PyReMoto

Generated by Doxygen 1.8.11

Contents

1 ReMoto in Python 1

2 Namespace Index 3

2.1 Packages . . . 3

3 Hierarchical Index 5

3.1 Class Hierarchy . . . 5

4 Class Index 7

4.1 Class List . . . 7

5 File Index 9

5.1 File List . . . 9

6 Namespace Documentation 11

6.1 AxonDelay Namespace Reference . . . 11

6.2 ChannelConductance Namespace Reference . . . 11

6.3 Compartment Namespace Reference . . . 11

6.3.1 Function Documentation . . . 12

6.3.1.1 calcGLeak(area, specificRes) . . . 12

6.4 Configuration Namespace Reference . . . 12

6.5 MotorUnit Namespace Reference . . . 12

6.5.1 Function Documentation . . . 13

6.5.1.1 calcGCoupling(cytR, lComp1, lComp2, dComp1, dComp2) . . . 13

6.5.1.2 compGCouplingMatrix(gc) . . . 13

6.5.1.3 runge_kutta(derivativeFunction, t, x, timeStep, timeStepByTwo, timeStepBySix) . 14

6.6 MotorUnitPool Namespace Reference 14
6.6.1 Function Documentation 15
6.6.1.1 twitchSaturation(activationsat, b) 15
6.7 NeuralTract Namespace Reference 15
6.8 NeuralTractUnit Namespace Reference 15
6.9 PointProcessGenerator Namespace Reference 16
6.9.1 Function Documentation 16
6.9.1.1 gammaPoint(GammaOrder, GammaOrderInv) 16
6.10 PulseConductanceState Namespace Reference 17
6.10.1 Function Documentation 17
6.10.1.1 compValOff(v0, alpha, beta, t, t0) 17
6.10.1.2 compValOn(v0, alpha, beta, t, t0) 17
6.11 simulation Namespace Reference 18
6.11.1 Function Documentation 18
6.11.1.1 simulador() 18
6.12 Synapse Namespace Reference 18
6.12.1 Function Documentation 19
6.12.1.1 compRiStart(ri, t, ti, tPeak, tauOff) 19
6.12.1.2 compRiStop(rInf, ri, expFinish) 19
6.12.1.3 compRoff(Roff, t0, t, tauOff) 20
6.12.1.4 compRoffStart(Roff, ri, synContrib) 20
6.12.1.5 compRoffStop(Roff, ri, synContrib) 21
6.12.1.6 compRon(Non, rInf, Ron, t0, t, tauOn) 21
6.12.1.7 compRonStart(Ron, ri, synContrib) 22
6.12.1.8 compRonStop(Ron, ri, synContrib) 22
6.12.1.9 compSynapCond(Gmax, Ron, Roff) 23
6.13 SynapsesFactory Namespace Reference 23

7 Class Documentation **25**

7.1 AxonDelay.AxonDelay Class Reference . . . 25
7.1.1 Detailed Description . . . 26
7.1.2 Constructor & Destructor Documentation . . . 26
7.1.2.1 __init__(self, conf, nerve, pool, index) . . . 26
7.1.3 Member Function Documentation . . . 26
7.1.3.1 addSpinalSpike(self, t) . . . 26
7.1.3.2 addTerminalSpike(self, t) . . . 27
7.1.4 Member Data Documentation . . . 27
7.1.4.1 index . . . 27
7.1.4.2 latencySpinalTerminal_ms . . . 27
7.1.4.3 latencyStimulusSpinal_ms . . . 27
7.1.4.4 latencyStimulusTerminal_ms . . . 27
7.1.4.5 length_m . . . 28
7.1.4.6 stimulusPositiontoTerminal . . . 28
7.1.4.7 terminalSpikeTrain . . . 28
7.1.4.8 velocity_m_s . . . 28
7.2 ChannelConductance.ChannelConductance Class Reference . . . 28
7.2.1 Detailed Description . . . 29
7.2.2 Constructor & Destructor Documentation . . . 29
7.2.2.1 __init__(self, kind, conf, compArea, pool, index) . . . 29
7.2.3 Member Function Documentation . . . 30
7.2.3.1 compCondKf(self, V_mV) . . . 30
7.2.3.2 compCondKs(self, V_mV) . . . 30
7.2.3.3 compCondNa(self, V_mV) . . . 31
7.2.3.4 computeCurrent(self, t, V_mV) . . . 31
7.2.4 Member Data Documentation . . . 31
7.2.4.1 compCond . . . 31
7.2.4.2 condState . . . 31
7.2.4.3 EqPot_mV . . . 32

7.2.4.4 gmax_muS 32
7.2.4.5 kind 32
7.2.4.6 lenStates 32
7.2.4.7 stateType 32
7.3 Compartment.Compartment Class Reference 32
7.3.1 Detailed Description 33
7.3.2 Constructor & Destructor Documentation 33
7.3.2.1 __init__(self, kind, conf, pool, index, neuronKind) 33
7.3.3 Member Function Documentation 34
7.3.3.1 computeCurrent(self, t, V_mV) 34
7.3.4 Member Data Documentation 34
7.3.4.1 capacitance_nF 34
7.3.4.2 Channels 34
7.3.4.3 diameter_mum 34
7.3.4.4 gLeak 34
7.3.4.5 index 34
7.3.4.6 kind 35
7.3.4.7 length_mum 35
7.3.4.8 neuronKind 35
7.3.4.9 numberChannels 35
7.3.4.10 SynapsesIn 35
7.3.4.11 SynapsesOut 35
7.4 Configuration.Configuration Class Reference 35
7.4.1 Detailed Description 36
7.4.2 Constructor & Destructor Documentation 36
7.4.2.1 __init__(self, filename) 36
7.4.3 Member Function Documentation 37
7.4.3.1 determineSynapses(self, neuralSource) 37
7.4.3.2 inputFunctionGet(self, function) 37
7.4.3.3 parameterSet(self, paramTag, pool, index) 37

7.4.4 Member Data Documentation 38
7.4.4.1 confArray 38
7.4.4.2 simDuration_ms 38
7.4.4.3 timeStep_ms 38
7.4.4.4 timeStepBySix_ms 38
7.4.4.5 timeStepByTwo_ms 38
7.5 MotorUnit.MotorUnit Class Reference 38
7.5.1 Detailed Description 40
7.5.2 Constructor & Destructor Documentation 40
7.5.2.1 __init__(self, conf, pool, index, kind) 40
7.5.3 Member Function Documentation 40
7.5.3.1 addSomaSpike(self, t) 40
7.5.3.2 atualizeCompartments(self, t) 41
7.5.3.3 atualizeDelay(self, t) 41
7.5.3.4 atualizeMotorUnit(self, t) 42
7.5.3.5 dVdt(self, t, V) 42
7.5.4 Member Data Documentation 43
7.5.4.1 bSat 43
7.5.4.2 capacitanceInv 43
7.5.4.3 compartment 43
7.5.4.4 compNumber 43
7.5.4.5 conf 43
7.5.4.6 Delay 43
7.5.4.7 G 43
7.5.4.8 iInjected 44
7.5.4.9 iIonic 44
7.5.4.10 index 44
7.5.4.11 kind 44
7.5.4.12 MNRefPer_ms 44
7.5.4.13 nerve 44

