

PyReMoto

Generated by Doxygen 1.8.6

Thu Jun 16 2016 15:34:30



# Contents

<b>1</b>	<b>ReMoto in Python</b>	<b>1</b>
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Packages	3
<b>3</b>	<b>Hierarchical Index</b>	<b>5</b>
3.1	Class Hierarchy	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List	9
<b>6</b>	<b>Namespace Documentation</b>	<b>11</b>
6.1	AxonDelay Namespace Reference	11
6.2	ChannelConductance Namespace Reference	11
6.3	Compartment Namespace Reference	11
6.3.1	Function Documentation	11
6.3.1.1	calcGLEak	11
6.4	Configuration Namespace Reference	12
6.5	MotorUnit Namespace Reference	12
6.5.1	Function Documentation	12
6.5.1.1	calcGCoupling	12
6.5.1.2	compGCouplingMatrix	13
6.5.1.3	runge_kutta	13
6.6	MotorUnitPool Namespace Reference	14
6.6.1	Function Documentation	14
6.6.1.1	twitchSaturation	14
6.7	NeuralTract Namespace Reference	15
6.8	NeuralTractUnit Namespace Reference	15
6.9	PointProcessGenerator Namespace Reference	15
6.9.1	Function Documentation	15

6.9.1.1	<a href="#">gammaPoint</a>	15
6.10	<a href="#">PulseConductanceState Namespace Reference</a>	16
6.10.1	<a href="#">Function Documentation</a>	16
6.10.1.1	<a href="#">compValOff</a>	16
6.10.1.2	<a href="#">compValOn</a>	16
6.11	<a href="#">simulation Namespace Reference</a>	16
6.11.1	<a href="#">Function Documentation</a>	17
6.11.1.1	<a href="#">simulador</a>	17
6.12	<a href="#">Synapse Namespace Reference</a>	17
6.12.1	<a href="#">Function Documentation</a>	17
6.12.1.1	<a href="#">compRiStart</a>	17
6.12.1.2	<a href="#">compRiStop</a>	17
6.12.1.3	<a href="#">compRoff</a>	18
6.12.1.4	<a href="#">compRoffStart</a>	18
6.12.1.5	<a href="#">compRoffStop</a>	18
6.12.1.6	<a href="#">compRon</a>	18
6.12.1.7	<a href="#">compRonStart</a>	18
6.12.1.8	<a href="#">compRonStop</a>	19
6.12.1.9	<a href="#">compSynapCond</a>	19
6.13	<a href="#">SynapsesFactory Namespace Reference</a>	19
<b>7</b>	<b><a href="#">Class Documentation</a></b>	<b>21</b>
7.1	<a href="#">AxonDelay.AxonDelay Class Reference</a>	21
7.1.1	<a href="#">Detailed Description</a>	22
7.1.2	<a href="#">Constructor &amp; Destructor Documentation</a>	22
7.1.2.1	<a href="#">__init__</a>	22
7.1.3	<a href="#">Member Function Documentation</a>	22
7.1.3.1	<a href="#">addSpinalSpike</a>	22
7.1.3.2	<a href="#">addTerminalSpike</a>	22
7.1.4	<a href="#">Member Data Documentation</a>	23
7.1.4.1	<a href="#">index</a>	23
7.1.4.2	<a href="#">latencySpinalTerminal_ms</a>	23
7.1.4.3	<a href="#">latencyStimulusSpinal_ms</a>	23
7.1.4.4	<a href="#">latencyStimulusTerminal_ms</a>	23
7.1.4.5	<a href="#">length_m</a>	23
7.1.4.6	<a href="#">stimulusPositiontoTerminal</a>	23
7.1.4.7	<a href="#">terminalSpikeTrain</a>	24
7.1.4.8	<a href="#">velocity_m_s</a>	24
7.2	<a href="#">ChannelConductance.ChannelConductance Class Reference</a>	24
7.2.1	<a href="#">Detailed Description</a>	25

7.2.2	Constructor & Destructor Documentation . . . . .	25
7.2.2.1	__init__ . . . . .	25
7.2.3	Member Function Documentation . . . . .	25
7.2.3.1	compCondKf . . . . .	25
7.2.3.2	compCondKs . . . . .	25
7.2.3.3	compCondNa . . . . .	26
7.2.3.4	computeCurrent . . . . .	26
7.2.4	Member Data Documentation . . . . .	26
7.2.4.1	compCond . . . . .	26
7.2.4.2	condState . . . . .	26
7.2.4.3	EqPot_mV . . . . .	26
7.2.4.4	gmax_muS . . . . .	26
7.2.4.5	kind . . . . .	26
7.2.4.6	lenStates . . . . .	27
7.2.4.7	stateType . . . . .	27
7.3	Compartment.Compartment Class Reference . . . . .	27
7.3.1	Detailed Description . . . . .	28
7.3.2	Constructor & Destructor Documentation . . . . .	28
7.3.2.1	__init__ . . . . .	28
7.3.3	Member Function Documentation . . . . .	28
7.3.3.1	computeCurrent . . . . .	28
7.3.4	Member Data Documentation . . . . .	28
7.3.4.1	capacitance_nF . . . . .	28
7.3.4.2	Channels . . . . .	28
7.3.4.3	diameter_mum . . . . .	29
7.3.4.4	gLeak . . . . .	29
7.3.4.5	index . . . . .	29
7.3.4.6	kind . . . . .	29
7.3.4.7	length_mum . . . . .	29
7.3.4.8	neuronKind . . . . .	29
7.3.4.9	numberChannels . . . . .	29
7.3.4.10	SynapsesIn . . . . .	29
7.3.4.11	SynapsesOut . . . . .	29
7.4	Configuration.Configuration Class Reference . . . . .	30
7.4.1	Detailed Description . . . . .	30
7.4.2	Constructor & Destructor Documentation . . . . .	30
7.4.2.1	__init__ . . . . .	30
7.4.3	Member Function Documentation . . . . .	31
7.4.3.1	determineSynapses . . . . .	31
7.4.3.2	inputFunctionGet . . . . .	31

7.4.3.3	parameterSet	31
7.4.4	Member Data Documentation	31
7.4.4.1	confArray	31
7.4.4.2	simDuration_ms	32
7.4.4.3	timeStep_ms	32
7.4.4.4	timeStepBySix_ms	32
7.4.4.5	timeStepByTwo_ms	32
7.5	MotorUnit.MotorUnit Class Reference	32
7.5.1	Detailed Description	33
7.5.2	Constructor & Destructor Documentation	34
7.5.2.1	__init__	34
7.5.3	Member Function Documentation	34
7.5.3.1	addSomaSpike	34
7.5.3.2	atualizeCompartments	34
7.5.3.3	atualizeDelay	35
7.5.3.4	atualizeMotorUnit	35
7.5.3.5	dVdt	36
7.5.4	Member Data Documentation	36
7.5.4.1	bSat	36
7.5.4.2	capacitanceInv	36
7.5.4.3	compartment	37
7.5.4.4	compNumber	37
7.5.4.5	conf	37
7.5.4.6	Delay	37
7.5.4.7	G	37
7.5.4.8	iInjected	37
7.5.4.9	ilonic	37
7.5.4.10	index	37
7.5.4.11	kind	37
7.5.4.12	MNRefPer_ms	38
7.5.4.13	nerve	38
7.5.4.14	somaIndex	38
7.5.4.15	somaSpikeTrain	38
7.5.4.16	terminalSpikeTrain	38
7.5.4.17	threshold_mV	38
7.5.4.18	tSomaSpike	38
7.5.4.19	TwitchAmp_N	38
7.5.4.20	TwitchTc_ms	38
7.5.4.21	twTet	39
7.5.4.22	v_mV	39

7.6	MotorUnitPool.MotorUnitPool Class Reference	39
7.6.1	Detailed Description	40
7.6.2	Constructor & Destructor Documentation	40
7.6.2.1	__init__	40
7.6.3	Member Function Documentation	40
7.6.3.1	atualizeActivationSignal	40
7.6.3.2	atualizeForceNoHill	41
7.6.3.3	atualizeMotorUnitPool	41
7.6.3.4	listSpikes	41
7.6.4	Member Data Documentation	41
7.6.4.1	activation_nonSat	41
7.6.4.2	activation_Sat	41
7.6.4.3	activationModel	41
7.6.4.4	ActMatrix	41
7.6.4.5	an	41
7.6.4.6	atualizeForce	41
7.6.4.7	bSat	42
7.6.4.8	conf	42
7.6.4.9	diracDeltaValue	42
7.6.4.10	force	42
7.6.4.11	hillModel	42
7.6.4.12	kind	42
7.6.4.13	MUnumber	42
7.6.4.14	pool	42
7.6.4.15	poolSomaSpikes	42
7.6.4.16	poolTerminalSpikes	42
7.6.4.17	timeIndex	43
7.6.4.18	twitchAmp_N	43
7.6.4.19	twTet	43
7.6.4.20	unit	43
7.7	NeuralTract.NeuralTract Class Reference	43
7.7.1	Detailed Description	43
7.7.2	Constructor & Destructor Documentation	44
7.7.2.1	__init__	44
7.7.3	Member Function Documentation	44
7.7.3.1	atualizePool	44
7.7.3.2	listSpikes	44
7.7.4	Member Data Documentation	44
7.7.4.1	FR	44
7.7.4.2	kind	44

7.7.4.3	Number	44
7.7.4.4	pool	44
7.7.4.5	poolTerminalSpikes	44
7.7.4.6	target	44
7.7.4.7	timeIndex	45
7.7.4.8	unit	45
7.8	NeuralTractUnit.NeuralTractUnit Class Reference	45
7.8.1	Detailed Description	45
7.8.2	Constructor & Destructor Documentation	45
7.8.2.1	__init__	45
7.8.3	Member Function Documentation	45
7.8.3.1	atualizeNeuralTractUnit	45
7.8.3.2	transmitSpikes	46
7.8.4	Member Data Documentation	46
7.8.4.1	GammaOrder	46
7.8.4.2	indicesOfSynapsesOnTarget	46
7.8.4.3	spikesGenerator	46
7.8.4.4	SynapsesOut	46
7.8.4.5	terminalSpikeTrain	46
7.8.4.6	transmitSpikesThroughSynapses	47
7.9	PointProcessGenerator.PointProcessGenerator Class Reference	47
7.9.1	Detailed Description	47
7.9.2	Constructor & Destructor Documentation	47
7.9.2.1	__init__	47
7.9.3	Member Function Documentation	48
7.9.3.1	atualizeGenerator	48
7.9.4	Member Data Documentation	48
7.9.4.1	GammaOrder	48
7.9.4.2	GammaOrderInv	48
7.9.4.3	index	48
7.9.4.4	points	48
7.9.4.5	threshold	48
7.9.4.6	y	49
7.10	PulseConductanceState.PulseConductanceState Class Reference	49
7.10.1	Detailed Description	49
7.10.2	Constructor & Destructor Documentation	49
7.10.2.1	__init__	49
7.10.3	Member Function Documentation	50
7.10.3.1	changeState	50
7.10.3.2	computeStateValue	50



