BluePyOpt Documentation

Release 1.2.post.dev159

BBP, EPFL

February 21, 2017

Contents

The Blue Brain Python Optimisation Library (BluePyOpt) is an extensible framework for data-driven model parameter optimisation that wraps and standardises several existing open-source tools.

It simplifies the task of creating and sharing these optimisations, and the associated techniques and knowledge. This is achieved by abstracting the optimisation and evaluation tasks into various reusable and flexible discrete elements according to established best-practices.

Further, BluePyOpt provides methods for setting up both small- and large-scale optimisations on a variety of platforms, ranging from laptops to Linux clusters and cloud-based compute infrastructures.

CHAPTER 1

Python API

General optimisation API

bluepyopt.optimisations	Optimisation class
bluepyopt.parameters	Parameter classes
bluepyopt.objectives	Objective classes
bluepyopt.evaluators	Cell evaluator class

bluepyopt.optimisations

Optimisation class

class bluepyopt.optimisations.**Optimisation**(*evaluator=None*)

Bases: object

Optimisation class

Constructor

bluepyopt.parameters

Parameter classes

class bluepyopt.parameters.**Parameter**(*name*, *value=None*, *frozen=False*, *bounds=None*)

Bases: object

Base parameter class

Constructor

check_bounds()

Check if parameter is within bounds

freeze(*value*)

Freeze parameter to certain value

lower_bound

Lower bound

unfreeze()

Unfreeze parameter

upper_bound
Lower bound

value
Parameter value

bluepyopt.objectives

Objective classes

class bluepyopt.objectives.**Objective**(*name*, *value=None*)
Bases: object

Objective of the optimisation algorithm

Constructor

bluepyopt.evaluators

Cell evaluator class

class bluepyopt.evaluators.**Evaluator**(*objectives=None*, *params=None*)
Bases: object

Evaluator class

An Evaluator maps a set of parameter values to objective values

Args:

objectives (Objectives): The objectives that will be the output of the evaluator.

params (Parameters): The parameters that will be evaluated.

Attributes:

objectives (Objectives): Objective objects.

params (Objectives): Parameter objects.

evaluate_with_dicts(*param_dict*)
Evaluate parameter a parameter set (abstract).

Parameters **params** (*dict with values Parameters, and keys parameter names*) – The parameter values to be evaluated.

Returns **objectives** – Dict of Objective with values calculated by the Evaluator.

Return type dict with values Parameters, and keys objective names

evaluate_with_lists(*params*)
Evaluate parameter a parameter set (abstract).

Parameters **params** (*list of Parameters*) – The parameter values to be evaluated.

Returns **objectives** – List of Objectives with values calculated by the Evaluator.

Return type list of Objectives

EPhys model API

bluepyopt.ephys.evaluators	Cell evaluator class
bluepyopt.ephys.models	Cell template class
bluepyopt.ephys.efeatures	eFeature classes
bluepyopt.ephys.locations	Location classes
bluepyopt.ephys.mechanisms	Mechanism classes
bluepyopt.ephys.morphologies	Morphology classes
bluepyopt.ephys.objectives	Objective classes
bluepyopt.ephys.parameters	Parameter classes
bluepyopt.ephys.parameterscalers	Parameter scaler classes
bluepyopt.ephys.protocols	Protocol classes
bluepyopt.ephys.recordings	Recording classes
bluepyopt.ephys.responses	Responses classes
bluepyopt.ephys.objectivescalculators	Score calculator classes
bluepyopt.ephys.stimuli	Stimuli classes

bluepyopt.ephys.evaluators

Cell evaluator class

class bluepyopt.ephys.evaluators.**CellEvaluator**(*cell_model=None*, *param_names=None*, *fitness_protocols=None*, *fitness_calculator=None*, *isolate_protocols=None*, *sim=None*, *use_params_for_seed=False*)