7.5.4.14 somaIndex 44
7.5.4.15 somaSpikeTrain 45
7.5.4.16 terminalSpikeTrain 45
7.5.4.17 threshold_mV 45
7.5.4.18 tSomaSpike 45
7.5.4.19 TwitchAmp_N 45
7.5.4.20 TwitchTc_ms 45
7.5.4.21 twTet 45
7.5.4.22 v_mV 46
7.6 MotorUnitPool.MotorUnitPool Class Reference 46
7.6.1 Detailed Description 47
7.6.2 Constructor & Destructor Documentation 47
7.6.2.1 __init__(self, conf, pool) 47
7.6.3 Member Function Documentation 48
7.6.3.1 atualizeActivationSignal(self, t) 48
7.6.3.2 atualizeForceNoHill(self) 48
7.6.3.3 atualizeMotorUnitPool(self, t) 49
7.6.3.4 listSpikes(self) 49
7.6.4 Member Data Documentation 49
7.6.4.1 activation_nonSat 49
7.6.4.2 activation_Sat 49
7.6.4.3 activationModel 49
7.6.4.4 ActMatrix 50
7.6.4.5 an 50
7.6.4.6 atualizeForce 50
7.6.4.7 bSat 50
7.6.4.8 conf 50
7.6.4.9 diracDeltaValue 51
7.6.4.10 force 51
7.6.4.11 hillModel 51

7.6.4.12 kind 51

7.6.4.13 MUnumber 51

7.6.4.14 pool 51

7.6.4.15 poolSomaSpikes 51

7.6.4.16 poolTerminalSpikes 52

7.6.4.17 timeIndex 52

7.6.4.18 twitchAmp_N 52

7.6.4.19 twTet 52

7.6.4.20 unit 52

7.7 NeuralTract.NeuralTract Class Reference 52

7.7.1 Detailed Description 53

7.7.2 Constructor & Destructor Documentation 53

7.7.2.1 __init__(self, conf, pool) 53

7.7.3 Member Function Documentation 53

7.7.3.1 atualizePool(self, t) 53

7.7.3.2 listSpikes(self) 53

7.7.4 Member Data Documentation 53

7.7.4.1 FR 53

7.7.4.2 kind 54

7.7.4.3 Number 54

7.7.4.4 pool 54

7.7.4.5 poolTerminalSpikes 54

7.7.4.6 target 54

7.7.4.7 timeIndex 54

7.7.4.8 unit 54

7.8 NeuralTractUnit.NeuralTractUnit Class Reference 54

7.8.1 Detailed Description 55

7.8.2 Constructor & Destructor Documentation 55

7.8.2.1 __init__(self, conf, pool, index) 55

7.8.3 Member Function Documentation 55

7.8.3.1 atualizeNeuralTractUnit(self, t, FR) 55
7.8.3.2 transmitSpikes(self, t) 56
7.8.4 Member Data Documentation 56
7.8.4.1 GammaOrder 56
7.8.4.2 indicesOfSynapsesOnTarget 56
7.8.4.3 spikesGenerator 56
7.8.4.4 SynapsesOut 56
7.8.4.5 terminalSpikeTrain 56
7.8.4.6 transmitSpikesThroughSynapses 56
7.9 object Class Reference 57
7.10 PointProcessGenerator.PointProcessGenerator Class Reference 57
7.10.1 Detailed Description 57
7.10.2 Constructor & Destructor Documentation 57
7.10.2.1 __init__(self, GammaOrder, index) 57
7.10.3 Member Function Documentation 58
7.10.3.1 atualizeGenerator(self, t, firingRate) 58
7.10.4 Member Data Documentation 58
7.10.4.1 GammaOrder 58
7.10.4.2 GammaOrderInv 58
7.10.4.3 index 58
7.10.4.4 points 58
7.10.4.5 threshold 59
7.10.4.6 y 59
7.11 PulseConductanceState.PulseConductanceState Class Reference 59
7.11.1 Detailed Description 60
7.11.2 Constructor & Destructor Documentation 60
7.11.2.1 __init__(self, kind, conf, pool, index) 60
7.11.3 Member Function Documentation 60
7.11.3.1 changeState(self, t) 60
7.11.3.2 computeStateValue(self, t) 61

7.11.4 Member Data Documentation 61
7.11.4.1 actType 61
7.11.4.2 alpha_ms1 61
7.11.4.3 beta_ms1 61
7.11.4.4 computeValueOff 61
7.11.4.5 computeValueOn 61
7.11.4.6 kind 61
7.11.4.7 PulseDur_ms 62
7.11.4.8 state 62
7.11.4.9 t0 62
7.11.4.10 v0 62
7.11.4.11 value 62
7.12 Synapse.Synapse Class Reference 62
7.12.1 Detailed Description 63
7.12.2 Constructor & Destructor Documentation 64
7.12.2.1 __init__(self, conf, pool, index, compartment, kind, neuronKind) 64
7.12.3 Member Data Documentation 64
7.12.3.1 alpha_ms1 64
7.12.3.2 beta_ms1 64
7.12.3.3 conductanceState 64
7.12.3.4 delay_ms 64
7.12.3.5 dynamics 64
7.12.3.6 EqPot_mV 64
7.12.3.7 expFinish 65
7.12.3.8 gmax_muS 65
7.12.3.9 gMaxTot_muS 65
7.12.3.10 kind 65
7.12.3.11 neuronKind 65
7.12.3.12 Non 65
7.12.3.13 numberOfIncomingSynapses 65

7.12.3.14 pool 65
7.12.3.15 ri 66
7.12.3.16 rInf 66
7.12.3.17 Roff 66
7.12.3.18 Ron 66
7.12.3.19 t0 66
7.12.3.20 tauOff 66
7.12.3.21 tauOn 66
7.12.3.22 tBeginOfPulse 67
7.12.3.23 tEndOfPulse 67
7.12.3.24 ti 67
7.12.3.25 Tmax_mM 67
7.12.3.26 tPeak_ms 67
7.13 SynapsesFactory.SynapsesFactory Class Reference 67
7.13.1 Detailed Description 68
7.13.2 Constructor & Destructor Documentation 68
7.13.2.1 __init__(self, conf, pools) 68
7.13.3 Member Data Documentation 68
7.13.3.1 numberOfSynapses 68

8 File Documentation 69
8.1 AxonDelay.py File Reference 69
8.2 ChannelConductance.py File Reference 69
8.3 Compartment.py File Reference 69
8.4 Configuration.py File Reference 70
8.5 MotorUnit.py File Reference 70
8.6 MotorUnitPool.py File Reference 70
8.7 NeuralTract.py File Reference 71
8.8 NeuralTractUnit.py File Reference 71
8.9 PointProcessGenerator.py File Reference 71
8.10 PulseConductanceState.py File Reference 72
8.11 simulation.py File Reference 72
8.12 Synapse.py File Reference 72
8.13 SynapsesFactory.py File Reference 73