7.10.4	Member Data Documentation . . . . .	50
7.10.4.1	actType . . . . .	50
7.10.4.2	alpha_ms1 . . . . .	50
7.10.4.3	beta_ms1 . . . . .	50
7.10.4.4	computeValueOff . . . . .	50
7.10.4.5	computeValueOn . . . . .	51
7.10.4.6	kind . . . . .	51
7.10.4.7	PulseDur_ms . . . . .	51
7.10.4.8	state . . . . .	51
7.10.4.9	t0 . . . . .	51
7.10.4.10	v0 . . . . .	51
7.10.4.11	value . . . . .	51
7.11	Synapse.Synapse Class Reference . . . . .	51
7.11.1	Detailed Description . . . . .	52
7.11.2	Constructor & Destructor Documentation . . . . .	52
7.11.2.1	__init__ . . . . .	52
7.11.3	Member Function Documentation . . . . .	53
7.11.3.1	addConductance . . . . .	53
7.11.3.2	computeConductance . . . . .	53
7.11.3.3	computeCurrent . . . . .	53
7.11.3.4	computeCurrent2 . . . . .	54
7.11.3.5	receiveSpike . . . . .	54
7.11.3.6	startConductanceDynamics . . . . .	54
7.11.3.7	startConductanceNone . . . . .	54
7.11.3.8	stopConductanceDynamics . . . . .	55
7.11.3.9	stopConductanceNone . . . . .	55
7.11.4	Member Data Documentation . . . . .	55
7.11.4.1	alpha_ms1 . . . . .	55
7.11.4.2	beta_ms1 . . . . .	55
7.11.4.3	computeCurrent . . . . .	56
7.11.4.4	conductanceState . . . . .	56
7.11.4.5	delay_ms . . . . .	56
7.11.4.6	dynamics . . . . .	56
7.11.4.7	EqPot_mV . . . . .	56
7.11.4.8	expFinish . . . . .	56
7.11.4.9	gmax_muS . . . . .	56
7.11.4.10	gMaxTot_muS . . . . .	56
7.11.4.11	kind . . . . .	56
7.11.4.12	neuronKind . . . . .	56
7.11.4.13	Non . . . . .	56

7.11.4.14 numberOfIncomingSynapses . . . . .	56
7.11.4.15 pool . . . . .	57
7.11.4.16 ri . . . . .	57
7.11.4.17 rInf . . . . .	57
7.11.4.18 Roff . . . . .	57
7.11.4.19 roff . . . . .	57
7.11.4.20 Ron . . . . .	57
7.11.4.21 ron . . . . .	57
7.11.4.22 spikesReceived . . . . .	57
7.11.4.23 startDynamicFunction . . . . .	57
7.11.4.24 startEntrance . . . . .	57
7.11.4.25 stopDynamicFunction . . . . .	57
7.11.4.26 stopEntrance . . . . .	57
7.11.4.27 synContrib . . . . .	58
7.11.4.28 t0 . . . . .	58
7.11.4.29 tauOff . . . . .	58
7.11.4.30 tauOn . . . . .	58
7.11.4.31 tBeginOfPulse . . . . .	58
7.11.4.32 tEndOfPulse . . . . .	58
7.11.4.33 ti . . . . .	58
7.11.4.34 Tmax_mM . . . . .	58
7.11.4.35 tPeak_ms . . . . .	58
7.12 SynapsesFactory.SynapsesFactory Class Reference . . . . .	58
7.12.1 Detailed Description . . . . .	59
7.12.2 Constructor & Destructor Documentation . . . . .	59
7.12.2.1 __init__ . . . . .	59
7.12.3 Member Data Documentation . . . . .	59
7.12.3.1 numberOfSynapses . . . . .	59
<b>8 File Documentation</b>	<b>61</b>
8.1 AxonDelay.py File Reference . . . . .	61
8.2 ChannelConductance.py File Reference . . . . .	61
8.3 Compartment.py File Reference . . . . .	61
8.4 Configuration.py File Reference . . . . .	62
8.5 MotorUnit.py File Reference . . . . .	62
8.6 MotorUnitPool.py File Reference . . . . .	62
8.7 NeuralTract.py File Reference . . . . .	63
8.8 NeuralTractUnit.py File Reference . . . . .	63
8.9 PointProcessGenerator.py File Reference . . . . .	63
8.10 PulseConductanceState.py File Reference . . . . .	63

---

8.11 <a href="#">simulation.py File Reference</a> . . . . .	64
8.12 <a href="#">Synapse.py File Reference</a> . . . . .	64
8.13 <a href="#">SynapsesFactory.py File Reference</a> . . . . .	64
<b><a href="#">Index</a></b>	<b>66</b>



## Chapter 1

# ReMoto in Python

This program ...



## Chapter 2

# Namespace Index

### 2.1 Packages

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

<a href="#">AxonDelay</a>	11
<a href="#">ChannelConductance</a>	11
<a href="#">Compartment</a>	11
<a href="#">Configuration</a>	12
<a href="#">MotorUnit</a>	12
<a href="#">MotorUnitPool</a>	14
<a href="#">NeuralTract</a>	15
<a href="#">NeuralTractUnit</a>	15
<a href="#">PointProcessGenerator</a>	15
<a href="#">PulseConductanceState</a>	16
<a href="#">simulation</a>	16
<a href="#">Synapse</a>	17
<a href="#">SynapsesFactory</a>	19





## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

object

AxonDelay.AxonDelay . . . . .	21
ChannelConductance.ChannelConductance . . . . .	24
Compartment.Compartment . . . . .	27
Configuration.Configuration . . . . .	30
MotorUnit.MotorUnit . . . . .	32
MotorUnitPool.MotorUnitPool . . . . .	39
NeuralTract.NeuralTract . . . . .	43
NeuralTractUnit.NeuralTractUnit . . . . .	45
PointProcessGenerator.PointProcessGenerator . . . . .	47
PulseConductanceState.PulseConductanceState . . . . .	49
Synapse.Synapse . . . . .	51
SynapsesFactory.SynapsesFactory . . . . .	58



## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">AxonDelay.AxonDelay</a>	
Class that implements a delay correspondent to the nerve . . . . .	21
<a href="#">ChannelConductance.ChannelConductance</a>	
Class that implements a model of the ionic Channels in a compartment . . . . .	24
<a href="#">Compartment.Compartment</a>	
Class that implements a neural compartment . . . . .	27
<a href="#">Configuration.Configuration</a>	
Class that builds an object of <a href="#">Configuration</a> , based on a configuration file . . . . .	30
<a href="#">MotorUnit.MotorUnit</a>	
Class that implements a motor unit model . . . . .	32
<a href="#">MotorUnitPool.MotorUnitPool</a>	
Class that implements a motor unit pool . . . . .	39
<a href="#">NeuralTract.NeuralTract</a>	
Classdocs . . . . .	43
<a href="#">NeuralTractUnit.NeuralTractUnit</a>	
Classdocs . . . . .	45
<a href="#">PointProcessGenerator.PointProcessGenerator</a>	
Generator of point processes . . . . .	47
<a href="#">PulseConductanceState.PulseConductanceState</a>	
Implements the Destexhe pulse approximation of the solution of the states of the Hodgkin-Huxley neuron model . . . . .	49
<a href="#">Synapse.Synapse</a>	
Classdocs . . . . .	51
<a href="#">SynapsesFactory.SynapsesFactory</a>	
Classdocs . . . . .	58



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">AxonDelay.py</a>	61
<a href="#">ChannelConductance.py</a>	61
<a href="#">Compartment.py</a>	61
<a href="#">Configuration.py</a>	62
<a href="#">MotorUnit.py</a>	62
<a href="#">MotorUnitPool.py</a>	62
<a href="#">NeuralTract.py</a>	63
<a href="#">NeuralTractUnit.py</a>	63
<a href="#">PointProcessGenerator.py</a>	63
<a href="#">PulseConductanceState.py</a>	63
<a href="#">simulation.py</a>	64
<a href="#">Synapse.py</a>	64
<a href="#">SynapsesFactory.py</a>	64



## Chapter 6

# Namespace Documentation

### 6.1 AxonDelay Namespace Reference

#### Classes

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

### 6.2 ChannelConductance Namespace Reference

#### Classes

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

### 6.3 Compartment Namespace Reference

#### Classes

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

#### Functions

- def [calcGLEak](#)  
*Computes the leak conductance of the compartment.*

#### 6.3.1 Function Documentation

##### 6.3.1.1 `def Compartment.calcGLEak ( area, specificRes )`

Computes the leak conductance of the compartment.

- Input:
  - **area**: area of the compartment in  $\text{cm}^2$ .

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

It is compute according to the following formula:

$$g = 10^6 \cdot \frac{A}{\rho} \quad (6.1)$$

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

Definition at line 32 of file Compartment.py.

## 6.4 Configuration Namespace Reference

### Classes

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

## 6.5 MotorUnit Namespace Reference

### Classes

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

### Functions

- def [calcGCoupling](#)  
*Calculates the coupling conductance between two compartments.*
- def [compGCouplingMatrix](#)  
*Computes the Coupling Matrix to be used in the dVdt function of the N compartments of the motor unit.*
- def [runge\\_kutta](#)  
*Function to implement the fourth order Runge-Kutta Method to solve numerically a differential equation.*

### 6.5.1 Function Documentation

#### 6.5.1.1 def MotorUnit.calcGCoupling ( cytR, IComp1, IComp2, dComp1, dComp2 )

Calculates the coupling conductance between two compartments.

- Inputs:
  - **cytR**: Cytoplasmatic resistivity in  $\Omega.cm$ .
  - **IComp1, IComp2**: length of the compartments in  $\mu m$ .
  - **dComp1, dComp2**: diameter of the compartments in  $\mu m$ .
- Output:



- coupling conductance in MS.

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

$$g_c = \frac{2 \cdot 10^2}{\frac{R_{\text{cyl}} l_1}{\pi r_1^2} + \frac{R_{\text{cyl}} l_2}{\pi r_2^2}} \quad (6.2)$$

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

Definition at line 46 of file MotorUnit.py.

#### 6.5.1.2 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

$$GC = \begin{bmatrix} -g_c[0] & g_c[0] & 0 & \dots & \dots & 0 & 0 & 0 \\ g_c[0] & -g_c[0] - g_c[1] & g_c[1] & 0 & \dots & \dots & 0 & 0 \\ \vdots & & \ddots & & & & & \\ 0 & \dots & g_c[i-1] & -g_c[i-1] - g_c[i] & g_c[i] & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & \dots & & g_c[N-2] & -g_c[N-2] - g_c[N-1] & g_c[N-1] & 0 \\ 0 & \dots & 0 & & & 0 & g_c[N-1] & -g_c[N-1] \end{bmatrix} \quad (6.3)$$

Definition at line 78 of file MotorUnit.py.

#### 6.5.1.3 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:

$$\frac{dx(t)}{dt} = f(t, x(t)) \quad (6.4)$$

First, four derivatives are computed:

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

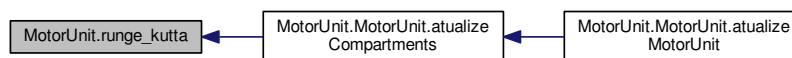
where  $\Delta t$  is the time step of the numerical solution of the differential equation.

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

$$x(t + \Delta t) = x(t) + \frac{\Delta t}{6} (k_1 + 2k_2 + 2k_3 + k_4) \quad (6.5)$$

Definition at line 133 of file MotorUnit.py.

Here is the caller graph for this function:



## 6.6 MotorUnitPool Namespace Reference

### Classes

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

### Functions

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

#### 6.6.1 Function Documentation

##### 6.6.1.1 def MotorUnitPool.twitchSaturation ( force, b )

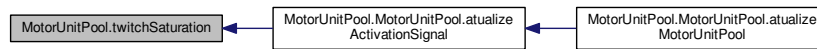
Computes the muscle unit force after the nonlinear saturation.

$$F_{sat} = \frac{1 - e^{-b \cdot force}}{1 + e^{-b \cdot force}} \quad (6.6)$$

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

Definition at line 31 of file MotorUnitPool.py.

Here is the caller graph for this function:



## 6.7 NeuralTract Namespace Reference

### Classes

- class [NeuralTract](#)  
*classdocs*

## 6.8 NeuralTractUnit Namespace Reference

### Classes

- class [NeuralTractUnit](#)  
*classdocs*

## 6.9 PointProcessGenerator Namespace Reference

### Classes

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

### Functions

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

#### 6.9.1 Function Documentation

6.9.1.1 `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.