Bases: *bluepyopt.evaluators.Evaluator*

Simple cell class

Constructor

Parameters

- **cell_model** (*ephys.models.CellModel*) – CellModel object to evaluate
- **param_names** (*list of str*) – names of the parameters (parameters will be initialised in this order)
- **fitness_protocols** (*dict of str -> ephys.protocols.Protocol*) – protocols used during the fitness evaluation
- **fitness_calculator** (*ObjectivesCalculator*) – ObjectivesCalculator object used for the transformation of Responses into Objective objects
- **isolate_protocols** (*bool*) – whether to use multiprocessing to isolate the simulations (disabling this could lead to unexpected behavior, and might hinder the reproducability of the simulations)
- **sim** (*ephys.simulators.NrnSimulator*) – simulator to use for the cell evaluation
- **use_params_for_seed** (*bool*) – use a hashed version of the parameter dictionary as a seed for the simulator

evaluate(*param_list=None*)
Run evaluation with lists as input and outputs

evaluate_with_dicts(*param_dict=None*)
Run evaluation with dict as input and output

evaluate_with_lists (*param_list=None*)
Run evaluation with lists as input and outputs

objective_dict (*objective_array*)
Convert objective_array in objective_dict

objective_list (*objective_dict*)
Convert objective_dict in objective_list

param_dict (*param_array*)
Convert param_array in param_dict

run_protocol (*protocol*, *param_values*, *isolate=None*, *cell_model=None*, *sim=None*)
Run protocol

run_protocols (*protocols*, *param_values*)
Run a set of protocols

static seed_from_param_dict (*param_dict*)
Return a seed value based on a param_dict

bluepyopt.ephys.models

Cell template class

class bluepyopt.ephys.models.**CellModel** (*name*, *morph=None*, *mechs=None*, *params=None*, *gid=0*)
Bases: *bluepyopt.ephys.models.Model*

Cell model class

Constructor

Parameters

- **name** (*str*) – name of this object should be alphanumeric string, underscores are allowed, first char should be a letter
- **morph** (Morphology) – underlying Morphology of the cell
- **mechs** (*list of Mechanisms*) – Mechanisms associated with the cell
- **params** (*list of Parameters*) – Parameters of the cell model

__str__ ()
Return string representation

check_name ()
Check if name complies with requirements

check_nonfrozen_params (*param_names*)
Check if all nonfrozen params are set

static create_empty_cell (*name*, *sim*, *seclist_names=None*, *secarray_names=None*)
Create an empty cell in Neuron

static create_empty_template (*template_name*, *seclist_names=None*, *secarray_names=None*)
create an hoc template named template_name for an empty cell

create_hoc (*param_values*, *ignored_globals=()*, *template='cell_template.jinja2'*, *disable_banner=False*)
Create hoc code for this model

destroy(*sim=None*)
Destroy instantiated model in simulator

freeze(*param_dict*)
Set params

instantiate(*sim=None*)
Instantiate model in simulator

params_by_names(*param_names*)
Get parameter objects by name

unfreeze(*param_names*)
Unset params

class bluepyopt.ephys.models.**HocCellModel**(*name*, *morphology_path*, *hoc_path=None*, *hoc_string=None*)
Bases: *bluepyopt.ephys.models.CellModel*

Wrapper class for a hoc template so it can be used by BluePyOpt

Constructor

Parameters

- **name** (*str*) – name of this object
- **sim** (*NrnSimulator*) – simulator in which to instatiate hoc_string
- **hoc_path** (*str*) – Path to a hoc file (hoc_path and hoc_string can't be used simultaneously, but one of them has to specified)
- **hoc_string** (*str*) – String that of hoc code that defines a template (hoc_path and hoc_string can't be used simultaneously, but one of them has to specified))
- **morphology_path** (*str path*) – path to morphology that can be loaded by Neuron

__str__()
Return string representation

static get_template_name(*hoc_string*)
Find the template name from hoc_string

Note: this will fail if there is a begintemplate in a /* */ style comment before the real begintemplate

static load_hoc_template(*sim*, *hoc_string*)
Have neuron hoc template, and detect what the name template name is

The template must have an init that takes two parameters, the second of which is the path to a morphology.

It must also have a CellRef member that is the result of *Import3d_GUI(...).instantiate()*

class bluepyopt.ephys.models.**HocMorphology**(*morphology_path*)
Bases: *bluepyopt.ephys.morphologies.Morphology*

wrapper for Morphology so that it has a morphology_path

class bluepyopt.ephys.models.**Model**(*name*)
Bases: object

Constructor :param name: name of the model

destroy(*sim=None*)
Destroy instantiated model in simulator

instantiate(*sim=None*)
Instantiate model in simulator

bluepyopt.ephys.efeatures

eFeature classes

class bluepyopt.ephys.efeatures.**EFeature**(*name=''*, *comment=''*)
Bases: bluepyopt.ephys.base.BaseEPhys