Index 75

Chapter 1

ReMoto in Python

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

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

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

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

Chapter 2

Namespace Index

2.1 Packages

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

AxonDelay 11
ChannelConductance 11
Compartment 11
Configuration 12
MotorUnit 12
MotorUnitPool 14
NeuralTract 15
NeuralTractUnit 15
PointProcessGenerator 16
PulseConductanceState 17
simulation 18
Synapse 18
SynapsesFactory 23

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

object 57
- AxonDelay.AxonDelay 25
- ChannelConductance.ChannelConductance 28
- Compartment.Compartment 32
- Configuration.Configuration 35
- MotorUnit.MotorUnit 38
- MotorUnitPool.MotorUnitPool 46
- NeuralTract.NeuralTract 52
- NeuralTractUnit.NeuralTractUnit 54
- PointProcessGenerator.PointProcessGenerator 57
- PulseConductanceState.PulseConductanceState 59
- Synapse.Synapse 62
- SynapsesFactory.SynapsesFactory 67

Chapter 4

Class Index

4.1 Class List

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

AxonDelay.AxonDelay
Class that implements a delay correspondent to the nerve 25
ChannelConductance.ChannelConductance
Class that implements a model of the ionic Channels in a compartment 28
Compartment.Compartment
Class that implements a neural compartment 32
Configuration.Configuration
Class that builds an object of Configuration, based on a configuration file 35
MotorUnit.MotorUnit
Class that implements a motor unit model 38
MotorUnitPool.MotorUnitPool
Class that implements a motor unit pool 46
NeuralTract.NeuralTract
Classdocs 52
NeuralTractUnit.NeuralTractUnit
Classdocs 54
object 57
PointProcessGenerator.PointProcessGenerator
Generator of point processes 57
PulseConductanceState.PulseConductanceState
Implements the Destexhe pulse approximation of the solution of the states of the Hodgkin-Huxley neuron model 59
Synapse.Synapse
Implements the synapse model from Destexhe (1994) using the computational method from Lytton (1996) 62
SynapsesFactory.SynapsesFactory
Class to build all the synapses in the system 67

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

AxonDelay.py 69
ChannelConductance.py 69
Compartment.py 69
Configuration.py 70
MotorUnit.py 70
MotorUnitPool.py 70
NeuralTract.py 71
NeuralTractUnit.py 71
PointProcessGenerator.py 71
PulseConductanceState.py 72
simulation.py 72
Synapse.py 72
SynapsesFactory.py 73

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 (area, specificRes)

  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 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:

$$g = 10^6.\frac{A}{\rho} \tag{6.1}$$

where A is the compartment area [cm ^{2}], ρ 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 (cytR, lComp1, lComp2, dComp1, dComp2)

  Calculates the coupling conductance between two compartments.
- def compGCouplingMatrix (gc)

  Computes the Coupling Matrix to be used in the dVdt function of the N compartments of the motor unit.
- def runge_kutta (derivativeFunction, t, x, timeStep, timeStepByTwo, timeStepBySix)

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

6.5.1 Function Documentation

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

Calculates the coupling conductance between two compartments.

- Inputs:
  - **cytR**: Cytoplasmatic resistivity in Ω.cm.
  - **lComp1, lComp2**: length of the compartments in μm.
  - **dComp1, dComp2**: diameter of the compartments in μm.
- Output:
  - coupling conductance in MS.

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

$$g_c = \frac{2.10^2}{\frac{R_{cyt}l_1}{\pi r_1^2} + \frac{R_{cyt}l_2}{\pi r_2^2}} \tag{6.2}$$

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

Definition at line 46 of file MotorUnit.py.

6.5.1.2 def MotorUnit.compGCouplingMatrix (*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 calcGcoupling.

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

$$GC = \begin{bmatrix} -g_c[0] & g_c[0] & 0 & ... & ... & 0 & 0 & 0 \\ g_c[0] & -g_c[0]-g_c[1] & g_c[1] & 0 & ... & ... & 0 & 0 \\ \vdots & & \ddots & & ... & & 0 & 0 \\ 0 & ... & g_c[i-1] & -g_c[i-1]-g_c[i] & g_c[i] & 0 & ... & 0 \\ 0 & 0 & 0 & ... & ... & & & 0 \\ 0 & & ... & & g_c[N-2] & -g_c[N-2]-g_c[N-1] & g_c[N-1] & 0 \\ 0 & ... & 0 & & & 0 & g_c[N-1] & -g_c[N - \end{bmatrix} \tag{6.3}$$

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:

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

First, four derivatives are computed:

$$k_1 = f(t, x(t)) \tag{6.5}$$
$$k_2 = f(t + \frac{\Delta t}{2}, x(t) + \frac{\Delta t}{2}.k_1) \tag{6.6}$$
$$k_3 = f(t + \frac{\Delta t}{2}, x(t) + \frac{\Delta t}{2}.k_2) \tag{6.7}$$
$$k_4 = f(t + \Delta t, x(t) + \Delta t.k_3) \tag{6.8}$$

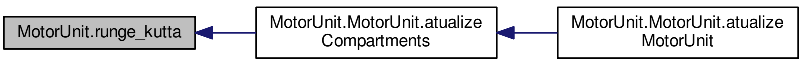
where Δt is the time step of the numerical solution of the differential equation.

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

$$x(t + \Delta t) = x(t) + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{6.9}$$

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 (activationsat, b)

    Computes the muscle unit force after the nonlinear saturation.

6.6.1 Function Documentation

6.6.1.1 def MotorUnitPool.twitchSaturation (*activationsat, b*)

Computes the muscle unit force after the nonlinear saturation.

$$a_{sat} = \frac{1 - e^{-b.a_{nSat}}}{1 + e^{-b.a_{nSat}}} \tag{6.10}$$

- Inputs:
    - **activationsat**: activation signal 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 (GammaOrder, GammaOrderInv)

  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.

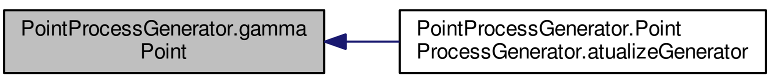
The number is generated according to:

$$\Gamma = -\frac{1}{\lambda}\ln(\prod_{i=1}^{\lambda} U(0,1)) \tag{6.11}$$

where λ 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 (v0, alpha, beta, t, t0)

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

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

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:

$$v(t) = 1 + (v_0 - 1)\exp[-\alpha(t - t_0)] \tag{6.12}$$

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:

$$v(t) = v_0 \exp[-\beta(t - t_0)] \tag{6.13}$$

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 ()

6.11.1 Function Documentation

6.11.1.1 def simulation.simulador ( )

Definition at line 21 of file simulation.py.