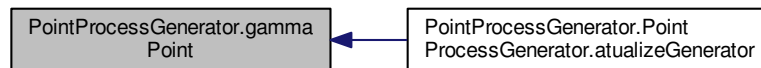
The number is generated according to:

$$\Gamma = -\frac{1}{\lambda} \ln\left(\prod_{i=1}^{\lambda} U(0, 1)\right) \quad (6.7)$$

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

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

Here is the caller graph for this function:



## 6.10 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](#)

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

- def [compValOff](#)

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

### 6.10.1 Function Documentation

#### 6.10.1.1 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:

$$v(t) = 1 + (v_0 - 1) \exp[-\alpha(t - t_0)] \quad (6.8)$$

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`.

6.10.1.2 `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:

$$v(t) = v_0 \exp[-\beta(t - t_0)] \quad (6.9)$$

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`.

## 6.11 simulation Namespace Reference

### Functions

- `def simulador`  
*Main example function.*

#### 6.11.1 Function Documentation

6.11.1.1 `def simulation.simulador ( )`

Main example function.

Definition at line 24 of file `simulation.py`.

## 6.12 Synapse Namespace Reference

### Classes

- `class Synapse`  
*classdocs*

### Functions

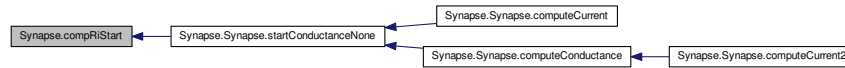
- `def compSynapCond`  
*Computes the synaptic conductance.*
- `def compRon`
- `def compRoff`
- `def compRiStart`
- `def compRiStop`
- `def compRonStart`
- `def compRoffStart`
- `def compRonStop`
- `def compRoffStop`

## 6.12.1 Function Documentation

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

Definition at line 24 of file Synapse.py.

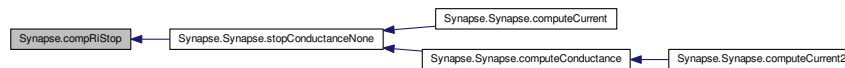
Here is the caller graph for this function:



### 6.12.1.2 `def Synapse.compRiStop ( rInf, ri, expFinish )`

Definition at line 27 of file Synapse.py.

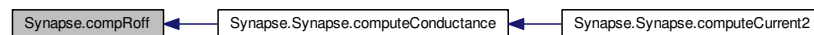
Here is the caller graph for this function:



### 6.12.1.3 `def Synapse.compRoff ( roff, t0, t, tauOff )`

Definition at line 21 of file Synapse.py.

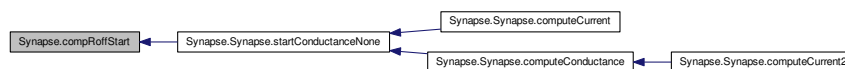
Here is the caller graph for this function:



### 6.12.1.4 `def Synapse.compRoffStart ( Roff, ri, synContrib )`

Definition at line 33 of file Synapse.py.

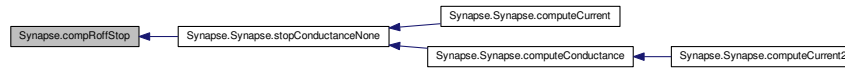
Here is the caller graph for this function:



#### 6.12.1.5 `def Synapse.compRoffStop ( Roff, ri, synContrib )`

Definition at line 39 of file Synapse.py.

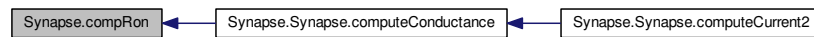
Here is the caller graph for this function:



#### 6.12.1.6 `def Synapse.compRon ( Non, rlnf, ron, t0, t, tauOn )`

Definition at line 18 of file Synapse.py.

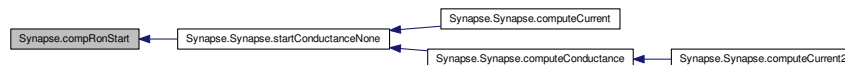
Here is the caller graph for this function:



#### 6.12.1.7 `def Synapse.compRonStart ( Ron, ri, synContrib )`

Definition at line 30 of file Synapse.py.

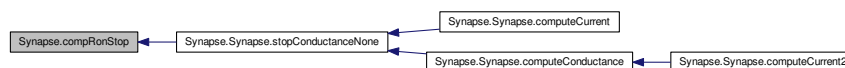
Here is the caller graph for this function:



#### 6.12.1.8 `def Synapse.compRonStop ( Ron, ri, synContrib )`

Definition at line 36 of file Synapse.py.

Here is the caller graph for this function:

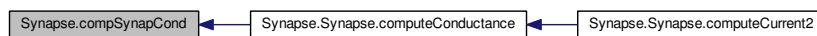


#### 6.12.1.9 `def Synapse.compSynapCond ( Gmax, Ron, Roff )`

Computes the synaptic conductance.

Definition at line 15 of file Synapse.py.

Here is the caller graph for this function:



## 6.13 SynapsesFactory Namespace Reference

### Classes

- class [SynapsesFactory](#)  
*classdocs*



# Chapter 7

## Class Documentation

### 7.1 AxonDelay.AxonDelay Class Reference

Class that implements a delay correspondent to the nerve.

#### Public Member Functions

- `def \_\_init\_\_`  
*Constructor.*
- `def addTerminalSpike`  
*Indicates to the [AxonDelay](#) object that a spike has occurred in the Terminal.*
- `def addSpinalSpike`  
*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.*

### 7.1.1 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.

### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 `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 34 of file AxonDelay.py.

### 7.1.3 Member Function Documentation

#### 7.1.3.1 `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 75 of file AxonDelay.py.

Here is the call graph for this function:



#### 7.1.3.2 `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 64 of file AxonDelay.py.

Here is the caller graph for this function:



## 7.1.4 Member Data Documentation

### 7.1.4.1 AxonDelay.AxonDelay.index

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

Definition at line 38 of file AxonDelay.py.

### 7.1.4.2 AxonDelay.AxonDelay.latencySpinalTerminal\_ms

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

Definition at line 49 of file AxonDelay.py.

### 7.1.4.3 AxonDelay.AxonDelay.latencyStimulusSpinal\_ms

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

Definition at line 47 of file AxonDelay.py.

### 7.1.4.4 AxonDelay.AxonDelay.latencyStimulusTerminal\_ms

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

Definition at line 51 of file AxonDelay.py.

### 7.1.4.5 AxonDelay.AxonDelay.length\_m

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

Definition at line 41 of file AxonDelay.py.

### 7.1.4.6 AxonDelay.AxonDelay.stimulusPositiontoTerminal

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

Definition at line 45 of file AxonDelay.py.

#### 7.1.4.7 AxonDelay.AxonDelay.terminalSpikeTrain

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

Definition at line 54 of file AxonDelay.py.

#### 7.1.4.8 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 43 of file AxonDelay.py.

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

- [AxonDelay.py](#)

## 7.2 ChannelConductance.ChannelConductance Class Reference

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

### Public Member Functions

- `def __init__`  
*Builds an ionic channel conductance.*
- `def computeCurrent`  
*Computes the current generated by the ionic Channel.*
- `def compCondKf`  
*Computes the conductance of a Kf Channel.*
- `def compCondKs`  
*Computes the conductance of a Ks Channel.*
- `def compCondNa`  
*Computes the conductance of a Na Channel.*

### Public Attributes

- `kind`  
*String with the type of the ionic channel (Na, Ks, Kf or Ca).*
- `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  $\mu S$ , 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.*

### 7.2.1 Detailed Description

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

Definition at line 16 of file ChannelConductance.py.

### 7.2.2 Constructor & Destructor Documentation

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

Builds an ionic channel conductance.

-Inputs:

- **kind**: string with the type of the ionic channel (Na, Ks, Kf or Ca).
- **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 30 of file ChannelConductance.py.

### 7.2.3 Member Function Documentation

7.2.3.1 `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  $\mu S$ .

Definition at line 97 of file ChannelConductance.py.

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

Computes the conductance of a Ks 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  $\mu S$ .

Definition at line 112 of file ChannelConductance.py.