EPhys feature

class bluepyopt.ephys.efeatures.**eFELFeature**(*name*, *efel_feature_name=None*, *recording_names=None*, *stim_start=None*, *stim_end=None*, *exp_mean=None*, *exp_std=None*, *threshold=None*, *stimulus_current=None*, *comment=''*, *interp_step=None*, *double_settings=None*, *int_settings=None*)
Bases: *bluepyopt.ephys.efeatures.EFeature*, bluepyopt.ephys.serializer.DictMixin

eFEL feature

Constructor

Parameters

- **name** (*str*) – name of the eFELFeature object
- **efel_feature_name** (*str*) – name of the eFeature in the eFEL library (ex: 'AP1_peak')
- **recording_names** (*dict*) – eFEL features can accept several recordings as input
- **stim_start** (*float*) – stimulation start time (ms)
- **stim_end** (*float*) – stimulation end time (ms)
- **exp_mean** (*float*) – experimental mean of this eFeature
- **exp_std** (*float*) – experimental standard deviation of this eFeature
- **threshold** (*float*) – spike detection threshold (mV)
- **comment** (*str*) – comment
- **interp_step** (*float*) – interpolation step (ms)
- **double_settings** (*dict*) – dictionary with efel double settings that should be set before extracting the features
- **int_settings** (*dict*) – dictionary with efel int settings that should be set before extracting the features

__str__()
String representation

calculate_feature(*responses*, *raise_warnings=False*)
Calculate feature value

calculate_score(*responses*, *trace_check=False*)
Calculate the score

bluepyopt.ephys.locations

Location classes

exception bluepyopt.ephys.locations.**EPhysLocInstantiateException**(*message*)
Bases: exceptions.Exception

All exception generated by location instantiation

Constructor

class bluepyopt.ephys.locations.**Location**(*name=''*, *comment=''*)
Bases: bluepyopt.ephys.base.BaseEPhys

class bluepyopt.ephys.locations.**NrnPointProcessLocation**(*name*, *pprocess_mech*, *comment=''*)
Bases: *bluepyopt.ephys.locations.Location*

Point process location

Constructor

Parameters

- **name** (*str*) – name of the object
- **pprocess_mech** (*str*) – point process mechanism

__str__()
String representation

instantiate(*sim=None*, *icell=None*)
Find the instantiated point processes

class bluepyopt.ephys.locations.**NrnSeclistCompLocation**(*name*, *seclist_name=None*, *sec_index=None*, *comp_x=None*, *comment=''*)
Bases: *bluepyopt.ephys.locations.Location*, bluepyopt.ephys.serializer.DictMixin

Compartment in a sectionlist

Constructor

Parameters

- **name** (*str*) – name of the object
- **seclist_name** (*str*) – name of Neuron section list (ex: 'somatic')
- **sec_index** (*int*) – index of the section in the section list
- **comp_x** (*float*) – segx (0..1) of segment inside section

__str__()
String representation

instantiate(*sim=None*, *icell=None*)
Find the instantiate compartment

class bluepyopt.ephys.locations.**NrnSeclistLocation**(*name*, *seclist_name=None*, *comment=''*)
Bases: *bluepyopt.ephys.locations.Location*, bluepyopt.ephys.serializer.DictMixin

Section in a sectionlist

Constructor

Parameters

- **name** (*str*) – name of the object
- **seclist_name** (*str*) – name of NEURON section list (ex: 'somatic')

__str__()
String representation

instantiate(*sim=None*, *icell=None*)
Find the instantiate compartment

class bluepyopt.ephys.locations.**NrnSeclistSecLocation**(*name*, *seclist_name=None*, *sec_index=None*, *comment=''*)
Bases: *bluepyopt.ephys.locations.Location*, bluepyopt.ephys.serializer.DictMixin

Section in a sectionlist

Constructor

Parameters

- **name** (*str*) – name of this object
- **seclist_name** (*str*) – name of Neuron section list (ex: 'somatic')
- **sec_index** (*int*) – index of the section

__str__()
String representation

instantiate(*sim=None*, *icell=None*)
Find the instantiate compartment

class bluepyopt.ephys.locations.**NrnSomaDistanceCompLocation**(*name*, *soma_distance=None*, *seclist_name=None*, *comment=''*)
Bases: *bluepyopt.ephys.locations.Location*, bluepyopt.ephys.serializer.DictMixin