6.12 Synapse Namespace Reference

Classes

- class Synapse

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

Functions

- def compSynapCond (Gmax, Ron, Roff)

  Computes the synaptic conductance.
- def compRon (Non, rInf, Ron, t0, t, tauOn)

  Computes the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that have neurotransmitters being released (during the pulse).
- def compRoff (Roff, t0, t, tauOff)

  Computes the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that do not have neurotransmitters being released (before and after the pulse).
- def compRiStart (ri, t, ti, tPeak, tauOff)

  Computes the fraction of bound postsynaptic receptors to neurotransmitters in individual synapses when the neurotransmitter begin (begin of the pulse).
- def compRiStop (rInf, ri, expFinish)

  Computes the fraction of bound postsynaptic receptors to neurotransmitters in individual synapses when the neurotransmitter release stops (the pulse ends).
- def compRonStart (Ron, ri, synContrib)

  Incorporates a new conductance to the set of conductances during a pulse.
- def compRoffStart (Roff, ri, synContrib)

  Incorporates a new conductance to the set of conductances that are not during a pulse.
- def compRonStop (Ron, ri, synContrib)

  Removes a conductance from the set of conductances during a pulse.
- def compRoffStop (Roff, ri, synContrib)

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

6.12.1 Function Documentation

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

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

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

It is computed by the following equation:

$$r_{i_{newValue}} = r_{i_{oldValue}} \exp\left(\frac{t_i + T_{dur} - t}{\tau_{off}}\right) \tag{6.14}$$

Definition at line 142 of file Synapse.py.

6.12.1.2 def Synapse.compRiStop (*rInf, ri, expFinish*)

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

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

It is computed by the following equation:

$$r_{i_{newValue}} = r_\infty + (r_{i_{oldValue}} - r_\infty) \exp\left(\frac{T_{dur}}{\tau_{on}}\right) \tag{6.15}$$

Definition at line 173 of file Synapse.py.

6.12.1.3 def Synapse.compRoff (*Roff*, *t0*, *t*, *tauOff*)

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

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

It is computed by the following formula:

$$R_{off_{newValue}} = R_{off_{oldValue}} \exp\left(-\frac{t-t0}{\tau_{off}}\right) \tag{6.16}$$

Definition at line 112 of file Synapse.py.

6.12.1.4 def Synapse.compRoffStart (*Roff*, *ri*, *synContrib*)

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

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

It is computed as:

$$R_{off_{newValue}} = R_{off_{oldValue}} - r_i S_{indCont} \tag{6.17}$$

Definition at line 235 of file Synapse.py.

6.12.1.5 def Synapse.compRoffStop (*Roff*, *ri*, *synContrib*)

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

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

It is computed as:

$$R_{off_{newValue}} = R_{off_{oldValue}} + r_i S_{indCont} \tag{6.18}$$

Definition at line 297 of file Synapse.py.

6.12.1.6 def Synapse.compRon (*Non*, *rInf*, *Ron*, *t0*, *t*, *tauOn*)

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

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

It is computed by the following equation:

$$R_{on_{newValue}} = N_{on} r_\infty \left[1 - \exp\left(-\frac{t-t_0}{\tau_{on}}\right)\right] + R_{on_{oldValue}} \exp\left(-\frac{t-t_0}{\tau_{on}}\right) \tag{6.19}$$

Definition at line 78 of file Synapse.py.

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

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

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

It is computed as:

$$R_{on_{newValue}} = R_{on_{oldValue}} + r_i S_{indCont} \tag{6.20}$$

Definition at line 203 of file Synapse.py.

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

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

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

It is computed as:

$$R_{on_{newValue}} = R_{on_{oldValue}} - r_i S_{indCont} \tag{6.21}$$

Definition at line 265 of file Synapse.py.

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

Computes the synaptic conductance.

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

It is computed by the following formula:

$$G = G_{max}(R_{on} + R_{off}) \tag{6.22}$$

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

Definition at line 39 of file Synapse.py.

6.13 SynapsesFactory Namespace Reference

Classes

- class SynapsesFactory

  Class to build all the synapses in the system.

Chapter 7

Class Documentation

7.1 AxonDelay.AxonDelay Class Reference

Class that implements a delay correspondent to the nerve.

Public Member Functions

- def __init__ (self, conf, nerve, pool, index)

  Constructor.
- def addTerminalSpike (self, t)

  Indicates to the AxonDelay object that a spike has occurred in the Terminal.
- def addSpinalSpike (self, t)

  Indicates to the AxonDelay object that a spike has occurred in the soma.

Public Attributes

- index

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

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

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

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

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

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

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

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

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 35 of file AxonDelay.py.

7.1.3 Member Function Documentation

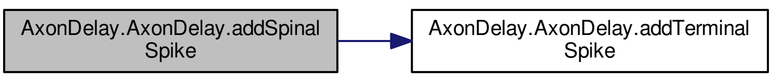
7.1.3.1 def AxonDelay.AxonDelay.addSpinalSpike (*self,* *t*)

Indicates to the AxonDelay object that a spike has occurred in the soma.

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

Definition at line 76 of file AxonDelay.py.

Here is the call graph for this function:



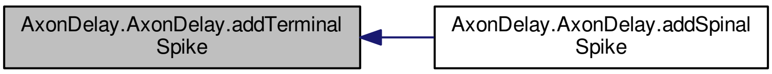
7.1.3.2 def AxonDelay.AxonDelay.addTerminalSpike (*self*, *t*)

Indicates to the AxonDelay object that a spike has occurred in the Terminal.

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

Definition at line 65 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 39 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 50 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 48 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 52 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 42 of file AxonDelay.py.

7.1.4.6 AxonDelay.AxonDelay.stimulusPositiontoTerminal

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

Definition at line 46 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 55 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 44 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__ (self, kind, conf, compArea, pool, index)

  Constructor.
- def computeCurrent (self, t, V_mV)

  Computes the current genrated by the ionic Channel.
- def compCondKf (self, V_mV)

  Computes the conductance of a Kf Channel.
- def compCondKs (self, V_mV)

  Computes the conductance of a slow potassium Channel.
- def compCondNa (self, V_mV)

  Computes the conductance of a Na Channel.

Public Attributes

- kind

  string with the type of the ionic channel.
- condState

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

  Equilibrium Potential of the ionic channel, mV.
- gmax_muS

  Maximal conductance, in μS, of the ionic channel.
- stateType

  String with type of dynamics of the states.
- compCond

  Function that computes the conductance dynamics.
- lenStates

  Integer with the number of states in the ionic channel.

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*)

Constructor.

Builds an ionic channel conductance.

-Inputs:

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

Definition at line 38 of file ChannelConductance.py.

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 μS.

It is computed as:

$$g = g_{max} n^4 (E_0 - V) \tag{7.1}$$

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

Definition at line 115 of file ChannelConductance.py.

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

Computes the conductance of a slow potassium Channel.

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

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

It is computed as:

$$g = g_{max} q^2 (E_0 - V) \tag{7.2}$$

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

Definition at line 138 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 μS.

It is computed as:

$$g = g_{max} m^3 h (E_0 - V) \tag{7.3}$$

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

Definition at line 159 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 91 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 60 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 44 of file ChannelConductance.py.

7.2.4.3 ChannelConductance.ChannelConductance.EqPot_mV

Equilibrium Potential of the ionic channel, mV.

Definition at line 47 of file ChannelConductance.py.

7.2.4.4 ChannelConductance.ChannelConductance.gmax_muS

Maximal conductance, in μS, of the ionic channel.

Definition at line 49 of file ChannelConductance.py.

7.2.4.5 ChannelConductance.ChannelConductance.kind

string with the type of the ionic channel.

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

Definition at line 42 of file ChannelConductance.py.