### 7.2.3.3 `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 \form#2S.
```

Definition at line 126 of file `ChannelConductance.py`.

### 7.2.3.4 `def ChannelConductance.ChannelConductance.computeCurrent ( self, t, V_mV )`

Computes the current genrated 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 81 of file `ChannelConductance.py`.

## 7.2.4 Member Data Documentation

### 7.2.4.1 `ChannelConductance.ChannelConductance.compCond`

Function that computes the conductance dynamics.

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

### 7.2.4.2 `ChannelConductance.ChannelConductance.condState`

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

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

### 7.2.4.3 `ChannelConductance.ChannelConductance.EqPot_mV`

Equilibrium Potential of the ionic channel, mV.

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

### 7.2.4.4 `ChannelConductance.ChannelConductance.gmax_muS`

Maximal conductance, in  $\mu S$ , of the ionic channel.

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

### 7.2.4.5 `ChannelConductance.ChannelConductance.kind`

String with the type of the ionic channel (Na, Ks, Kf or Ca).

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

## 7.2.4.6 ChannelConductance.ChannelConductance.lenStates

Integer with the number of states in the ionic channel.

Definition at line 64 of file ChannelConductance.py.

## 7.2.4.7 ChannelConductance.ChannelConductance.stateType

String with type of dynamics of the states.

For now it accepts the string pulse.

Definition at line 42 of file ChannelConductance.py.

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

- [ChannelConductance.py](#)

## 7.3 Compartment.Compartment Class Reference

Class that implements a neural compartment.

### Public Member Functions

- `def __init__`  
*Constructor.*
- `def computeCurrent`  
*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  $\mu\text{m}$ .*
- `diameter_mum`  
*Diameter of the compartment, in  $\mu\text{m}$ .*
- `capacitance_nF`  
*Capacitance of the compartment, in nF.*
- `gLeak`  
*Leak conductance of the compartment, in MS.*
- `numberChannels`  
*Integer with the number of ionic channels.*

### 7.3.1 Detailed Description

Class that implements a neural compartment.

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

Definition at line 40 of file Compartment.py.

### 7.3.2 Constructor & Destructor Documentation

7.3.2.1 `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.

### 7.3.3 Member Function Documentation

7.3.3.1 `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 116 of file Compartment.py.

### 7.3.4 Member Data Documentation

7.3.4.1 `Compartment.Compartment.capacitance_nF`

Capacitance of the compartment, in nF.

Definition at line 89 of file Compartment.py.

7.3.4.2 `Compartment.Compartment.Channels`

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

Definition at line 63 of file Compartment.py.



#### 7.3.4.3 Compartment.Compartment.diameter\_mum

Diameter of the compartment, in  $\mu\text{m}$ .

Definition at line 85 of file Compartment.py.

#### 7.3.4.4 Compartment.Compartment.gLeak

Leak conductance of the compartment, in MS.

Definition at line 91 of file Compartment.py.

#### 7.3.4.5 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.

#### 7.3.4.6 Compartment.Compartment.kind

The kind of compartment.

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

Definition at line 76 of file Compartment.py.

#### 7.3.4.7 Compartment.Compartment.length\_mum

Length of the compartment, in  $\mu\text{m}$ .

Definition at line 83 of file Compartment.py.

#### 7.3.4.8 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.

#### 7.3.4.9 Compartment.Compartment.numberChannels

Integer with the number of ionic channels.

Definition at line 102 of file Compartment.py.

#### 7.3.4.10 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.

#### 7.3.4.11 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](#)

## 7.4 Configuration.Configuration Class Reference

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

### Public Member Functions

- `def __init__`  
*Constructor.*
- `def parameterSet`  
*Function that returns the value of wished parameter specified in the paramTag variable.*
- `def inputFunctionGet`  
*Returns a numpy array with the values of the function for the whole simulation.*
- `def determineSynapses`  
*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 computaional efficiency.*
- `timeStepBySix_ms`  
*The variable timeStep divided by six, for computaional efficiency.*

#### 7.4.1 Detailed Description

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

Definition at line 15 of file Configuration.py.

#### 7.4.2 Constructor & Destructor Documentation

##### 7.4.2.1 `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 29 of file Configuration.py.

### 7.4.3 Member Function Documentation

#### 7.4.3.1 `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 130 of file Configuration.py.

#### 7.4.3.2 `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 114 of file Configuration.py.

#### 7.4.3.3 `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 rmt0 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 70 of file Configuration.py.

### 7.4.4 Member Data Documentation

#### 7.4.4.1 `Configuration.Configuration.confArray`

An array with all the simulation parameters.

Definition at line 32 of file Configuration.py.

#### 7.4.4.2 Configuration.Configuration.simDuration\_ms

Total length of the simulation in ms.

Definition at line 42 of file Configuration.py.

#### 7.4.4.3 Configuration.Configuration.timeStep\_ms

Time step of the numerical solution of the differential equation.

Definition at line 39 of file Configuration.py.

#### 7.4.4.4 Configuration.Configuration.timeStepBySix\_ms

The variable timeStep divided by six, for computaional efficiency.

Definition at line 46 of file Configuration.py.

#### 7.4.4.5 Configuration.Configuration.timeStepByTwo\_ms

The variable timeStep divided by two, for computaional efficiency.

Definition at line 44 of file Configuration.py.

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

- [Configuration.py](#)

## 7.5 MotorUnit.MotorUnit Class Reference

Class that implements a motor unit model.

### Public Member Functions

- `def __init__`  
*Constructor.*
- `def atualizeMotorUnit`  
*Atualize the dynamical and nondynamical (delay) parts of the motor unit.*
- `def atualizeCompartments`  
*Atualize all neural compartments.*
- `def dVdt`  
*Compute the potential derivative of all compartments of the motor unit.*
- `def addSomaSpike`  
*When the soma potential is above the threshold a spike is added tom the soma.*
- `def atualizeDelay`  
*Atualize the terminal spike train, by considering the Delay of the nerve.*

### Public Attributes

- `conf`  
*Configuration object with the simulation parameters.*
- `kind`  
*String with the type of the motor unit.*

- [tSomaSpike](#)  
*The instant of the last spie 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.*
- [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.*
- [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](#)  
*Amplutude of the muscle unit twitch, in N.*
- [bSat](#)  
*Parameter of the saturation.*
- [twTet](#)  
*Twitch- tetanus relationship.*

### 7.5.1 Detailed Description

Class that implements a motor unit model.

Encompasses a motoneuron and a muscle unit.

Definition at line 147 of file MotorUnit.py.

## 7.5.2 Constructor & Destructor Documentation

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

Constructor.

- Inputs:
  - **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).
  - **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 165 of file MotorUnit.py.

## 7.5.3 Member Function Documentation

### 7.5.3.1 `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 316 of file MotorUnit.py.

Here is the caller graph for this function:



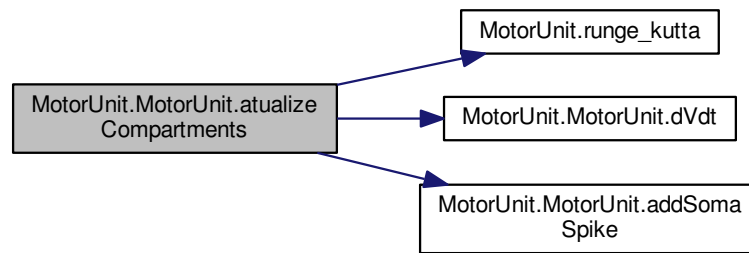
### 7.5.3.2 `def MotorUnit.MotorUnit.atualizeCompartments ( self, t )`

Atualize all neural compartments.

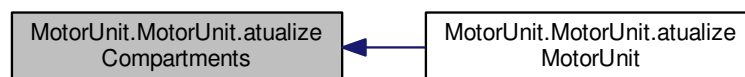
- Inputs:
  - **t**: current instant, in ms.

Definition at line 286 of file MotorUnit.py.

Here is the call graph for this function:



Here is the caller graph for this function:



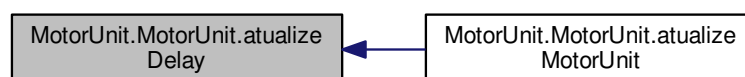
#### 7.5.3.3 `def MotorUnit.MotorUnit.atualizeDelay ( self, t )`

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

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

Definition at line 332 of file `MotorUnit.py`.

Here is the caller graph for this function:



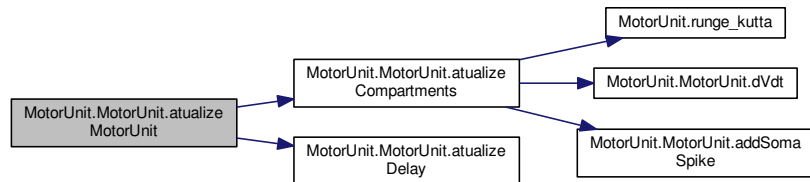
#### 7.5.3.4 `def MotorUnit.MotorUnit.atualizeMotorUnit ( self, t )`

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

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

Definition at line 274 of file MotorUnit.py.

Here is the call graph for this function:



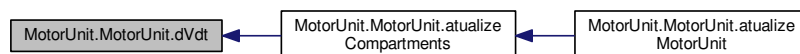
#### 7.5.3.5 `def MotorUnit.MotorUnit.dVdt ( self, t, V )`

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 301 of file MotorUnit.py.

Here is the caller graph for this function:



## 7.5.4 Member Data Documentation

### 7.5.4.1 `MotorUnit.MotorUnit.bSat`

Parameter of the saturation.

Definition at line 259 of file MotorUnit.py.

### 7.5.4.2 `MotorUnit.MotorUnit.capacitanceInv`

Vector with the inverse of the capacitance of all compartments.

Definition at line 211 of file MotorUnit.py.



#### 7.5.4.3 MotorUnit.MotorUnit.compartment

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

Definition at line 182 of file MotorUnit.py.

#### 7.5.4.4 MotorUnit.MotorUnit.compNumber

Number of compartments.

Definition at line 189 of file MotorUnit.py.

#### 7.5.4.5 MotorUnit.MotorUnit.conf

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

Definition at line 168 of file MotorUnit.py.

#### 7.5.4.6 MotorUnit.MotorUnit.Delay

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

Definition at line 244 of file MotorUnit.py.

#### 7.5.4.7 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 226 of file MotorUnit.py.

#### 7.5.4.8 MotorUnit.MotorUnit.ilnjected

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

Definition at line 217 of file MotorUnit.py.