Compartment at distance from soma

Constructor

Parameters

- **name** (*str*) – name of this object
- **soma_distance** (*float*) – distance from soma to this segment
- **seclist_name** (*str*) – name of Neuron section list (ex: 'apical')

__str__()
String representation

instantiate(*sim=None*, *icell=None*)
Find the instantiate compartment

bluepyopt.ephys.mechanisms

Mechanism classes

Theses classes represent mechanisms in the model

class bluepyopt.ephys.mechanisms.**Mechanism**(*name=''*, *comment=''*)
Bases: bluepyopt.ephys.base.BaseEPhys

Base parameter class

class bluepyopt.ephys.mechanisms.**NrnMODMechanism**(*name*, *mod_path=None*, *suffix=None*, *locations=None*, *preloaded=True*, *deterministic=True*, *prefix=None*, *comment=''*)

Bases: *bluepyopt.ephys.mechanisms.Mechanism*, bluepyopt.ephys.serializer.DictMixin

Neuron mechanism

Constructor

Parameters

- **name** (*str*) – name of this object
- **mod_path** (*str*) – path to the MOD file (not used for the moment)
- **suffix** (*str*) – suffix of this mechanism in the MOD file
- **locations** (*list of Locations*) – a list of Location objects pointing to where this mechanism should be added to.
- **preloaded** (*bool*) – should this mechanism be side-loaded by BluePyOpt, or was it already loaded and compiled by the user ? (not used for the moment)
- **prefix** (*str*) – Deprecated. Use suffix instead.

__str__()
String representation

destroy(*sim=None*)
Destroy mechanism instantiation

generate_reinitrng_hoc_block()
"Create re_init_rng code blocks for this channel

static hash_hoc(*string*, *sim*)
Calculate hash value of string in Python

static hash_py(*string*)
Calculate hash value of string in Python

instantiate(*sim=None*, *icell=None*)
Instantiate

instantiate_determinism(*deterministic*, *icell*, *isec*, *sim*)
Instantiate enable/disable determinism

prefix
Deprecated, prefix is now replaced by suffix

class bluepyopt.ephys.mechanisms.**NrnMODPointProcessMechanism**(*name*, *mod_path=None*, *suffix=None*, *locations=None*, *preloaded=True*, *comment=''*)

Bases: *bluepyopt.ephys.mechanisms.Mechanism*

Neuron mechanism

Constructor

Parameters

- **name** (*str*) – name of this object

- **mod_path** (*str*) – path to the MOD file (not used for the moment)
- **suffix** (*str*) – suffix of this mechanism in the MOD file
- **locations** (*list of Locations*) – a list of Location objects pointing to compartments where this mechanism should be added to.
- **preloaded** (*bool*) – should this mechanism be side-loaded by BluePyOpt, or was it already loaded and compiled by the user ? (not used for the moment)

__str__()
String representation

destroy(*sim=None*)
Destroy mechanism instantiation

instantiate(*sim=None*, *icell=None*)
Instantiate

bluepyopt.ephys.morphologies

Morphology classes

class bluepyopt.ephys.morphologies.**Morphology**(*name=''*, *comment=''*)
Bases: bluepyopt.ephys.base.BaseEPhys

Morphology class

class bluepyopt.ephys.morphologies.**NrnFileMorphology**(*morphology_path*, *do_replace_axon=False*, *do_set_nseg=True*, *comment=''*, *replace_axon_hoc=None*)
Bases: *bluepyopt.ephys.morphologies.Morphology*, bluepyopt.ephys.serializer.DictMixin

Morphology loaded from a file

Constructor

Parameters

- **morphology_path** (*str*) – location of the file describing the morphology
- **do_replace_axon** (*bool*) – Does the axon need to be replaced by an AIS stub ?
- **replace_axon_hoc** (*str*) – String replacement for the 'replace_axon'
- **in hoc Must include 'proc replace_axon(){ ... } If None,** (*command*) –
- **default replace_axon is used in any created hoc files** (*the*) –

__str__()
Return string representation

destroy(*sim=None*)
Destroy morphology instantiation

instantiate(*sim=None*, *icell=None*)
Load morphology

static replace_axon(*sim=None*, *icell=None*)
Replace axon

static set_nseg(*icell*)
Set the nseg of every section

bluepyopt.ephys.objectives

Objective classes

class bluepyopt.ephys.objectives.**EFeatureObjective**(*name*, *features=None*)
Bases: *bluepyopt.objectives.Objective*