7.2.4.6 ChannelConductance.ChannelConductance.lenStates

Integer with the number of states in the ionic channel.

Definition at line 74 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 52 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__ (self, kind, conf, pool, index, neuronKind)

  Constructor.
- def computeCurrent (self, t, V_mV)

  Computes the active currents of the compartment.

Public Attributes

- Channels

    List of ChannelConductance objects in the Compartment.
- neuronKind

    String with the type of the motor unit.
- SynapsesOut

    List of summed synapses (see Lytton, 1996) that the Compartment do with other neural components.
- SynapsesIn

    List of summed synapses (see Lytton, 1996) that the Compartment receive from other neural components.
- kind

    The kind of compartment.
- index

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

    Length of the compartment, in μm.
- diameter_mum

    Diameter of the compartment, in μm.
- capacitance_nF

    Capacitance of the compartment, in nF.
- gLeak

    Leak conductance of the compartment, in MS.
- numberChannels

    Integer with the number of ionic channels.

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 μm.

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 μm.

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__ (self, filename)

    Constructor.
- def parameterSet (self, paramTag, pool, index)

    Function that returns the value of wished parameter specified in the paramTag variable.
- def inputFunctionGet (self, function)

    Returns a numpy array with the values of the function for the whole simulation.
- def determineSynapses (self, neuralSource)

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

Public Attributes

- confArray

    An array with all the simulation parameters.
- timeStep_ms

    Time step of the numerical solution of the differential equation.
- simDuration_ms

    Total length of the simulation in ms.
- timeStepByTwo_ms

    The variable timeStep divided by two, for 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 38 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 52 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 153 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:
  - narray with the function values for each instant.

Definition at line 137 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 rmto file.
  - **pool**: pool from which the unit that will receive the parameter value belongs. For example SOL. It is used only in the parameters that have a range.
  - **index**: index of the unit. It is is an integer.
- Outputs:
  - required parameter value

Definition at line 93 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 55 of file Configuration.py.

7.4.4.2 Configuration.Configuration.simDuration_ms

Total length of the simulation in ms.

Definition at line 65 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 62 of file Configuration.py.

7.4.4.4 Configuration.Configuration.timeStepBySix_ms

The variable timeStep divided by six, for computaional efficiency.

Definition at line 69 of file Configuration.py.

7.4.4.5 Configuration.Configuration.timeStepByTwo_ms

The variable timeStep divided by two, for computaional efficiency.

Definition at line 67 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__ (self, conf, pool, index, kind)

  Constructor.
- def atualizeMotorUnit (self, t)

  Atualize the dynamical and nondynamical (delay) parts of the motor unit.
- def atualizeCompartments (self, t)

  Atualize all neural compartments.
- def dVdt (self, t, V)

  Compute the potential derivative of all compartments of the motor unit.
- def addSomaSpike (self, t)

  When the soma potential is above the threshold a spike is added tom the soma.
- def atualizeDelay (self, t)

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

Public Attributes

- conf

  Configuration object with the simulation parameters.
- kind

  String with the type of the motor unit.
- tSomaSpike

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

  Vector with the instants of spikes at the soma.
- index

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

  Vector of Compartment of the Motor Unit.
- threshold_mV

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

  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

  Amplitude 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 148 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 167 of file MotorUnit.py.

7.5.3 Member Function Documentation

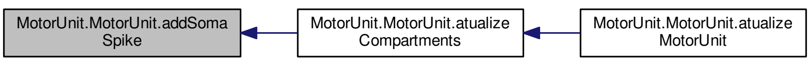
7.5.3.1 def MotorUnit.MotorUnit.addSomaSpike (*self, t*)

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

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

Definition at line 328 of file MotorUnit.py.

Here is the caller graph for this function:



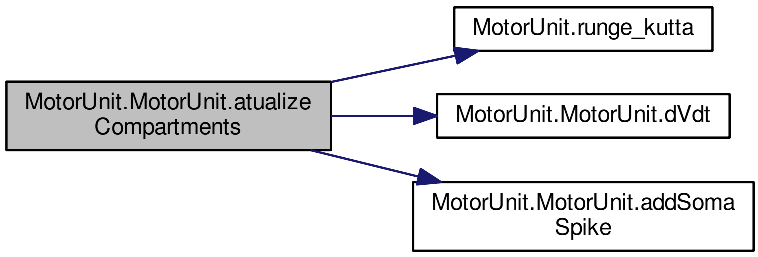
7.5.3.2 def MotorUnit.MotorUnit.atualizeCompartments (*self*, *t*)

Atualize all neural compartments.

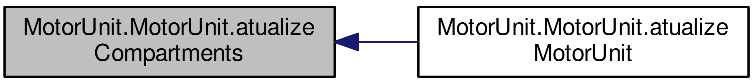
- Inputs:
  - **t**: current instant, in ms.

Definition at line 291 of file MotorUnit.py.

Here is the call graph for this function:



Here is the caller graph for this function:



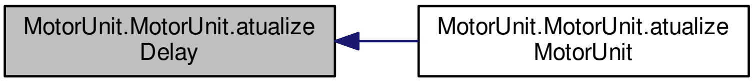
7.5.3.3 def MotorUnit.MotorUnit.atualizeDelay (*self*, *t*)

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

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

Definition at line 344 of file MotorUnit.py.

Here is the caller graph for this function:



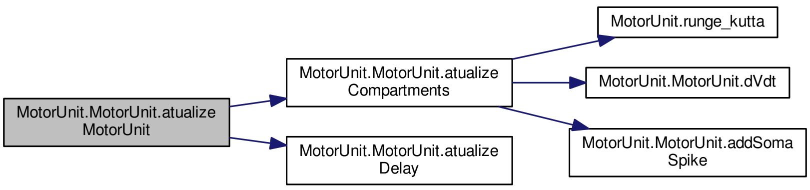
7.5.3.4 def MotorUnit.MotorUnit.atualizeMotorUnit (*self*, *t*)

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

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

Definition at line 279 of file MotorUnit.py.

Here is the call graph for this function:



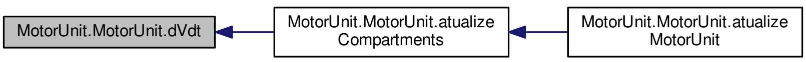
7.5.3.5 def MotorUnit.MotorUnit.dVdt (*self*, *t*, *V*)

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 314 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 264 of file MotorUnit.py.

7.5.4.2 MotorUnit.MotorUnit.capacitanceInv

Vector with the inverse of the capacitance of all compartments.

Definition at line 216 of file MotorUnit.py.

7.5.4.3 MotorUnit.MotorUnit.compartment

Vector of Compartment of the Motor Unit.

Definition at line 187 of file MotorUnit.py.

7.5.4.4 MotorUnit.MotorUnit.compNumber

Number of compartments.

Definition at line 194 of file MotorUnit.py.

7.5.4.5 MotorUnit.MotorUnit.conf

Configuration object with the simulation parameters.

Definition at line 170 of file MotorUnit.py.

7.5.4.6 MotorUnit.MotorUnit.Delay

AxonDelay object of the motor unit.

Definition at line 249 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 231 of file MotorUnit.py.