#### 7.5.4.9 MotorUnit.MotorUnit.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 215 of file MotorUnit.py.

#### 7.5.4.10 MotorUnit.MotorUnit.index

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

Definition at line 180 of file MotorUnit.py.

#### 7.5.4.11 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 171 of file MotorUnit.py.

#### 7.5.4.12 `MotorUnit.MotorUnit.MNRefPer_ms`

Refractory period, in ms, of the motoneuron.

Definition at line 233 of file `MotorUnit.py`.

#### 7.5.4.13 `MotorUnit.MotorUnit.nerve`

String with type of the nerve.

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

Definition at line 239 of file `MotorUnit.py`.

#### 7.5.4.14 `MotorUnit.MotorUnit.somaIndex`

index of the soma compartment.

Definition at line 230 of file `MotorUnit.py`.

#### 7.5.4.15 `MotorUnit.MotorUnit.somaSpikeTrain`

Vector with the instants of spikes at the soma.

Definition at line 178 of file `MotorUnit.py`.

#### 7.5.4.16 `MotorUnit.MotorUnit.terminalSpikeTrain`

Vector with the instants of spikes at the terminal.

Definition at line 248 of file `MotorUnit.py`.

#### 7.5.4.17 `MotorUnit.MotorUnit.threshold_mV`

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

Definition at line 184 of file `MotorUnit.py`.

#### 7.5.4.18 `MotorUnit.MotorUnit.tSomaSpike`

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

Definition at line 175 of file `MotorUnit.py`.

#### 7.5.4.19 `MotorUnit.MotorUnit.TwitchAmp_N`

Amplitude of the muscle unit twitch, in N.

Definition at line 257 of file `MotorUnit.py`.

#### 7.5.4.20 `MotorUnit.MotorUnit.TwitchTc_ms`

Contraction time of the twitch muscle unit, in ms.

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

## 7.5.4.21 MotorUnit.MotorUnit.twTet

Twitch- tetanus relationship.

Definition at line 261 of file MotorUnit.py.

## 7.5.4.22 MotorUnit.MotorUnit.v\_mV

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

Definition at line 191 of file MotorUnit.py.

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

- [MotorUnit.py](#)

## 7.6 MotorUnitPool.MotorUnitPool Class Reference

Class that implements a motor unit pool.

### Public Member Functions

- [def \\_\\_init\\_\\_](#)  
*Constructor.*
- [def atualizeMotorUnitPool](#)
- [def atualizeActivationSignal](#)
- [def atualizeForceNoHill](#)  
*Compute the muscle force when no muscle dynamics (Hill model) is used.*
- [def listSpikes](#)

### 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.*
- [activationModel](#)  
*Model of the activation signal.*
- [ActMatrix](#)
- [an](#)
- [activation\\_nonSat](#)
- [bSat](#)

- [twTet](#)
- [twitchAmp\\_N](#)
- [activation\\_Sat](#)
- [diracDeltaValue](#)
- [force](#)
- [hillModel](#)
- [atualizeForce](#)
- [timeIndex](#)

### 7.6.1 Detailed Description

Class that implements a motor unit pool.

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

Definition at line 40 of file MotorUnitPool.py.

### 7.6.2 Constructor & Destructor Documentation

7.6.2.1 `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.

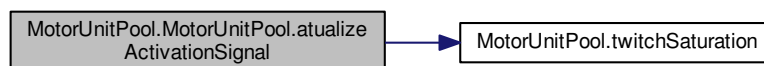
Definition at line 52 of file MotorUnitPool.py.

### 7.6.3 Member Function Documentation

7.6.3.1 `def MotorUnitPool.MotorUnitPool.atualizeActivationSignal( self, t )`

Definition at line 136 of file MotorUnitPool.py.

Here is the call graph for this function:



Here is the caller graph for this function:



### 7.6.3.2 `def MotorUnitPool.MotorUnitPool.atualizeForceNoHill ( self )`

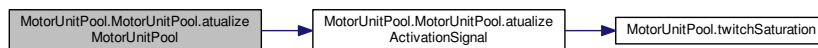
Compute the muscle force when no muscle dynamics (Hill model) is used.

Definition at line 152 of file MotorUnitPool.py.

### 7.6.3.3 `def MotorUnitPool.MotorUnitPool.atualizeMotorUnitPool ( self, t )`

Definition at line 129 of file MotorUnitPool.py.

Here is the call graph for this function:



### 7.6.3.4 `def MotorUnitPool.MotorUnitPool.listSpikes ( self )`

Definition at line 157 of file MotorUnitPool.py.

## 7.6.4 Member Data Documentation

### 7.6.4.1 `MotorUnitPool.MotorUnitPool.activation_nonSat`

Definition at line 105 of file MotorUnitPool.py.

### 7.6.4.2 `MotorUnitPool.MotorUnitPool.activation_Sat`

Definition at line 114 of file MotorUnitPool.py.

### 7.6.4.3 `MotorUnitPool.MotorUnitPool.activationModel`

Model of the activation signal.

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

Definition at line 86 of file MotorUnitPool.py.

### 7.6.4.4 `MotorUnitPool.MotorUnitPool.ActMatrix`

Definition at line 94 of file MotorUnitPool.py.

### 7.6.4.5 `MotorUnitPool.MotorUnitPool.an`

Definition at line 103 of file MotorUnitPool.py.

### 7.6.4.6 `MotorUnitPool.MotorUnitPool.atualizeForce`

Definition at line 121 of file MotorUnitPool.py.

#### 7.6.4.7 MotorUnitPool.MotorUnitPool.bSat

Definition at line 106 of file MotorUnitPool.py.

#### 7.6.4.8 MotorUnitPool.MotorUnitPool.conf

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

Definition at line 58 of file MotorUnitPool.py.

#### 7.6.4.9 MotorUnitPool.MotorUnitPool.diracDeltaValue

Definition at line 116 of file MotorUnitPool.py.

#### 7.6.4.10 MotorUnitPool.MotorUnitPool.force

Definition at line 119 of file MotorUnitPool.py.

#### 7.6.4.11 MotorUnitPool.MotorUnitPool.hillModel

Definition at line 120 of file MotorUnitPool.py.

#### 7.6.4.12 MotorUnitPool.MotorUnitPool.kind

Indicates that is Motor Unit pool.

Definition at line 55 of file MotorUnitPool.py.

#### 7.6.4.13 MotorUnitPool.MotorUnitPool.MUnumber

Number of motor units.

Definition at line 65 of file MotorUnitPool.py.

#### 7.6.4.14 MotorUnitPool.MotorUnitPool.pool

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

Definition at line 60 of file MotorUnitPool.py.

#### 7.6.4.15 MotorUnitPool.MotorUnitPool.poolSomaSpikes

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

Definition at line 80 of file MotorUnitPool.py.

#### 7.6.4.16 MotorUnitPool.MotorUnitPool.poolTerminalSpikes

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

Definition at line 82 of file MotorUnitPool.py.

#### 7.6.4.17 MotorUnitPool.MotorUnitPool.timeIndex

Definition at line 123 of file MotorUnitPool.py.

#### 7.6.4.18 MotorUnitPool.MotorUnitPool.twitchAmp\_N

Definition at line 108 of file MotorUnitPool.py.

#### 7.6.4.19 MotorUnitPool.MotorUnitPool.twTet

Definition at line 107 of file MotorUnitPool.py.

#### 7.6.4.20 MotorUnitPool.MotorUnitPool.unit

List of [MotorUnit](#) objects.

Definition at line 68 of file MotorUnitPool.py.

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

- [MotorUnitPool.py](#)

## 7.7 NeuralTract.NeuralTract Class Reference

classdocs

### Public Member Functions

- def [\\_\\_init\\_\\_](#)  
*Constructor.*
- def [atualizePool](#)
- def [listSpikes](#)

### Public Attributes

- [kind](#)
- [pool](#)
- [Number](#)
- [unit](#)
- [poolTerminalSpikes](#)
- [target](#)
- [FR](#)
- [timeIndex](#)

#### 7.7.1 Detailed Description

classdocs

Definition at line 14 of file NeuralTract.py.

## 7.7.2 Constructor & Destructor Documentation

### 7.7.2.1 `def NeuralTract.NeuralTract.__init__( self, conf, pool )`

Constructor.

- Inputs:
  - **conf**:
  - **pool**:

Definition at line 26 of file NeuralTract.py.

## 7.7.3 Member Function Documentation

### 7.7.3.1 `def NeuralTract.NeuralTract.atualizePool( self, t )`

Definition at line 50 of file NeuralTract.py.

### 7.7.3.2 `def NeuralTract.NeuralTract.listSpikes( self )`

Definition at line 55 of file NeuralTract.py.

## 7.7.4 Member Data Documentation

### 7.7.4.1 `NeuralTract.NeuralTract.FR`

Definition at line 43 of file NeuralTract.py.

### 7.7.4.2 `NeuralTract.NeuralTract.kind`

Definition at line 27 of file NeuralTract.py.

### 7.7.4.3 `NeuralTract.NeuralTract.Number`

Definition at line 29 of file NeuralTract.py.

### 7.7.4.4 `NeuralTract.NeuralTract.pool`

Definition at line 28 of file NeuralTract.py.

### 7.7.4.5 `NeuralTract.NeuralTract.poolTerminalSpikes`

Definition at line 34 of file NeuralTract.py.

### 7.7.4.6 `NeuralTract.NeuralTract.target`

Definition at line 36 of file NeuralTract.py.



## 7.7.4.7 NeuralTract.NeuralTract.timeIndex

Definition at line 46 of file NeuralTract.py.

## 7.7.4.8 NeuralTract.NeuralTract.unit

Definition at line 31 of file NeuralTract.py.

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

- [NeuralTract.py](#)

## 7.8 NeuralTractUnit.NeuralTractUnit Class Reference

classdocs

### Public Member Functions

- [def \\_\\_init\\_\\_](#)  
*Constructor.*
- [def atualizeNeuralTractUnit](#)
- [def transmitSpikes](#)

### Public Attributes

- [GammaOrder](#)
- [spikesGenerator](#)
- [terminalSpikeTrain](#)
- [SynapsesOut](#)
- [transmitSpikesThroughSynapses](#)
- [indicesOfSynapsesOnTarget](#)

### 7.8.1 Detailed Description

classdocs

Definition at line 20 of file NeuralTractUnit.py.

### 7.8.2 Constructor & Destructor Documentation

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

Constructor.

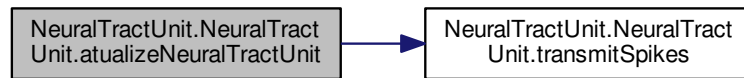
Definition at line 27 of file NeuralTractUnit.py.

### 7.8.3 Member Function Documentation

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

Definition at line 49 of file NeuralTractUnit.py.

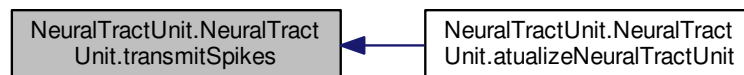
Here is the call graph for this function:



**7.8.3.2** `def NeuralTractUnit.NeuralTractUnit.transmitSpikes ( self, t )`

Definition at line 59 of file `NeuralTractUnit.py`.

Here is the caller graph for this function:



## 7.8.4 Member Data Documentation

### 7.8.4.1 `NeuralTractUnit.NeuralTractUnit.GammaOrder`

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

### 7.8.4.2 `NeuralTractUnit.NeuralTractUnit.indicesOfSynapsesOnTarget`

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

### 7.8.4.3 `NeuralTractUnit.NeuralTractUnit.spikesGenerator`

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

### 7.8.4.4 `NeuralTractUnit.NeuralTractUnit.SynapsesOut`

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

### 7.8.4.5 `NeuralTractUnit.NeuralTractUnit.terminalSpikeTrain`

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

#### 7.8.4.6 NeuralTractUnit.NeuralTractUnit.transmitSpikesThroughSynapses

Definition at line 40 of file NeuralTractUnit.py.

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

- [NeuralTractUnit.py](#)

## 7.9 PointProcessGenerator.PointProcessGenerator Class Reference

Generator of point processes.

### Public Member Functions

- `def __init__`  
*Constructor.*
- `def atualizeGenerator`