EPhys feature objective

Constructor

Parameters

- **name** (*str*) – name of this object
- **features** (*list of eFeatures*) – features used in the Objective

calculate_feature_scores(*responses*)
Calculate the scores for the individual features

class bluepyopt.ephys.objectives.**MaxObjective**(*name*, *features=None*)
Bases: *bluepyopt.ephys.objectives.EFeatureObjective*

Max of list of EPhys feature

Constructor

Parameters

- **name** (*str*) – name of this object
- **features** (*list of eFeatures*) – features used in the Objective

calculate_score(*responses*)
Objective score

class bluepyopt.ephys.objectives.**SingletonObjective**(*name*, *feature*)
Bases: *bluepyopt.ephys.objectives.EFeatureObjective*

Single EPhys feature

Constructor

Parameters

- **name** (*str*) – name of this object
- **features** (EFeature) – single eFeature inside this objective

__str__()
String representation

calculate_score(*responses*)
Objective score

class bluepyopt.ephys.objectives.**WeightedSumObjective**(*name*, *features*, *weights*)
Bases: *bluepyopt.ephys.objectives.EFeatureObjective*

Weighted sum of list of eFeatures

Constructor

Parameters

- **name** (*str*) – name of this object
- **features** (*list of EFeatures*) – eFeatures in the objective
- **weights** (*list of float*) – weights of the eFeatures

calculate_score(*responses*)
Objective score

bluepyopt.ephys.parameters

Parameter classes

class bluepyopt.ephys.parameters.**MetaParameter**(*name*, *obj=None*, *attr_name=None*, *value=None*, *frozen=False*, *bounds=None*)
Bases: *bluepyopt.ephys.parameters.NrnParameter*

Parameter class that controls attributes of other objects

Constructor

__str__()
String representation

value
Parameter value

class bluepyopt.ephys.parameters.**NrnGlobalParameter**(*name*, *value=None*, *frozen=False*, *bounds=None*, *param_name=None*)
Bases: *bluepyopt.ephys.parameters.NrnParameter*, bluepyopt.ephys.serializer.DictMixin

Parameter set in the global namespace of neuron

Contructor

Parameters

- **name** (*str*) – name of this object
- **value** (*float*) – Value for the parameter, required if Frozen=True
- **frozen** (*bool*) – Whether the parameter can be varied, or its values
- **permently set** (*is*) –
- **bounds** (*indexable*) – two elements; the lower and upper bounds (Optional)
- **param_name** (*str*) – name used within NEURON

__str__()
String representation

instantiate(*sim=None*, *icell=None*)
Instantiate

class bluepyopt.ephys.parameters.**NrnParameter**(*name*, *value=None*, *frozen=False*, *bounds=None*)
Bases: *bluepyopt.parameters.Parameter*

Abstract Parameter class for Neuron object parameters

Contructor

destroy(*sim=None*)
Remove parameter from the simulator

instantiate(*sim=None*, *icell=None*)
Instantiate the parameter in the simulator

class bluepyopt.ephys.parameters.**NrnPointProcessParameter**(*name*, *value=None*, *frozen=False*, *bounds=None*, *locations=None*, *param_name=None*)
Bases: *bluepyopt.ephys.parameters.NrnParameter*, bluepyopt.ephys.serializer.DictMixin

Parameter of a section

Constructor

Parameters

- **name** (*str*) – name of the Parameter
- **value** (*float*) – Value for the parameter, required if Frozen=True
- **frozen** (*bool*) – Whether the parameter can be varied, or its values
- **permently set** (*is*) –
- **bounds** (*indexable*) – two elements; the lower and upper bounds (Optional)
- **locations** – an iterator of the point process locations you want to set the parameters of
- **param_name** (*str*) – name of parameter used within the point process

__str__()
String representation

instantiate(*sim=None*, *icell=None*)
Instantiate

class bluepyopt.ephys.parameters.**NrnRangeParameter**(*name*, *value=None*, *frozen=False*, *bounds=None*, *param_name=None*, *value_scaler=None*, *locations=None*)
Bases: *bluepyopt.ephys.parameters.NrnParameter*, bluepyopt.ephys.serializer.DictMixin