7.5.4.8 MotorUnit.MotorUnit.iInjected

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

Definition at line 222 of file MotorUnit.py.

7.5.4.9 MotorUnit.MotorUnit.iIonic

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 220 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 185 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 175 of file MotorUnit.py.

7.5.4.12 MotorUnit.MotorUnit.MNRefPer_ms

Refractory period, in ms, of the motoneuron.

Definition at line 238 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 244 of file MotorUnit.py.

7.5.4.14 MotorUnit.MotorUnit.somaIndex

index of the soma compartment.

Definition at line 235 of file MotorUnit.py.

7.5.4.15 MotorUnit.MotorUnit.somaSpikeTrain

Vector with the instants of spikes at the soma.

Definition at line 183 of file MotorUnit.py.

7.5.4.16 MotorUnit.MotorUnit.terminalSpikeTrain

Vector with the instants of spikes at the terminal.

Definition at line 253 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 189 of file MotorUnit.py.

7.5.4.18 MotorUnit.MotorUnit.tSomaSpike

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

Definition at line 180 of file MotorUnit.py.

7.5.4.19 MotorUnit.MotorUnit.TwitchAmp_N

Amplitude of the muscle unit twitch, in N.

Definition at line 262 of file MotorUnit.py.

7.5.4.20 MotorUnit.MotorUnit.TwitchTc_ms

Contraction time of the twitch muscle unit, in ms.

Definition at line 260 of file MotorUnit.py.

7.5.4.21 MotorUnit.MotorUnit.twTet

Twitch- tetanus relationship.

Definition at line 266 of file MotorUnit.py.

7.5.4.22 MotorUnit.MotorUnit.v_mV

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

Definition at line 196 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__ (self, conf, pool)

    Constructor.
- def atualizeMotorUnitPool (self, t)

    Update all parts of the Motor Unit pool.
- def atualizeActivationSignal (self, t)

    Update the activation signal of the motor units.
- def atualizeForceNoHill (self)

    Compute the muscle force when no muscle dynamics (Hill model) is used.
- def listSpikes (self)

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

Public Attributes

- kind

    Indicates that is Motor Unit pool.
- conf

    Configuration object with the simulation parameters.
- pool

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

    Number of motor units.
- unit

    List of MotorUnit objects.
- poolSomaSpikes

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

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

    Model of the activation signal.
- ActMatrix

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

$$Av(n) = \left[\begin{array}{cccccccccccc} a_1(n-1) & a_1(n-2) & e_1(n-1) & \ldots & a_i(n-i) & a_i(n-2) & e_i(n-1) & \ldots & a_{N_{MU}(n-1)} & a_{N_{MU}(n-2)} & e_{N_{MU}}(n-1) \end{array} \right]^T \quad (7.4)$$

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

- an

  Is a vector formed as:

$$Av(n) = \left[\begin{array}{cccccccccccc} a_1(n-1) & a_1(n-2) & e_1(n-1) & \ldots & a_i(n-i) & a_i(n-2) & e_i(n-1) & \ldots & a_{N_{MU}(n-1)} & a_{N_{MU}(n-2)} & e_{N_{MU}}(n-1) \end{array} \right]^T \quad (7.5)$$

  It is multiplied by the matriz actMatrix to obtain the activation signal (see actMatrix explanation)
- activation_nonSat

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

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

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

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

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

  Dirac's delta approximation amplitude value.
- force

  Muscle force along time, in N.
- hillModel

  String indicating whther a Hill model is used or not.
- 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 41 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 53 of file MotorUnitPool.py.

7.6.3 Member Function Documentation

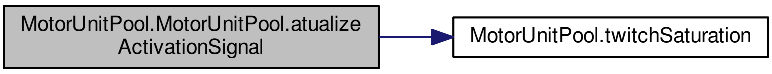
7.6.3.1 def MotorUnitPool.MotorUnitPool.atualizeActivationSignal (*self*, *t*)

Update the activation signal of the motor units.

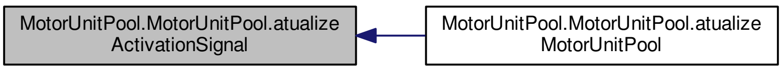
- Inputs:
  - **t**: current instant, in ms.

Definition at line 188 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.

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

Definition at line 220 of file MotorUnitPool.py.

7.6.3.3 def MotorUnitPool.MotorUnitPool.atualizeMotorUnitPool (*self*, *t*)

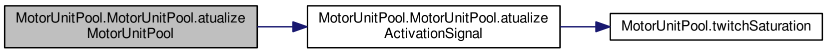
Update all parts of the Motor Unit pool.

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

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

Definition at line 173 of file MotorUnitPool.py.

Here is the call graph for this function:



7.6.3.4 def MotorUnitPool.MotorUnitPool.listSpikes (*self*)

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

Definition at line 228 of file MotorUnitPool.py.

7.6.4 Member Data Documentation

7.6.4.1 MotorUnitPool.MotorUnitPool.activation_nonSat

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

Definition at line 132 of file MotorUnitPool.py.

7.6.4.2 MotorUnitPool.MotorUnitPool.activation_Sat

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

Definition at line 146 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 87 of file MotorUnitPool.py.

7.6.4.4 MotorUnitPool.MotorUnitPool.ActMatrix

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

$$Av(n) = \left[\begin{array}{cccccccccccc} a_1(n-1) & a_1(n-2) & e_1(n-1) & \ldots & a_i(n-i) & a_i(n-2) & e_i(n-1) & \ldots & a_{N_{MU}(n-1)} & a_{N_{MU}(n-2)} & e_{N_{MU}}(n-1) \end{array} \right]^T \qquad (7.6)$$

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

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

$$A=\left[\begin{array}{ccccccccccc} 2\exp\left(-\frac{T}{T_{c_1}}\right) & -\exp\left(-2\frac{T}{T_{c_1}}\right) & \frac{T^2}{T_{c_1}}\exp\left(1-\frac{T}{T_{c_1}}\right) & 0 & \ldots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ldots & & & & & \ldots & 0 \\ 0 & \ldots & & 0 & 2\exp\left(-\frac{T}{T_{c_i}}\right) & -\exp\left(-2\frac{T}{T_{c_i}}\right) & \frac{T^2}{T_{c_i}}\exp\left(1-\frac{T}{T_{c_i}}\right) & 0 & & & 0 \\ 0 & 0 & \ldots & & & 0 & 0 & \ddots & 0 & 0 & \\ 0 & 0 & 0 & \ldots & & & & 0 & 2\exp\left(-\frac{T}{T_{c_{N_{MU}}}}\right) & -\exp\left(-2\frac{T}{T_{c_{N_{MU}}}}\right) & \frac{T^2}{T_{c_{MU}}}\exp\left(1-\frac{T}{T_{c_{N_{MU}}}}\right) \end{array}\right] \qquad (7.7)$$

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

$$a = A.Av \qquad (7.8)$$

where each elemement o a is the activation signal of a motor unit.

Definition at line 115 of file MotorUnitPool.py.

