Definitions

GUI-editable parameters

Regular classes

CACLALearner

- **Alpha** (*NumericValue subclass*) : Learning gain [0..1]
- **Policy** (*Policy subclass*) : The policy to be learned

IncrementalNaturalActorCritic

- **VFunction** (*LinearStateVFA*) : The Value-function
- **V-ETraces** (*ETraces*) : Traces used by the critic
- **Alpha-v** (*NumericValue subclass*) : Learning gain used by the critic
- **Alpha-r** (*NumericValue subclass*) : Learning gain used to average the reward
- **Alpha-u** (*NumericValue subclass*) : Learning gain used by the actor
- **Policy** (*Multiple Policy subclass*) : The policy

OffPolicyActorCritic

- **VFunction** (*LinearStateVFA*) : The Value-function
- **ETraces** (*ETraces*) : Traces used by the critic and the actor
- **Alpha-v** (*NumericValue subclass*) : Learning gain used by the critic
- **Alpha-w** (*NumericValue subclass*) : Learning gain used to average the reward
- **Alpha-u** (*NumericValue subclass*) : Learning gain used by the actor
- **Policy** (*Multiple Policy subclass*) : The policy

OffPolicyDeterministicActorCritic

- **QFunction** (*LinearStateActionVFA*) : The Q-function
- **Alpha-w** (*NumericValue subclass*) : Learning gain used by the critic
- **Alpha-theta** (*NumericValue subclass*) : Learning gain used by the actor
- **Policy** (*Multiple DeterministicPolicy subclass*) : The deterministic policy

ActorCritic

- **Actor** (*Actor*) : The actor
- **Critic** (*ICritic subclass*) : The critic

RegularPolicyGradientLearner

- **Alpha** (*NumericValue subclass*) : The learning gain
- **Policy** (*Policy subclass*) : The policy to be learned

Actor

- **Base-Controller** (*Controller subclass*) : The base controller used to initialize the weights of the actor
- **Output** (*Multiple PolicyLearner subclass*) : The outputs of the actor. One for each output dimension

SimionApp

- **Log** (*Logger*) : The logger class
- **World** (*World*) : The simulation environment and its parameters
- **Experiment** (*Experiment*) : The parameters of the experiment
- **SimGod** (*SimGod*) : The omniscient class that controls all aspects of the simulation process

AsyncQLearning

- **i-async-update** (*int*) : Steps before the network's weights are updated by the optimizer with the accumulated DeltaTheta
- **i-target** (*int*) : Steps before the target network's weights are updated
- **neural-network** (*NN Definition*) : Neural Network Architecture
- **experience-replay** (*ExperienceReplay*) : Experience replay
- **Policy** (*DiscreteDeepPolicy subclass*) : The policy
- **Output-Action** (*Action*) : The output action variable

LQRGain

- **Gain** (*double*) : The gain applied to the input state variable
- **Variable** (*State*) : The input state variable

LQRController

- **Output-Action** (*Action*) : The output action
- **LQR-Gain** (*Multiple LQRGain*) : An LQR gain on an input state variable

PIDController

- **KP** (*NumericValue subclass*) : Proportional gain
- **KI** (*NumericValue subclass*) : Integral gain
- **KD** (*NumericValue subclass*) : Derivative gain
- **Input-Variable** (*State*) : The input state variable

- **Output-Action** (*Action*) : The output action

WindTurbineVidalController

- **A** (*NumericValue subclass*) : A parameter of the torque controller
- **K_alpha** (*NumericValue subclass*) : K_alpha parameter of the torque controller
- **KP** (*NumericValue subclass*) : Proportional gain of the pitch controller
- **KI** (*NumericValue subclass*) : Integral gain of the pitch controller

WindTurbineBoukhezzarController

- **C_0** (*NumericValue subclass*) : C_0 parameter
- **KP** (*NumericValue subclass*) : Proportional gain of the pitch controller
- **KI** (*NumericValue subclass*) : Integral gain of the pitch controller

WindTurbineJonkmanController

- **CornerFreq** (*double*) : Corner Freq. parameter
- **VS_SlPc** (*double*) : SlPc parameter
- **VS_Rgn2K** (*double*) : Rgn2K parameter
- **VS_Rgn2Sp** (*double*) : Rgn2Sp parameter
- **VS_CtInSp** (*double*) : CtInSp parameter
- **VS_Rgn3MP** (*double*) : Rgn3MP parameter
- **PC_RefSpd** (*double*) : Pitch control reference speed
- **PC_KK** (*NumericValue subclass*) : Pitch angle were the the derivative of the...
- **PC_KP** (*NumericValue subclass*) : Proportional gain of the pitch controller
- **PC_KI** (*NumericValue subclass*) : Integral gain of the pitch controller