Parameter that has a range over a section

Contructor

Parameters

- **name** (*str*) – name of the Parameter
- **value** (*float*) – Value for the parameter, required if Frozen=True
- **frozen** (*bool*) – Whether the parameter can be varied, or its values
- **permently set** (*is*) –
- **bounds** (*indexable*) – two elements; the lower and upper bounds (Optional)
- **param_name** (*str*) – name used within NEURON
- **value_scaler** (*float*) – value used to scale the parameter value
- **locations** (*list of ephys.locations.Location*) – locations on which to instantiate the parameter

__str__()
String representation

instantiate(*sim=None*, *icell=None*)
Instantiate

class bluepyopt.ephys.parameters.**NrnSectionParameter**(*name*, *value=None*, *frozen=False*, *bounds=None*, *param_name=None*, *value_scaler=None*, *locations=None*)
Bases: *bluepyopt.ephys.parameters.NrnParameter*, bluepyopt.ephys.serializer.DictMixin

Parameter of a section

Contructor

Parameters

- **name** (*str*) – name of the Parameter
- **value** (*float*) – Value for the parameter, required if Frozen=True
- **frozen** (*bool*) – Whether the parameter can be varied, or its values
- **permently set** (*is*) –
- **bounds** (*indexable*) – two elements; the lower and upper bounds (Optional)
- **param_name** (*str*) – name used within NEURON
- **value_scaler** (*float*) – value used to scale the parameter value
- **locations** (*list of ephys.locations.Location*) – locations on which to instantiate the parameter

__str__()
String representation

instantiate(*sim=None*, *icell=None*)
Instantiate

bluepyopt.ephys.parameterscalers

Parameter scaler classes

class bluepyopt.ephys.parameterscalers.**MissingFormatDict**
Bases: dict

Extend dict for string formatting with missing values

__missing__(*key*)
Return string with format key for missing keys

class bluepyopt.ephys.parameterscalers.**NrnSegmentLinearScaler**(*name=None*, *multiplier=1.0*, *offset=0.0*, *comment=''*)
Bases: *bluepyopt.ephys.parameterscalers.ParameterScaler*, bluepyopt.ephys.serializer.DictMixin

Linear scaler

Constructor

Parameters

- **name** (*str*) – name of this object

- **multiplier** (*float*) – slope of the linear scaler
- **offset** (*float*) – intercept of the linear scaler

__str__()
String representation

scale(*value*, *segment=None*, *sim=None*)
Scale a value based on a segment

class bluepyopt.ephys.parameterscalers.**NrnSegmentSomaDistanceScaler**(*name=None*, *distribution=None*, *comment=''*, *dist_param_names=None*)
Bases: *bluepyopt.ephys.parameterscalers.ParameterScaler*, bluepyopt.ephys.serializer.DictMixin

Scaler based on distance from soma

Constructor

Parameters

- **name** (*str*) – name of this object
- **distribution** (*str*) – distribution of parameter dependent on distance from soma. string can contain *distance* and/or *value* as placeholders for the distance to the soma and parameter value respectivily
- **dist_params** (*list*) – list of names of parameters that parametrise the distribution. These names will become attributes of this object. The distribution string should contain these names, and they will be replaced by values of the corresponding attributes

__str__()
String representation

eval_dist(*value*, *distance*)
Create the final dist string

inst_distribution
The instantiated distribution

scale(*value*, *segment*, *sim=None*)
Scale a value based on a segment

class bluepyopt.ephys.parameterscalers.**ParameterScaler**(*name=''*, *comment=''*)
Bases: bluepyopt.ephys.base.BaseEPhys

Parameter scalers

bluepyopt.ephys.parameterscalers.**format_float**(*value*)
Return formatted float string

bluepyopt.ephys.protocols

Protocol classes

class bluepyopt.ephys.protocols.**Protocol**(*name=None*)
Bases: object

Class representing a protocol (stimulus and recording).