7.6.4.5 MotorUnitPool.MotorUnitPool.an

Is a vector formed as:

$$Av(n) = \left[\begin{array}{cccccccccccc} a_1(n-1) & a_1(n-2) & e_1(n-1) & \ldots & a_i(n-i) & a_i(n-2) & e_i(n-1) & \ldots & a_{N_{MU}(n-1)} & a_{N_{MU}(n-2)} & e_{N_{MU}}(n-1) \end{array} \right]^T \qquad (7.9)$$

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

Definition at line 129 of file MotorUnitPool.py.

7.6.4.6 MotorUnitPool.MotorUnitPool.atualizeForce

Definition at line 156 of file MotorUnitPool.py.

7.6.4.7 MotorUnitPool.MotorUnitPool.bSat

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

Definition at line 135 of file MotorUnitPool.py.

7.6.4.8 MotorUnitPool.MotorUnitPool.conf

Configuration object with the simulation parameters.

Definition at line 59 of file MotorUnitPool.py.

7.6.4.9 MotorUnitPool.MotorUnitPool.diracDeltaValue

Dirac's delta approximation amplitude value.

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

Definition at line 149 of file MotorUnitPool.py.

7.6.4.10 MotorUnitPool.MotorUnitPool.force

Muscle force along time, in N.

Definition at line 153 of file MotorUnitPool.py.

7.6.4.11 MotorUnitPool.MotorUnitPool.hillModel

String indicating whther a Hill model is used or not.

For now, it can be *No*.

Definition at line 155 of file MotorUnitPool.py.

7.6.4.12 MotorUnitPool.MotorUnitPool.kind

Indicates that is Motor Unit pool.

Definition at line 56 of file MotorUnitPool.py.

7.6.4.13 MotorUnitPool.MotorUnitPool.MUnumber

Number of motor units.

Definition at line 66 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 61 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 81 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 83 of file MotorUnitPool.py.

7.6.4.17 MotorUnitPool.MotorUnitPool.timeIndex

Definition at line 158 of file MotorUnitPool.py.

7.6.4.18 MotorUnitPool.MotorUnitPool.twitchAmp_N

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

Definition at line 139 of file MotorUnitPool.py.

7.6.4.19 MotorUnitPool.MotorUnitPool.twTet

Twitch- tetanus relationship (see atualizeForceNoHill function explanation)

Definition at line 137 of file MotorUnitPool.py.

7.6.4.20 MotorUnitPool.MotorUnitPool.unit

List of MotorUnit objects.

Definition at line 69 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__ (self, conf, pool)
  Constructor.
- def atualizePool (self, t)
- def listSpikes (self)

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__ (self, conf, pool, index)
  Constructor.
- def atualizeNeuralTractUnit (self, t, FR)
- def transmitSpikes (self, t)

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.

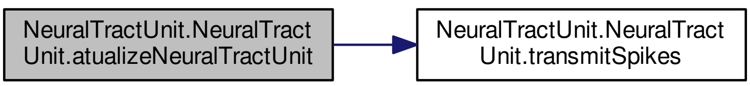
Definition at line 27 of file NeuralTractUnit.py.

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.

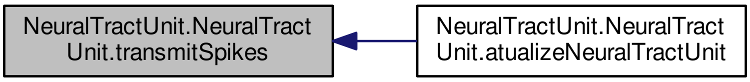
Here is the call graph for this function:



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 object Class Reference

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

- NeuralTract.py

7.10 PointProcessGenerator.PointProcessGenerator Class Reference

Generator of point processes.

Public Member Functions

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

Public Attributes

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

7.10.1 Detailed Description

Generator of point processes.

Definition at line 46 of file PointProcessGenerator.py.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 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.

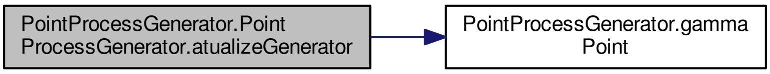
7.10.3 Member Function Documentation

7.10.3.1 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:



7.10.4 Member Data Documentation

7.10.4.1 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.

7.10.4.2 PointProcessGenerator.PointProcessGenerator.GammaOrderInv

Inverse of the GammaOrder.

This is necessary for computational efficiency.

Definition at line 63 of file PointProcessGenerator.py.

7.10.4.3 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.

7.10.4.4 PointProcessGenerator.PointProcessGenerator.points

List of spike instants of the generator.

Definition at line 76 of file PointProcessGenerator.py.

7.10.4.5 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.

7.10.4.6 PointProcessGenerator.PointProcessGenerator.y

Auxiliary variable cummulating 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

7.11 PulseConductanceState.PulseConductanceState Class Reference

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

Public Member Functions

- def __init__ (self, kind, conf, pool, index)

  Initializes the pulse conductance state.
- def changeState (self, t)

  Void function that modify the current situation (true/false) of the state.
- def computeStateValue (self, t)

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

Public Attributes

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

7.11.1 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.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 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.

7.11.3 Member Function Documentation

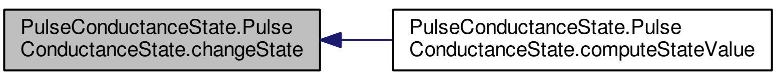
7.11.3.1 def PulseConductanceState.PulseConductanceState.changeState (*self, t*)

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:



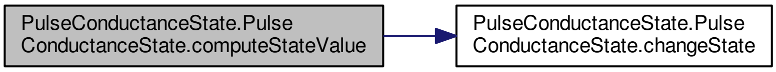
7.11.3.2 def PulseConductanceState.PulseConductanceState.computeStateValue (*self*, *t*)

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:



7.11.4 Member Data Documentation

7.11.4.1 PulseConductanceState.PulseConductanceState.actType

Definition at line 80 of file PulseConductanceState.py.

7.11.4.2 PulseConductanceState.PulseConductanceState.alpha_ms1

Definition at line 76 of file PulseConductanceState.py.

7.11.4.3 PulseConductanceState.PulseConductanceState.beta_ms1

Definition at line 75 of file PulseConductanceState.py.

7.11.4.4 PulseConductanceState.PulseConductanceState.computeValueOff

Definition at line 90 of file PulseConductanceState.py.

7.11.4.5 PulseConductanceState.PulseConductanceState.computeValueOn

Definition at line 89 of file PulseConductanceState.py.

7.11.4.6 PulseConductanceState.PulseConductanceState.kind

Definition at line 66 of file PulseConductanceState.py.

7.11.4.7 PulseConductanceState.PulseConductanceState.PulseDur_ms

Definition at line 77 of file PulseConductanceState.py.

7.11.4.8 PulseConductanceState.PulseConductanceState.state

Definition at line 73 of file PulseConductanceState.py.

7.11.4.9 PulseConductanceState.PulseConductanceState.t0

Definition at line 71 of file PulseConductanceState.py.

7.11.4.10 PulseConductanceState.PulseConductanceState.v0

Definition at line 70 of file PulseConductanceState.py.

7.11.4.11 PulseConductanceState.PulseConductanceState.value

Definition at line 67 of file PulseConductanceState.py.

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

- PulseConductanceState.py

7.12 Synapse.Synapse Class Reference

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

Public Member Functions

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

  Constructor.