### 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.*

### 7.9.1 Detailed Description

Generator of point processes.

Definition at line 46 of file PointProcessGenerator.py.

### 7.9.2 Constructor & Destructor Documentation

#### 7.9.2.1 `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 57 of file PointProcessGenerator.py.

### 7.9.3 Member Function Documentation

#### 7.9.3.1 `def PointProcessGenerator.PointProcessGenerator.atualizeGenerator ( self, t, firingRate )`

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

Definition at line 86 of file `PointProcessGenerator.py`.

Here is the call graph for this function:



### 7.9.4 Member Data Documentation

#### 7.9.4.1 `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 60 of file `PointProcessGenerator.py`.

#### 7.9.4.2 `PointProcessGenerator.PointProcessGenerator.GammaOrderInv`

Inverse of the GammaOrder.

This is necessary for computational efficiency.

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

#### 7.9.4.3 `PointProcessGenerator.PointProcessGenerator.index`

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

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

#### 7.9.4.4 `PointProcessGenerator.PointProcessGenerator.points`

List of spike instants of the generator.

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

#### 7.9.4.5 `PointProcessGenerator.PointProcessGenerator.threshold`

Spike threshold.

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

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

## 7.9.4.6 PointProcessGenerator.PointProcessGenerator.y

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

Definition at line 70 of file PointProcessGenerator.py.

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

- [PointProcessGenerator.py](#)

## 7.10 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\\_\\_](#)  
*Initializes the pulse conductance state.*
- def [changeState](#)  
*Void function that modify the current situation (true/false) of the state.*
- def [computeStateValue](#)  
*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](#)

## 7.10.1 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.

## 7.10.2 Constructor &amp; Destructor Documentation

## 7.10.2.1 def PulseConductanceState.PulseConductanceState.\_\_init\_\_( self, kind, conf, pool, 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. index - the index of the unit that this state belongs.

Definition at line 65 of file PulseConductanceState.py.

### 7.10.3 Member Function Documentation

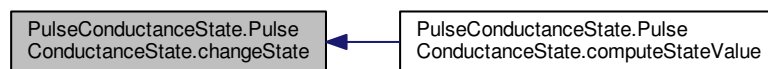
#### 7.10.3.1 `def PulseConductanceState.PulseConductanceState.changeState ( self, t )`

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

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

Definition at line 104 of file `PulseConductanceState.py`.

Here is the caller graph for this function:



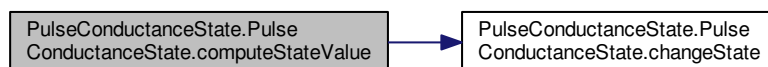
#### 7.10.3.2 `def PulseConductanceState.PulseConductanceState.computeStateValue ( self, t )`

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 116 of file `PulseConductanceState.py`.

Here is the call graph for this function:



### 7.10.4 Member Data Documentation

#### 7.10.4.1 `PulseConductanceState.PulseConductanceState.actType`

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

#### 7.10.4.2 `PulseConductanceState.PulseConductanceState.alpha_ms1`

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

#### 7.10.4.3 `PulseConductanceState.PulseConductanceState.beta_ms1`

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

#### 7.10.4.4 `PulseConductanceState.PulseConductanceState.computeValueOff`

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

#### 7.10.4.5 `PulseConductanceState.PulseConductanceState.computeValueOn`

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

#### 7.10.4.6 `PulseConductanceState.PulseConductanceState.kind`

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

#### 7.10.4.7 `PulseConductanceState.PulseConductanceState.PulseDur_ms`

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

#### 7.10.4.8 `PulseConductanceState.PulseConductanceState.state`

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

#### 7.10.4.9 `PulseConductanceState.PulseConductanceState.t0`

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

#### 7.10.4.10 `PulseConductanceState.PulseConductanceState.v0`

Definition at line 70 of file `PulseConductanceState.py`.

#### 7.10.4.11 `PulseConductanceState.PulseConductanceState.value`

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

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

- [PulseConductanceState.py](#)

## 7.11 Synapse.Synapse Class Reference

classdocs

### Public Member Functions

- def [\\_\\_init\\_\\_](#)  
*Constructor.*
- def [computeCurrent](#)

- def [computeCurrent2](#)
- def [computeConductance](#)
- def [startConductanceNone](#)
- def [startConductanceDynamics](#)
- def [stopConductanceNone](#)
- def [stopConductanceDynamics](#)
- def [receiveSpike](#)
- def [addConductance](#)

### Public Attributes

- [pool](#)
- [kind](#)
- [neuronKind](#)
- [EqPot\\_mV](#)
- [alpha\\_ms1](#)
- [beta\\_ms1](#)
- [Tmax\\_mM](#)
- [tPeak\\_ms](#)
- [gmax\\_muS](#)
- [delay\\_ms](#)
- [dynamics](#)
- [gMaxTot\\_muS](#)
- [numberOfIncomingSynapses](#)
- [rInf](#)
- [tauOn](#)
- [tauOff](#)
- [expFinish](#)
- [Non](#)
- [Ron](#)
- [ron](#)
- [Roff](#)
- [roff](#)
- [t0](#)
- [spikesReceived](#)
- [conductanceState](#)
- [tBeginOfPulse](#)
- [tEndOfPulse](#)
- [ri](#)
- [ti](#)
- [synContrib](#)
- [startDynamicFunction](#)
- [stopDynamicFunction](#)
- [startEntrance](#)
- [stopEntrance](#)
- [computeCurrent](#)

#### 7.11.1 Detailed Description

classdocs

Definition at line 46 of file Synapse.py.



## 7.11.2 Constructor & Destructor Documentation

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

Constructor.

Definition at line 51 of file Synapse.py.

## 7.11.3 Member Function Documentation

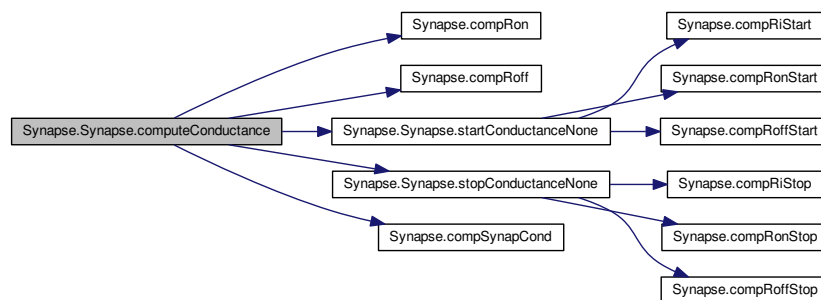
7.11.3.1 `def Synapse.Synapse.addConductance( self, gmax, delay, dynamics, weight )`

Definition at line 193 of file Synapse.py.

7.11.3.2 `def Synapse.Synapse.computeConductance( self, t )`

Definition at line 128 of file Synapse.py.

Here is the call graph for this function:



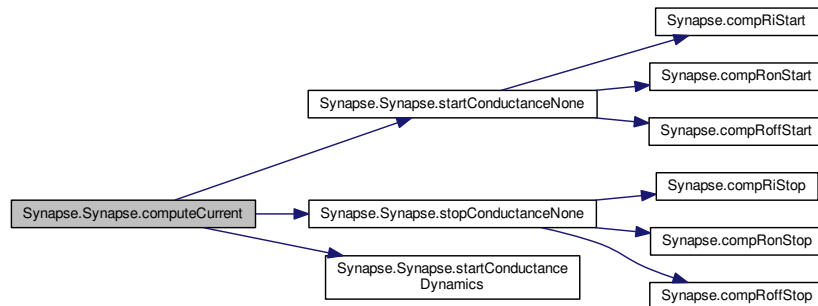
Here is the caller graph for this function:



7.11.3.3 `def Synapse.Synapse.computeCurrent( self, t, V_mV )`

Definition at line 100 of file Synapse.py.

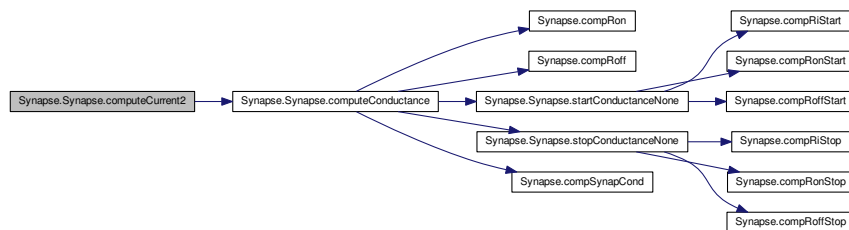
Here is the call graph for this function:



#### 7.11.3.4 def Synapse.Synapse.computeCurrent2 ( self, t, V\_mV )

Definition at line 119 of file Synapse.py.

Here is the call graph for this function:



#### 7.11.3.5 def Synapse.Synapse.receiveSpike ( self, t, synapseNumber )

Definition at line 188 of file Synapse.py.

#### 7.11.3.6 def Synapse.Synapse.startConductanceDynamics ( self, t, synapsesNumber )

Definition at line 161 of file Synapse.py.

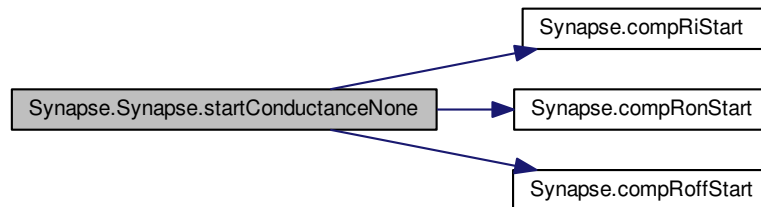
Here is the caller graph for this function:



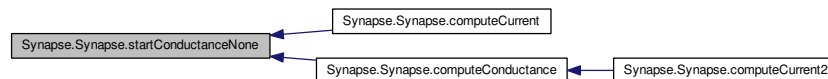
### 7.11.3.7 `def Synapse.Synapse.startConductanceNone ( self, t, idxBeginPulse )`

Definition at line 143 of file Synapse.py.

Here is the call graph for this function:



Here is the caller graph for this function:



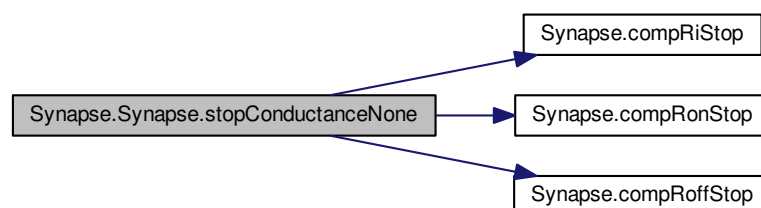
### 7.11.3.8 `def Synapse.Synapse.stopConductanceDynamics ( self, t, synapseNumber )`

Definition at line 182 of file Synapse.py.

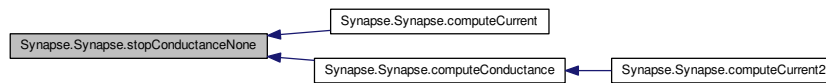
### 7.11.3.9 `def Synapse.Synapse.stopConductanceNone ( self, t, idxEndPulse )`

Definition at line 166 of file Synapse.py.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 7.11.4 Member Data Documentation

##### 7.11.4.1 `Synapse.Synapse.alpha_ms1`

Definition at line 57 of file `Synapse.py`.

##### 7.11.4.2 `Synapse.Synapse.beta_ms1`

Definition at line 58 of file `Synapse.py`.

##### 7.11.4.3 `Synapse.Synapse.computeCurrent`

Definition at line 113 of file `Synapse.py`.

##### 7.11.4.4 `Synapse.Synapse.conductanceState`

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

##### 7.11.4.5 `Synapse.Synapse.delay_ms`

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

##### 7.11.4.6 `Synapse.Synapse.dynamics`

Definition at line 64 of file `Synapse.py`.

##### 7.11.4.7 `Synapse.Synapse.EqPot_mV`

Definition at line 56 of file `Synapse.py`.

##### 7.11.4.8 `Synapse.Synapse.expFinish`

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

##### 7.11.4.9 `Synapse.Synapse.gmax_muS`

Definition at line 62 of file `Synapse.py`.

##### 7.11.4.10 `Synapse.Synapse.gMaxTot_muS`

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

#### 7.11.4.11 Synapse.Synapse.kind

Definition at line 53 of file Synapse.py.

#### 7.11.4.12 Synapse.Synapse.neuronKind

Definition at line 54 of file Synapse.py.

#### 7.11.4.13 Synapse.Synapse.Non

Definition at line 76 of file Synapse.py.

#### 7.11.4.14 Synapse.Synapse.numberOfIncomingSynapses

Definition at line 67 of file Synapse.py.

#### 7.11.4.15 Synapse.Synapse.pool

Definition at line 52 of file Synapse.py.

#### 7.11.4.16 Synapse.Synapse.ri

Definition at line 88 of file Synapse.py.

#### 7.11.4.17 Synapse.Synapse.rInf

Definition at line 70 of file Synapse.py.

#### 7.11.4.18 Synapse.Synapse.Roff

Definition at line 79 of file Synapse.py.

#### 7.11.4.19 Synapse.Synapse.roff

Definition at line 80 of file Synapse.py.

#### 7.11.4.20 Synapse.Synapse.Ron

Definition at line 77 of file Synapse.py.

#### 7.11.4.21 Synapse.Synapse.ron

Definition at line 78 of file Synapse.py.

#### 7.11.4.22 Synapse.Synapse.spikesReceived

Definition at line 83 of file Synapse.py.

**7.11.4.23 Synapse.Synapse.startDynamicFunction**

Definition at line 91 of file Synapse.py.

**7.11.4.24 Synapse.Synapse.startEntrance**

Definition at line 94 of file Synapse.py.

**7.11.4.25 Synapse.Synapse.stopDynamicFunction**

Definition at line 92 of file Synapse.py.

**7.11.4.26 Synapse.Synapse.stopEntrance**

Definition at line 95 of file Synapse.py.

**7.11.4.27 Synapse.Synapse.synContrib**

Definition at line 90 of file Synapse.py.

**7.11.4.28 Synapse.Synapse.t0**

Definition at line 81 of file Synapse.py.

**7.11.4.29 Synapse.Synapse.tauOff**

Definition at line 72 of file Synapse.py.

**7.11.4.30 Synapse.Synapse.tauOn**

Definition at line 71 of file Synapse.py.

