutta](#) (derivativeFunction, t, x, timeStep, timeStepByTwo, timeStepBySix)
Function to implement the fourth order Runge-Kutta Method to solve numerically a differential equation.

9.6 InterneuronPool.py File Reference

Classes

- class [InterneuronPool.InterneuronPool](#)
Class that implements a motor unit pool.

Namespaces

- [InterneuronPool](#)

9.7 jointAnkleForceTask.py File Reference

Classes

- class [jointAnkleForceTask.jointAnkleForceTask](#)

Namespaces

- [jointAnkleForceTask](#)

9.8 jointAnklePositionTask.py File Reference

Classes

- class [jointAnklePositionTask.jointAnklePositionTask](#)

Namespaces

- [jointAnklePositionTask](#)

9.9 MotorUnit.py File Reference

Classes

- class [MotorUnit.MotorUnit](#)
Class that implements a motor unit model.

Namespaces

- [MotorUnit](#)

Functions

- def [MotorUnit.calcGCoupling](#) (cytR, IComp1, IComp2, dComp1, dComp2)
Calculates the coupling conductance between two compartments.
- def [MotorUnit.compGCouplingMatrix](#) (gc)
Computes the Coupling Matrix to be used in the dVdt function of the N compartments of the motor unit.
- def [MotorUnit.runge_kutta](#) (derivativeFunction, t, x, timeStep, timeStepByTwo, timeStepBySix)
Function to implement the fourth order Runge-Kutta Method to solve numerically a differential equation.

9.10 MotorUnitPool.py File Reference

Classes

- class [MotorUnitPool.MotorUnitPool](#)
Class that implements a motor unit pool.

Namespaces

- [MotorUnitPool](#)

9.11 MuscleHill.py File Reference

Classes

- class [MuscleHill.MuscleHill](#)

Namespaces

- [MuscleHill](#)

9.12 MuscleNoHill.py File Reference

Classes

- class [MuscleNoHill.MuscleNoHill](#)

Namespaces

- [MuscleNoHill](#)

9.13 MuscularActivation.py File Reference

Classes

- class [MuscularActivation.MuscularActivation](#)

Namespaces

- [MuscularActivation](#)

Functions

- def [MuscularActivation.twitchSaturation](#) (activationsat, b)
Computes the muscle unit force after the nonlinear saturation.

9.14 NeuralTract.py File Reference

Classes

- class [NeuralTract.NeuralTract](#)
Class that implements a a neural tract, composed by the descending commands from the motor cortex.

Namespaces

- [NeuralTract](#)

9.15 NeuralTractUnit.py File Reference

Classes

- class [NeuralTractUnit.NeuralTractUnit](#)
Class that implements a neural tract unit.

Namespaces

- [NeuralTractUnit](#)

9.16 PointProcessGenerator.py File Reference

Classes

- class [PointProcessGenerator.PointProcessGenerator](#)
Generator of point processes.

Namespaces

- [PointProcessGenerator](#)

Functions

- def [PointProcessGenerator.gammaPoint](#) (GammaOrder, GammaOrderInv)
*Generates a number according to a Gamma Distribution with an integer order **GammaOrder**.*

9.17 PulseConductanceState.py File Reference

Classes

- class [PulseConductanceState.PulseConductanceState](#)
Implements the Destexhe pulse approximation of the solution of the states of the Hodgkin-Huxley neuron model.

Namespaces

- [PulseConductanceState](#)

Functions

- def [PulseConductanceState.compValOn](#) (v0, alpha, beta, t, t0)
Time course of the state during the pulse for the inactivation states and before and after the pulse for the activation states.
- def [PulseConductanceState.compValOff](#) (v0, alpha, beta, t, t0)
Time course of the state during the pulse for the activation states and before and after the pulse for the inactivation states.

9.18 README.md File Reference

9.19 simulation.py File Reference

Namespaces

- [simulation](#)

Functions

- def [simulation.simulator](#) ()

9.20 Synapse.py File Reference

Classes

- class [Synapse.Synapse](#)
Implements the synapse model from Destexhe (1994) using the computational method from Lytton (1996).

Namespaces

- [Synapse](#)

Functions

- def [Synapse.compSynapCond](#) (Gmax, Ron, Roff)
Computes the synaptic conductance.
- def [Synapse.compRon](#) (Non, rInf, Ron, t0, t, tauOn)
Computes the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that have neurotransmitters being released (during the pulse).
- def [Synapse.compRoff](#) (Roff, t0, t, tauOff)
Computes the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that do not have neurotransmitters being released (before and after the pulse).
- def [Synapse.compRiStart](#) (ri, t, ti, tPeak, tauOff)
Computes the fraction of bound postsynaptic receptors to neurotransmitters in individual synapses when the neurotransmitter begin (begin of the pulse).
- def [Synapse.compRiStop](#) (rInf, ri, expFinish)
Computes the fraction of bound postsynaptic receptors to neurotransmitters in individual synapses when the neurotransmitter release stops (the pulse ends).
- def [Synapse.compRonStart](#) (Ron, ri, synContrib)
Incorporates a new conductance to the set of conductances during a pulse.
- def [Synapse.compRoffStart](#) (Roff, ri, synContrib)
Incorporates a new conductance to the set of conductances that are not during a pulse.
- def [Synapse.compRonStop](#) (Ron, ri, synContrib)
Removes a conductance from the set of conductances during a pulse.
- def [Synapse.compRoffStop](#) (Roff, ri, synContrib)
Removes a conductance from the set of conductances that are not during a pulse.
- def [Synapse.compDynamicGmax](#) (t, gmax, lastPulse, tau, dynamicGmax, var)

9.21 SynapsesFactory.py File Reference

Classes

- class [SynapsesFactory.SynapsesFactory](#)
Class to build all the synapses in the system.

Namespaces

- [SynapsesFactory](#)

9.22 SynapticNoise.py File Reference

Classes

- class [SynapticNoise.SynapticNoise](#)
Class that implements a synaptic noise for a pool of neurons.

Namespaces

- [SynapticNoise](#)