Public Attributes

- pool
- kind
- neuronKind
- EqPot_mV
- alpha_ms1
- beta_ms1
- Tmax_mM
- tPeak_ms

  Pulse duration, in ms.
- gmax_muS
- delay_ms
- dynamics
- gMaxTot_muS

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

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

  Time constant during a pulse, in ms.
- tauOff

  Time constant after a pulse, in ms.
- expFinish

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

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

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

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

  Instant that the last spike arrived to the compartment.
- conductanceState
- tBeginOfPulse
- tEndOfPulse
- ri

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

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

7.12.1 Detailed Description

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

Definition at line 305 of file Synapse.py.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 def Synapse.Synapse.__init__ (*self, conf, pool, index, compartment, kind, neuronKind*)

Constructor.

- Input:
  - **conf**:
  - **pool**:
  - **index**:
  - **compartment**:
  - **kind**:
  - **neuronKind**:

Definition at line 323 of file Synapse.py.

7.12.3 Member Data Documentation

7.12.3.1 Synapse.Synapse.alpha_ms1

Definition at line 329 of file Synapse.py.

7.12.3.2 Synapse.Synapse.beta_ms1

Definition at line 330 of file Synapse.py.

7.12.3.3 Synapse.Synapse.conductanceState

Definition at line 376 of file Synapse.py.

7.12.3.4 Synapse.Synapse.delay_ms

Definition at line 336 of file Synapse.py.

7.12.3.5 Synapse.Synapse.dynamics

Definition at line 337 of file Synapse.py.

7.12.3.6 Synapse.Synapse.EqPot_mV

Definition at line 328 of file Synapse.py.

7.12.3.7 Synapse.Synapse.expFinish

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

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

Definition at line 358 of file Synapse.py.

7.12.3.8 Synapse.Synapse.gmax_muS

Definition at line 335 of file Synapse.py.

7.12.3.9 Synapse.Synapse.gMaxTot_muS

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

Definition at line 341 of file Synapse.py.

7.12.3.10 Synapse.Synapse.kind

Definition at line 325 of file Synapse.py.

7.12.3.11 Synapse.Synapse.neuronKind

Definition at line 326 of file Synapse.py.

7.12.3.12 Synapse.Synapse.Non

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

(

Definition at line 363 of file Synapse.py.

7.12.3.13 Synapse.Synapse.numberOfIncomingSynapses

Definition at line 342 of file Synapse.py.

7.12.3.14 Synapse.Synapse.pool

Definition at line 324 of file Synapse.py.

7.12.3.15 Synapse.Synapse.ri

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

Definition at line 382 of file Synapse.py.

7.12.3.16 Synapse.Synapse.rInf

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

Definition at line 348 of file Synapse.py.

7.12.3.17 Synapse.Synapse.Roff

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

Definition at line 372 of file Synapse.py.

7.12.3.18 Synapse.Synapse.Ron

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

Definition at line 367 of file Synapse.py.

7.12.3.19 Synapse.Synapse.t0

Instant that the last spike arrived to the compartment.

Definition at line 374 of file Synapse.py.

7.12.3.20 Synapse.Synapse.tauOff

Time constant after a pulse, in ms.

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

Definition at line 354 of file Synapse.py.

7.12.3.21 Synapse.Synapse.tauOn

Time constant during a pulse, in ms.

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

Definition at line 351 of file Synapse.py.

7.12.3.22 Synapse.Synapse.tBeginOfPulse

Definition at line 377 of file Synapse.py.

7.12.3.23 Synapse.Synapse.tEndOfPulse

Definition at line 378 of file Synapse.py.

7.12.3.24 Synapse.Synapse.ti

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

Definition at line 385 of file Synapse.py.

7.12.3.25 Synapse.Synapse.Tmax_mM

Definition at line 331 of file Synapse.py.

7.12.3.26 Synapse.Synapse.tPeak_ms

Pulse duration, in ms.

Definition at line 333 of file Synapse.py.

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

- Synapse.py

7.13 SynapsesFactory.SynapsesFactory Class Reference

Class to build all the synapses in the system.

Public Member Functions

- def __init__ (self, conf, pools)

  Constructor.

Public Attributes

- numberOfSynapses

7.13.1 Detailed Description

Class to build all the synapses in the system.

Definition at line 15 of file SynapsesFactory.py.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 def SynapsesFactory.SynapsesFactory.__init__ (*self*, *conf*, *pools*)

Constructor.

Definition at line 24 of file SynapsesFactory.py.

7.13.3 Member Data Documentation

7.13.3.1 SynapsesFactory.SynapsesFactory.numberOfSynapses

Definition at line 26 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 (area, specificRes)
  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 (cytR, lComp1, lComp2, dComp1, dComp2)
  Calculates the coupling conductance between two compartments.
- def MotorUnit.compGCouplingMatrix (gc)
  Computes the Coupling Matrix to be used in the dVdt function of the N compartments of the motor unit.
- def MotorUnit.runge_kutta (derivativeFunction, t, x, timeStep, timeStepByTwo, timeStepBySix)
  Function to implement the fourth order Runge-Kutta Method to solve numerically a differential equation.

8.6 MotorUnitPool.py File Reference

Classes

- class MotorUnitPool.MotorUnitPool
  Class that implements a motor unit pool.

Namespaces

- MotorUnitPool

Functions

- def MotorUnitPool.twitchSaturation (activationsat, b)
  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 (GammaOrder, GammaOrderInv)

    *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 (v0, alpha, beta, t, t0)

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

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

8.11 simulation.py File Reference

Namespaces

- simulation

Functions

- def simulation.simulador ()

8.12 Synapse.py File Reference

Classes

- class Synapse.Synapse

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

Namespaces

- Synapse

Functions

- def Synapse.compSynapCond (Gmax, Ron, Roff)

  Computes the synaptic conductance.
- def Synapse.compRon (Non, rInf, Ron, t0, t, tauOn)

  Computes the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that have neurotransmitters being released (during the pulse).
- def Synapse.compRoff (Roff, t0, t, tauOff)

  Computes the fraction of postsynaptic receptors that are bound to neurotransmitters of all the individual synapses that do not have neurotransmitters being released (before and after the pulse).
- def Synapse.compRiStart (ri, t, ti, tPeak, tauOff)

  Computes the fraction of bound postsynaptic receptors to neurotransmitters in individual synapses when the neurotransmitter begin (begin of the pulse).
- def Synapse.compRiStop (rInf, ri, expFinish)

  Computes the fraction of bound postsynaptic receptors to neurotransmitters in individual synapses when the neurotransmitter release stops (the pulse ends).
- def Synapse.compRonStart (Ron, ri, synContrib)

  Incorporates a new conductance to the set of conductances during a pulse.
- def Synapse.compRoffStart (Roff, ri, synContrib)

  Incorporates a new conductance to the set of conductances that are not during a pulse.
- def Synapse.compRonStop (Ron, ri, synContrib)

  Removes a conductance from the set of conductances during a pulse.
- def Synapse.compRoffStop (Roff, ri, synContrib)

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

8.13 SynapsesFactory.py File Reference

Classes

- class SynapsesFactory.SynapsesFactory

  Class to build all the synapses in the system.

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, 33
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
iIonic
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
iIonic, 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