**7.11.4.31 Synapse.Synapse.tBeginOfPulse**

Definition at line 86 of file Synapse.py.

**7.11.4.32 Synapse.Synapse.tEndOfPulse**

Definition at line 87 of file Synapse.py.

**7.11.4.33 Synapse.Synapse.ti**

Definition at line 89 of file Synapse.py.

**7.11.4.34 Synapse.Synapse.Tmax\_mM**

Definition at line 59 of file Synapse.py.

## 7.11.4.35 Synapse.Synapse.tPeak\_ms

Definition at line 60 of file Synapse.py.

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

- [Synapse.py](#)

## 7.12 SynapsesFactory.SynapsesFactory Class Reference

classdocs

### Public Member Functions

- def [\\_\\_init\\_\\_](#)  
*Constructor.*

### Public Attributes

- [numberOfSynapses](#)

### 7.12.1 Detailed Description

classdocs

Definition at line 15 of file SynapsesFactory.py.

### 7.12.2 Constructor & Destructor Documentation

7.12.2.1 def SynapsesFactory.SynapsesFactory.\_\_init\_\_( self, conf, pools )

Constructor.

Definition at line 22 of file SynapsesFactory.py.

### 7.12.3 Member Data Documentation

7.12.3.1 SynapsesFactory.SynapsesFactory.numberOfSynapses

Definition at line 24 of file SynapsesFactory.py.

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

- [SynapsesFactory.py](#)





## Chapter 8

# File Documentation

### 8.1 AxonDelay.py File Reference

#### Classes

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

#### Namespaces

- [AxonDelay](#)

### 8.2 ChannelConductance.py File Reference

#### Classes

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

#### Namespaces

- [ChannelConductance](#)

### 8.3 Compartment.py File Reference

#### Classes

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

#### Namespaces

- [Compartment](#)

## Functions

- def [Compartment.calcGLeak](#)  
*Computes the leak conductance of the compartment.*

## 8.4 Configuration.py File Reference

### Classes

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

### Namespaces

- [Configuration](#)

## 8.5 MotorUnit.py File Reference

### Classes

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

### Namespaces

- [MotorUnit](#)

### Functions

- def [MotorUnit.calcGCoupling](#)  
*Calculates the coupling conductance between two compartments.*
- def [MotorUnit.compGCouplingMatrix](#)  
*Computes the Coupling Matrix to be used in the dVdt function of the N compartments of the motor unit.*
- def [MotorUnit.runge\\_kutta](#)  
*Function to implement the fourth order Runge-Kutta Method to solve numerically a differential equation.*

## 8.6 MotorUnitPool.py File Reference

### Classes

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

### Namespaces

- [MotorUnitPool](#)

## Functions

- def [MotorUnitPool.twitchSaturation](#)  
*Computes the muscle unit force after the nonlinear saturation.*

## 8.7 NeuralTract.py File Reference

### Classes

- class [NeuralTract.NeuralTract](#)  
*classdocs*

### Namespaces

- [NeuralTract](#)

## 8.8 NeuralTractUnit.py File Reference

### Classes

- class [NeuralTractUnit.NeuralTractUnit](#)  
*classdocs*

### Namespaces

- [NeuralTractUnit](#)

## 8.9 PointProcessGenerator.py File Reference

### Classes

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

### Namespaces

- [PointProcessGenerator](#)

## Functions

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

## 8.10 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](#)  
*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](#)  
*Time course of the state during the pulse for the activation states and before and after the pulse for the inactivation states.*

## 8.11 simulation.py File Reference

### Namespaces

- [simulation](#)

### Functions

- def [simulation.simulador](#)  
*Main example function.*

## 8.12 Synapse.py File Reference

### Classes

- class [Synapse.Synapse](#)  
*classdocs*

### Namespaces

- [Synapse](#)

### Functions

- def [Synapse.compSynapCond](#)  
*Computes the synaptic conductance.*
- def [Synapse.compRon](#)
- def [Synapse.compRoff](#)
- def [Synapse.compRiStart](#)

- def [Synapse.compRiStop](#)
- def [Synapse.compRonStart](#)
- def [Synapse.compRoffStart](#)
- def [Synapse.compRonStop](#)
- def [Synapse.compRoffStop](#)

## 8.13 SynapsesFactory.py File Reference

### Classes

- class [SynapsesFactory.SynapsesFactory](#)  
*classdocs*

### Namespaces

- [SynapsesFactory](#)

# Index

- `__init__`
    - `AxonDelay::AxonDelay`, [20](#)
    - `ChannelConductance::ChannelConductance`, [23](#)
    - `Compartment::Compartment`, [25](#)
    - `Configuration::Configuration`, [27](#)
    - `MotorUnit::MotorUnit`, [31](#)
    - `MotorUnitPool::MotorUnitPool`, [37](#)
    - `NeuralTract::NeuralTract`, [40](#)
    - `NeuralTractUnit::NeuralTractUnit`, [42](#)
    - `PointProcessGenerator::PointProcessGenerator`, [44](#)
    - `PulseConductanceState::PulseConductanceState`, [45](#)
    - `Synapse::Synapse`, [48](#)
    - `SynapsesFactory::SynapsesFactory`, [55](#)
- `ActMatrix`
  - `MotorUnitPool::MotorUnitPool`, [38](#)
- `actType`
  - `PulseConductanceState::PulseConductanceState`, [46](#)
- `activation_Sat`
  - `MotorUnitPool::MotorUnitPool`, [38](#)
- `activation_nonSat`
  - `MotorUnitPool::MotorUnitPool`, [38](#)
- `activationModel`
  - `MotorUnitPool::MotorUnitPool`, [38](#)
- `addConductance`
  - `Synapse::Synapse`, [48](#)
- `addSomaSpike`
  - `MotorUnit::MotorUnit`, [31](#)
- `addSpinalSpike`
  - `AxonDelay::AxonDelay`, [20](#)
- `addTerminalSpike`
  - `AxonDelay::AxonDelay`, [20](#)
- `alpha_ms1`
  - `PulseConductanceState::PulseConductanceState`, [46](#)
  - `Synapse::Synapse`, [51](#)
- `an`
  - `MotorUnitPool::MotorUnitPool`, [38](#)
- `area_cm2`
  - `Compartment::Compartment`, [25](#)
- `atualizeActivationSignal`
  - `MotorUnitPool::MotorUnitPool`, [37](#)
- `atualizeCompartments`
  - `MotorUnit::MotorUnit`, [31](#)
- `atualizeDelay`
  - `MotorUnit::MotorUnit`, [32](#)
- `atualizeForce`
  - `MotorUnitPool::MotorUnitPool`, [38](#)
- `atualizeForceNoHill`
  - `MotorUnitPool::MotorUnitPool`, [37](#)
- `atualizeGenerator`
  - `PointProcessGenerator::PointProcessGenerator`, [44](#)
- `atualizeMotorUnit`
  - `MotorUnit::MotorUnit`, [32](#)
- `atualizeMotorUnitPool`
  - `MotorUnitPool::MotorUnitPool`, [37](#)
- `atualizeNeuralTractUnit`
  - `NeuralTractUnit::NeuralTractUnit`, [42](#)
- `atualizePool`
  - `NeuralTract::NeuralTract`, [40](#)
- `AxonDelay`, [11](#)
- `AxonDelay.AxonDelay`, [19](#)
- `AxonDelay.py`, [57](#)
- `AxonDelay::AxonDelay`
  - `__init__`, [20](#)
  - `addSpinalSpike`, [20](#)
  - `addTerminalSpike`, [20](#)
  - `index`, [21](#)
  - `latencySpinalTerminal_ms`, [21](#)
  - `latencyStimulusSpinal_ms`, [21](#)
  - `latencyStimulusTerminal_ms`, [21](#)
  - `length_m`, [21](#)
  - `stimulusPositiontoTerminal`, [21](#)
  - `terminalSpikeTrain`, [21](#)
  - `velocity_m_s`, [22](#)
- `bSat`
  - `MotorUnit::MotorUnit`, [33](#)
  - `MotorUnitPool::MotorUnitPool`, [38](#)
- `beta_ms1`
  - `PulseConductanceState::PulseConductanceState`, [46](#)
  - `Synapse::Synapse`, [51](#)
- `calcGCoupling`
  - `MotorUnit`, [12](#)
- `calcGLEak`
  - `Compartment`, [11](#)
- `capacitance_nF`
  - `Compartment::Compartment`, [25](#)
- `capacitanceInv`
  - `MotorUnit::MotorUnit`, [33](#)
- `changeState`
  - `PulseConductanceState::PulseConductanceState`, [45](#)
- `ChannelConductance`, [11](#)

ChannelConductance.ChannelConductance, [22](#)  
 ChannelConductance.py, [57](#)  
 ChannelConductance::ChannelConductance  
     \_\_init\_\_, [23](#)  
     compCond, [24](#)  
     compCondKf, [23](#)  
     compCondKs, [23](#)  
     compCondNa, [23](#)  
     computeCurrent, [23](#)  
     condState, [24](#)  
     EqPot\_mV, [24](#)  
     gmax\_muS, [24](#)  
     kind, [24](#)  
     lenStates, [24](#)  
     stateType, [24](#)  
 Channels  
     Compartment::Compartment, [25](#)  
 compCond  
     ChannelConductance::ChannelConductance, [24](#)  
 compCondKf  
     ChannelConductance::ChannelConductance, [23](#)  
 compCondKs  
     ChannelConductance::ChannelConductance, [23](#)  
 compCondNa  
     ChannelConductance::ChannelConductance, [23](#)  
 compGCouplingMatrix  
     MotorUnit, [12](#)  
 compNumber  
     MotorUnit::MotorUnit, [34](#)  
 compRiStart  
     Synapse, [16](#)  
 compRiStop  
     Synapse, [16](#)  
 compRoff  
     Synapse, [16](#)  
 compRoffStart  
     Synapse, [17](#)  
 compRoffStop  
     Synapse, [17](#)  
 compRon  
     Synapse, [17](#)  
 compRonStart  
     Synapse, [17](#)  
 compRonStop  
     Synapse, [18](#)  
 compSynapCond  
     Synapse, [18](#)  
 compValOff  
     PulseConductanceState, [15](#)  
 compValOn  
     PulseConductanceState, [15](#)  
 Compartment, [11](#)  
     calcGLEak, [11](#)  
 compartment  
     MotorUnit::MotorUnit, [34](#)  
 Compartment.Compartment, [24](#)  
 Compartment.py, [57](#)  
 Compartment::Compartment  
     \_\_init\_\_, [25](#)  
     area\_cm2, [25](#)  
     capacitance\_nF, [25](#)  
     Channels, [25](#)  
     computeCurrent, [25](#)  
     diameter\_mum, [25](#)  
     gLeak, [26](#)  
     index, [26](#)  
     kind, [26](#)  
     length\_mum, [26](#)  
     neuronKind, [26](#)  
     numberChannels, [26](#)  
     numberOfMultiSynapses, [26](#)  
     specifRes\_Ohmcm2, [26](#)  
     SynapsesIn, [26](#)  
     SynapsesOut, [26](#)  
 computeConductance  
     Synapse::Synapse, [48](#)  
 computeCurrent  
     ChannelConductance::ChannelConductance, [23](#)  
     Compartment::Compartment, [25](#)  
     Synapse::Synapse, [49](#), [51](#)  
 computeCurrent2  
     Synapse::Synapse, [49](#)  
 computeForce  
     MotorUnitPool, [14](#)  
 computeStateValue  
     PulseConductanceState::PulseConductanceState,  
         [45](#)  
 computeValueOff  
     PulseConductanceState::PulseConductanceState,  
         [46](#)  
 computeValueOn  
     PulseConductanceState::PulseConductanceState,  
         [46](#)  
 condState  
     ChannelConductance::ChannelConductance, [24](#)  
 conductanceState  
     Synapse::Synapse, [52](#)  
 conf  
     MotorUnit::MotorUnit, [34](#)  
     MotorUnitPool::MotorUnitPool, [38](#)  
 confArray  
     Configuration::Configuration, [28](#)  
 Configuration, [12](#)  
 Configuration.Configuration, [27](#)  
 Configuration.py, [58](#)  
 Configuration::Configuration  
     \_\_init\_\_, [27](#)  
     confArray, [28](#)  
     determineSynapses, [28](#)  