Constructor

Parameters **name** (*str*) – name of the feature

class bluepyopt.ephys.protocols.**SequenceProtocol**(*name=None*, *protocols=None*)
Bases: *bluepyopt.ephys.protocols.Protocol*

A protocol consisting of a sequence of other protocols

Constructor

Parameters

- **name** (*str*) – name of this object
- **protocols** (*list of Protocols*) – subprotocols this protocol consists of

run(*cell_model*, *param_values*, *sim=None*, *isolate=None*)
Instantiate protocol

subprotocols()
Return subprotocols

class bluepyopt.ephys.protocols.**StepProtocol**(*name=None*, *step_stimulus=None*, *holding_stimulus=None*, *recordings=None*, *cvode_active=None*)
Bases: *bluepyopt.ephys.protocols.SweepProtocol*

Protocol consisting of step and holding current

Constructor

Parameters

- **name** (*str*) – name of this object
- **step_stimulus** (*list of Stimuli*) – Stimulus objects used in protocol
- **recordings** (*list of Recordings*) – Recording objects used in the protocol
- **cvode_active** (*bool*) – whether to use variable time step

step_delay
Time stimulus starts

step_duration
Time stimulus starts

class bluepyopt.ephys.protocols.**SweepProtocol**(*name=None*, *stimuli=None*, *recordings=None*, *cvode_active=None*)
Bases: *bluepyopt.ephys.protocols.Protocol*

Sweep protocol

Constructor

Parameters

- **name** (*str*) – name of this object
- **stimuli** (*list of Stimuli*) – Stimulus objects used in the protocol
- **recordings** (*list of Recordings*) – Recording objects used in the protocol
- **cvode_active** (*bool*) – whether to use variable time step

__str__()
String representation

destroy(*sim=None*)
Destroy protocol

instantiate(*sim=None*, *icell=None*)
Instantiate

run(*cell_model*, *param_values*, *sim=None*, *isolate=None*)
Instantiate protocol

subprotocols()
Return subprotocols

total_duration
Total duration

bluepyopt.ephys.recordings

Recording classes

class bluepyopt.ephys.recordings.**CompRecording**(*name=None*, *location=None*, *variable='v'*)
Bases: *bluepyopt.ephys.recordings.Recording*

Response to stimulus

Constructor

Parameters

- **name** (*str*) – name of this object
- **location** (Location) – location in the model of the recording
- **variable** (*str*) – which variable to record from (e.g. 'v')

__str__()
String representation

destroy(*sim=None*)
Destroy recording

instantiate(*sim=None*, *icell=None*)
Instantiate recording

response
Return recording response

class bluepyopt.ephys.recordings.**Recording**(*name=None*)
Bases: object

Class to represent object that record variables during simulations

Constructor

Parameters **name** (*str*) – name of this object

bluepyopt.ephys.responses

Responses classes

class bluepyopt.ephys.responses.**Response**(*name*)
Bases: object

Response to stimulus

Constructor

Parameters **name** (*str*) – name of this object

class bluepyopt.ephys.responses.**TimeVoltageResponse**(*name*, *time=None*, *voltage=None*)
Bases: *bluepyopt.ephys.responses.Response*

Response to stimulus

Constructor

Parameters

- **name** (*str*) – name of this object
- **time** (*list of floats*) – time series
- **voltage** (*list of floats*) – voltage series

__getitem__(*index*)
Return item at index

plot(*axes*)
Plot the response

read_csv(*filename*)
Load response from csv file

to_csv(*filename*)
Write response to csv file

bluepyopt.ephys.objectivescalculators

Score calculator classes

class bluepyopt.ephys.objectivescalculators.**ObjectivesCalculator**(*objectives=None*)
Bases: object

Score calculator

Constructor

Parameters **objectives** (*list of Objective*) – objectives over which to calculate

calculate_scores(*responses*)
Calculator the score for every objective

bluepyopt.ephys.stimuli

Stimuli classes

class bluepyopt.ephys.stimuli.**NrnCurrentPlayStimulus**(*time_points=None*, *current_points=None*, *location=None*)
Bases: *bluepyopt.ephys.stimuli.Stimulus*

Current stimulus based on current amplitude and time series

Constructor

Parameters

- **time_points()** – time series (ms)
- **current_points()** – current series of injected current amplitudes (nA)
- **location** (Location) – location of stimulus

__str__()
String representation

destroy(*sim=None*)
Destroy stimulus

instantiate(*sim=None*, *icell=None*)
Run stimulus

class bluepyopt.ephys.stimuli.**NrnNetStimStimulus**(*locations=None*, *total_duration=None*, *interval=None*, *number=None*, *start=None*, *noise=0*, *weight=1*)
Bases: *bluepyopt.ephys.stimuli.Stimulus*