     inputFunctionGet, [28](#)  
     parameterSet, [28](#)  
     simDuration\_ms, [28](#)  
     timeStep\_ms, [29](#)  
     timeStepBySix\_ms, [29](#)  
     timeStepByTwo\_ms, [29](#)  
 dVdt

- MotorUnit::MotorUnit, 33
- Delay
  - MotorUnit::MotorUnit, 34
- delay\_ms
  - Synapse::Synapse, 52
- determineSynapses
  - Configuration::Configuration, 28
- diameter\_mum
  - Compartment::Compartment, 25
- diracDeltaValue
  - MotorUnitPool::MotorUnitPool, 38
- dynamics
  - Synapse::Synapse, 52
- EqPot\_mV
  - ChannelConductance::ChannelConductance, 24
  - Synapse::Synapse, 52
- expFinish
  - Synapse::Synapse, 52
- FR
  - NeuralTract::NeuralTract, 40
- force
  - MotorUnitPool::MotorUnitPool, 39
- G
  - MotorUnit::MotorUnit, 34
- gLeak
  - Compartment::Compartment, 26
- gMaxTot\_muS
  - Synapse::Synapse, 52
- GammaOrder
  - NeuralTractUnit::NeuralTractUnit, 42
  - PointProcessGenerator::PointProcessGenerator, 44
- GammaOrderInv
  - PointProcessGenerator::PointProcessGenerator, 44
- gammaPoint
  - PointProcessGenerator, 15
- gmax\_muS
  - ChannelConductance::ChannelConductance, 24
  - Synapse::Synapse, 52
- hillModel
  - MotorUnitPool::MotorUnitPool, 39
- iInjected
  - MotorUnit::MotorUnit, 34
- ilonic
  - MotorUnit::MotorUnit, 34
- index
  - AxonDelay::AxonDelay, 21
  - Compartment::Compartment, 26
  - MotorUnit::MotorUnit, 34
  - PointProcessGenerator::PointProcessGenerator, 44
- indicesOfSynapsesOnTarget
  - NeuralTractUnit::NeuralTractUnit, 42
- inputFunctionGet
  - Configuration::Configuration, 28
- kind
  - ChannelConductance::ChannelConductance, 24
  - Compartment::Compartment, 26
  - MotorUnit::MotorUnit, 34
  - MotorUnitPool::MotorUnitPool, 39
  - NeuralTract::NeuralTract, 40
  - PulseConductanceState::PulseConductanceState, 46
  - Synapse::Synapse, 52
- latencySpinalTerminal\_ms
  - AxonDelay::AxonDelay, 21
- latencyStimulusSpinal\_ms
  - AxonDelay::AxonDelay, 21
- latencyStimulusTerminal\_ms
  - AxonDelay::AxonDelay, 21
- lenStates
  - ChannelConductance::ChannelConductance, 24
- length\_m
  - AxonDelay::AxonDelay, 21
- length\_mum
  - Compartment::Compartment, 26
- listSpikes
  - MotorUnitPool::MotorUnitPool, 38
  - NeuralTract::NeuralTract, 40
- MNRefPer\_ms
  - MotorUnit::MotorUnit, 34
- MUnumber
  - MotorUnitPool::MotorUnitPool, 39
- MotorUnit, 12
  - calcGCoupling, 12
  - compGCouplingMatrix, 12
  - runge\_kutta, 13
- MotorUnit.MotorUnit, 29
- MotorUnit.py, 58
- MotorUnit::MotorUnit
  - \_\_init\_\_, 31
  - addSomaSpike, 31
  - atualizeCompartments, 31
  - atualizeDelay, 32
  - atualizeMotorUnit, 32
  - bSat, 33
  - capacitanceInv, 33
  - compNumber, 34
  - compartment, 33
  - conf, 34
  - dVdt, 33
  - Delay, 34
  - G, 34
  - iInjected, 34
  - ilonic, 34
  - index, 34
  - kind, 34
  - MNRefPer\_ms, 34
  - nerve, 35



- somaIndex, 35
- somaSpikeTrain, 35
- tSomaSpike, 35
- terminalSpikeTrain, 35
- threshold\_mV, 35
- twTet, 35
- TwitchAmp\_N, 35
- TwitchTc\_ms, 35
- v\_mV, 36
- MotorUnitPool, 13
  - computeForce, 14
  - twitchSaturation, 14
- MotorUnitPool.MotorUnitPool, 36
- MotorUnitPool.py, 58
- MotorUnitPool::MotorUnitPool
  - \_\_init\_\_, 37
  - ActMatrix, 38
  - activation\_Sat, 38
  - activation\_nonSat, 38
  - activationModel, 38
  - an, 38
  - atualizeActivationSignal, 37
  - atualizeForce, 38
  - atualizeForceNoHill, 37
  - atualizeMotorUnitPool, 37
  - bSat, 38
  - conf, 38
  - diracDeltaValue, 38
  - force, 39
  - hillModel, 39
  - kind, 39
  - listSpikes, 38
  - MUnumber, 39
  - pool, 39
  - poolSomaSpikes, 39
  - poolTerminalSpikes, 39
  - timeIndex, 39
  - twTet, 39
  - twitchAmp\_N, 39
  - unit, 39
- nerve
  - MotorUnit::MotorUnit, 35
- NeuralTract, 14
- NeuralTract.NeuralTract, 40
- NeuralTract.py, 59
- NeuralTract::NeuralTract
  - \_\_init\_\_, 40
  - atualizePool, 40
  - FR, 40
  - kind, 40
  - listSpikes, 40
  - Number, 41
  - pool, 41
  - poolTerminalSpikes, 41
  - target, 41
  - timeIndex, 41
  - unit, 41
- NeuralTractUnit, 14
- NeuralTractUnit.NeuralTractUnit, 41
- NeuralTractUnit.py, 59
- NeuralTractUnit::NeuralTractUnit
  - \_\_init\_\_, 42
  - atualizeNeuralTractUnit, 42
  - GammaOrder, 42
  - indicesOfSynapsesOnTarget, 42
  - spikesGenerator, 43
  - SynapsesOut, 43
  - terminalSpikeTrain, 43
  - transmitSpikes, 42
  - transmitSpikesThroughSynapses, 43
- neuronKind
  - Compartment::Compartment, 26
  - Synapse::Synapse, 52
- Non
  - Synapse::Synapse, 52
- Number
  - NeuralTract::NeuralTract, 41
- numberChannels
  - Compartment::Compartment, 26
- numberOfIncomingSynapses
  - Synapse::Synapse, 52
- numberOfSynapses
  - SynapsesFactory::SynapsesFactory, 55
- numberOfMultiSynapses
  - Compartment::Compartment, 26
- parameterSet
  - Configuration::Configuration, 28
- PointProcessGenerator, 15
  - gammaPoint, 15
- PointProcessGenerator.PointProcessGenerator, 43
- PointProcessGenerator.py, 59
- PointProcessGenerator::PointProcessGenerator
  - \_\_init\_\_, 44
  - atualizeGenerator, 44
  - GammaOrder, 44
  - GammaOrderInv, 44
  - index, 44
  - points, 44
  - threshold, 44
  - y, 44
- points
  - PointProcessGenerator::PointProcessGenerator, 44
- pool
  - MotorUnitPool::MotorUnitPool, 39
  - NeuralTract::NeuralTract, 41
  - Synapse::Synapse, 52
- poolSomaSpikes
  - MotorUnitPool::MotorUnitPool, 39
- poolTerminalSpikes
  - MotorUnitPool::MotorUnitPool, 39
  - NeuralTract::NeuralTract, 41
- PulseConductanceState, 15
  - compValOff, 15
  - compValOn, 15
- PulseConductanceState.PulseConductanceState, 44

PulseConductanceState.py, 59  
 PulseConductanceState::PulseConductanceState  
     \_\_init\_\_, 45  
     actType, 46  
     alpha\_ms1, 46  
     beta\_ms1, 46  
     changeState, 45  
     computeStateValue, 45  
     computeValueOff, 46  
     computeValueOn, 46  
     kind, 46  
     PulseDur\_ms, 46  
     state, 46  
     t0, 46  
     v0, 47  
     value, 47  
 PulseDur\_ms  
     PulseConductanceState::PulseConductanceState, 46  
  
 rInf  
     Synapse::Synapse, 53  
 receiveSpike  
     Synapse::Synapse, 50  
 ri  
     Synapse::Synapse, 53  
 Roff  
     Synapse::Synapse, 53  
 roff  
     Synapse::Synapse, 53  
 Ron  
     Synapse::Synapse, 53  
 ron  
     Synapse::Synapse, 53  
 runge\_kutta  
     MotorUnit, 13  
  
 simDuration\_ms  
     Configuration::Configuration, 28  
 simulador  
     simulation, 15  
 simulation, 15  
     simulador, 15  
 simulation.py, 60  
 somaIndex  
     MotorUnit::MotorUnit, 35  
 somaSpikeTrain  
     MotorUnit::MotorUnit, 35  
 specifRes\_Ohmcm2  
     Compartment::Compartment, 26  
 spikesGenerator  
     NeuralTractUnit::NeuralTractUnit, 43  
 spikesReceived  
     Synapse::Synapse, 53  
 startConductanceDynamics  
     Synapse::Synapse, 50  
 startConductanceNone  
     Synapse::Synapse, 50  
 startDynamicFunction  
     Synapse::Synapse, 53  
  
 startEntrance  
     Synapse::Synapse, 53  
 state  
     PulseConductanceState::PulseConductanceState, 46  
 stateType  
     ChannelConductance::ChannelConductance, 24  
 stimulusPositiontoTerminal  
     AxonDelay::AxonDelay, 21  
 stopConductanceDynamics  
     Synapse::Synapse, 51  
 stopConductanceNone  
     Synapse::Synapse, 51  
 stopDynamicFunction  
     Synapse::Synapse, 53  
 stopEntrance  
     Synapse::Synapse, 53  
 synContrib  
     Synapse::Synapse, 53  
 Synapse, 16  
     compRiStart, 16  
     compRiStop, 16  
     compRoff, 16  
     compRoffStart, 17  
     compRoffStop, 17  
     compRon, 17  
     compRonStart, 17  
     compRonStop, 18  
     compSynapCond, 18  
 Synapse.py, 60  
 Synapse.Synapse, 47  
 Synapse::Synapse  
     \_\_init\_\_, 48  
     addConductance, 48  
     alpha\_ms1, 51  
     beta\_ms1, 51  
     computeConductance, 48  
     computeCurrent, 49, 51  
     computeCurrent2, 49  
     conductanceState, 52  
     delay\_ms, 52  
     dynamics, 52  
     EqPot\_mV, 52  
     expFinish, 52  
     gMaxTot\_muS, 52  
     gmax\_muS, 52  
     kind, 52  
     neuronKind, 52  
     Non, 52  
     numberOfIncomingSynapses, 52  
     pool, 52  
     rInf, 53  
     receiveSpike, 50  
     ri, 53  
     Roff, 53  
     roff, 53  
     Ron, 53

- ron, 53
- spikesReceived, 53
- startConductanceDynamics, 50
- startConductanceNone, 50
- startDynamicFunction, 53
- startEntrance, 53
- stopConductanceDynamics, 51
- stopConductanceNone, 51
- stopDynamicFunction, 53
- stopEntrance, 53
- synContrib, 53
- t0, 54
- tBeginOfPulse, 54
- tEndOfPulse, 54
- tPeak\_ms, 54
- tauOff, 54
- tauOn, 54
- ti, 54
- Tmax\_mM, 54
- SynapsesFactory, 18
- SynapsesFactory.py, 60
- SynapsesFactory.SynapsesFactory, 54
- SynapsesFactory::SynapsesFactory
  - \_\_init\_\_, 55
  - numberOfSynapses, 55
- SynapsesIn
  - Compartment::Compartment, 26
- SynapsesOut
  - Compartment::Compartment, 26
  - NeuralTractUnit::NeuralTractUnit, 43
- t0
  - PulseConductanceState::PulseConductanceState, 46
  - Synapse::Synapse, 54
- tBeginOfPulse
  - Synapse::Synapse, 54
- tEndOfPulse
  - Synapse::Synapse, 54
- tPeak\_ms
  - Synapse::Synapse, 54
- tSomaSpike
  - MotorUnit::MotorUnit, 35
- target
  - NeuralTract::NeuralTract, 41
- tauOff
  - Synapse::Synapse, 54
- tauOn
  - Synapse::Synapse, 54
- terminalSpikeTrain
  - AxonDelay::AxonDelay, 21
  - MotorUnit::MotorUnit, 35
  - NeuralTractUnit::NeuralTractUnit, 43
- threshold
  - PointProcessGenerator::PointProcessGenerator, 44
- threshold\_mV
  - MotorUnit::MotorUnit, 35
- ti
  - Synapse::Synapse, 54
- timeIndex
  - MotorUnitPool::MotorUnitPool, 39
  - NeuralTract::NeuralTract, 41
- timeStep\_ms
  - Configuration::Configuration, 29
- timeStepBySix\_ms
  - Configuration::Configuration, 29
- timeStepByTwo\_ms
  - Configuration::Configuration, 29
- Tmax\_mM
  - Synapse::Synapse, 54
- transmitSpikes
  - NeuralTractUnit::NeuralTractUnit, 42
- transmitSpikesThroughSynapses
  - NeuralTractUnit::NeuralTractUnit, 43
- twTet
  - MotorUnit::MotorUnit, 35
  - MotorUnitPool::MotorUnitPool, 39
- TwitchAmp\_N
  - MotorUnit::MotorUnit, 35
- twitchAmp\_N
  - MotorUnitPool::MotorUnitPool, 39
- twitchSaturation
  - MotorUnitPool, 14
- TwitchTc\_ms
  - MotorUnit::MotorUnit, 35
- unit
  - MotorUnitPool::MotorUnitPool, 39
  - NeuralTract::NeuralTract, 41
- v0
  - PulseConductanceState::PulseConductanceState, 47
- v\_mV
  - MotorUnit::MotorUnit, 36
- value
  - PulseConductanceState::PulseConductanceState, 47
- velocity\_m\_s
  - AxonDelay::AxonDelay, 22
- y
  - PointProcessGenerator::PointProcessGenerator, 44