Current stimulus based on current amplitude and time series

Constructor

Parameters

- **location** – synapse point process location to connect to
- **interval** – time between spikes (ms)
- **number** – average number of spikes
- **start** – most likely start time of first spike (ms)
- **noise** – fractional randomness (0 deterministic, 1 negexp interval distrubtion)

__str__()
String representation

destroy(*sim=None*)
Destroy stimulus

instantiate(*sim=None*, *icell=None*)
Run stimulus

class bluepyopt.ephys.stimuli.**NrnRampPulse**(*ramp_amplitude_start=None*, *ramp_amplitude_end=None*, *ramp_delay=None*, *ramp_duration=None*, *total_duration=None*, *location=None*)
Bases: *bluepyopt.ephys.stimuli.Stimulus*

Ramp current clamp injection

Constructor

Parameters

- **ramp_amplitude_start** (*float*) – amplitude at start of ramp (nA)
- **ramp_amplitude_start** – amplitude at end of ramp (nA)
- **ramp_delay** (*float*) – delay of ramp (ms)
- **ramp_duration** (*float*) – duration oframp (ms)
- **total_duration** (*float*) – total duration (ms)

- **location** (Location) – stimulus Location

__str__()
String representation

destroy(*sim=None*)
Destroy stimulus

instantiate(*sim=None*, *icell=None*)
Run stimulus

class bluepyopt.ephys.stimuli.**NrnSquarePulse**(*step_amplitude=None*, *step_delay=None*, *step_duration=None*, *total_duration=None*, *location=None*)
Bases: *bluepyopt.ephys.stimuli.Stimulus*

Square pulse current clamp injection

Constructor

Parameters

- **step_amplitude** (*float*) – amplitude (nA)
- **step_delay** (*float*) – delay (ms)
- **step_duration** (*float*) – duration (ms)
- **total_duration** (*float*) – total duration (ms)
- **location** (Location) – stimulus Location

__str__()
String representation

destroy(*sim=None*)
Destroy stimulus

instantiate(*sim=None*, *icell=None*)
Run stimulus

class bluepyopt.ephys.stimuli.**Stimulus**
Bases: object

Stimulus protocol

Deap extension API

bluepyopt.deapext.optimisations	Optimisation class

bluepyopt.deapext.optimisations

Optimisation class

class bluepyopt.deapext.optimisations.**DEAPOptimisation**(*evaluator=None*, *use_scoop=False*, *seed=1*, *offspring_size=10*, *eta=10*, *mutpb=1.0*, *cxpb=1.0*, *map_function=None*, *hof=None*, *selector_name=None*)

Bases: *bluepyopt.optimisations.Optimisation*

DEAP Optimisation class

Constructor

Parameters

- **evaluator** (*Evaluator*) – Evaluator object
- **seed** (*float*) – Random number generator seed
- **offspring_size** (*int*) – Number of offspring individuals in each generation
- **eta** (*float*) – Parameter that controls how far the crossover and
- **operator disturbe the original individuals** (*mutation*) –
- **mutpb** (*float*) – Mutation probability
- **cxpb** (*float*) – Crossover probability
- **map_function** (*function*) – Function used to map (parallelise) the evaluation function calls
- **hof** (*hof*) – Hall of Fame object
- **selector_name** (*str*) – The selector used in the evolutionary algorithm, possible values are 'IBEA' or 'NSGA2'

run(*max_ngen=10*, *offspring_size=None*, *continue_cp=False*, *cp_filename=None*, *cp_frequency=1*)
Run optimisation

setup_deap()
Set up optimisation

class bluepyopt.deapext.optimisations.**IBEADEAPOptimisation**(**args*, ***kwargs*)
Bases: *bluepyopt.deapext.optimisations.DEAPOptimisation*

IBEA DEAP class

Constructor

class bluepyopt.deapext.optimisations.**WSListIndividual**(**args*, ***kwargs*)
Bases: list

Individual consisting of list with weighted sum field

Constructor

class bluepyopt.deapext.optimisations.**WeightedSumFitness**(*values=()*, *obj_size=None*)
Bases: deap.base.Fitness

Fitness that compares by weighted sum

__deepcopy__(*_*)
Override deepcopy

sum
Weighted sum of values

weighted_sum
Weighted sum of wvalues

Indices and tables

- genindex
- modindex
- search

b

Symbols

B

C

D

E

F

G

H

O

P

R

S

T

U

V

W