Controller-Factory

- **Controller** (*Controller subclass*) : The specific controller to be used

TDLambdaCritic

- **E-Traces** (*ETraces*) : Eligibility traces of the critic
- **Alpha** (*NumericValue subclass*) : Learning gain
- **V-Function** (*LinearStateVFA*) : The V-function to be learned

TDCLambdaCritic

- **E-Traces** (*ETraces*) : Elilgibility traces of the critic
- **Alpha** (*NumericValue subclass*) : Learning gain of the critic

- **Beta** (*NumericValue subclass*) : Learning gain applied to the omega vector
- **V-Function** (*LinearStateVFA*) : The V-function to be learned

TrueOnlineTDLambdaCritic

- **E-Traces** (*ETraces*) : Eligibility traces of the critic
- **Alpha** (*NumericValue subclass*) : Learning gain of the critic
- **V-Function** (*LinearStateVFA*) : The V-function to be learned

VLearnerCritic

- **V-Function** (*LinearStateVFA*) : The V-function to be learned

ICritic-Factory

- **Critic** (*ICritic subclass*) : Critic type

DDPG

- **Tau** (*double*) : The rate by which the target weights approach the online weights
- **Learning-Rate** (*double*) : The learning rate at which the agent learns
- **Critic-Network** (*NN Definition*) : Neural Network for the Critic -a Q function-
- **Actor-Network** (*NN Definition*) : Neural Network for the Actor -deterministic policy-
- **Exploration-Noise** (*Noise subclass*) : Noise added to the output of the policy
- **Input-State** (*Multiple State*) : Set of state variables used as input
- **Output-Action** (*Multiple Action*) : The output action variable

DiscreteEpsilonGreedyDeepPolicy

- **epsilon** (*NumericValue subclass*) : Epsilon

DiscreteSoftmaxDeepPolicy

- **temperature** (*NumericValue subclass*) : Tempreature

DiscreteDeepPolicy-Factory

- **Policy** (*DiscreteDeepPolicy subclass*) : The policy type

DQN

- **Num-Action-Steps** (*int*) : Number of discrete values used for the output action
- **Learning-Rate** (*double*) : The learning rate at which the agent learns
- **neural-network** (*NN Definition*) : Neural Network Architecture
- **Policy** (*DiscreteDeepPolicy subclass*) : The policy
- **Output-Action** (*Action*) : The output action variable

- **Input-State** (*Multiple State*) : Set of variables used as input of the QNetwork

DoubleDQN

- **Num-Action-Steps** (*int*) : Number of discrete values used for the output action
- **Learning-Rate** (*double*) : The learning rate at which the agent learns
- **neural-network** (*NN Definition*) : Neural Network Architecture
- **Policy** (*DiscreteDeepPolicy subclass*) : The policy
- **Output-Action** (*Action*) : The output action variable
- **Input-State** (*Multiple State*) : Set of variables used as input of the QNetwork

ETraces

- **Threshold** (*double*) : Threshold applied to trace factors
- **Lambda** (*double*) : Lambda parameter
- **Replace** (*bool*) : Replace existing traces? Or add?

ExperienceReplay

- **Buffer-Size** (*int*) : Size of the buffer used to store experience tuples
- **Update-Batch-Size** (*int*) : Number of tuples used each time-step in the update

Experiment

- **Random-Seed** (*int*) : Random seed used to generate random sequences of numbers
- **Num-Episodes** (*int*) : Number of episodes. Zero if we only want to run one evaluation episode
- **Eval-Freq** (*int*) : Evaluation frequency (in episodes). If zero then only training episodes will be run
- **Progress-Update-Freq** (*double*) : Progress update frequency (seconds)
- **Episode-Length** (*double*) : Length of an episode(seconds)

TileCodingFeatureMap

- **Num-Tiles** (*int*) : Number of tile layers of the grid
- **Tile-Offset** (*double*) : Offset of each tile relative to the previous one. It is scaled by the value range of the input variable

FeatureMap

- **Num-Features-Per-Dimension** (*int*) : Number of features per input variable

StateFeatureMap

- **Feature-Mapper** (*FeatureMapper subclass*) : The feature calculator used to map/unmap features
- **Input-State** (*Multiple State*) : State variables used as input of the feature map

- **Num-Features-Per-Dimension** (*int*) : Number of features per input variable

ActionFeatureMap

- **Feature-Mapper** (*FeatureMapper subclass*) : The feature calculator used to map/unmap features
- **Input-Action** (*Multiple Action*) : Action variables used as input of the feature map
- **Num-Features-Per-Dimension** (*int*) : Number of features per input variable

FeatureMapper-Factory

- **Type** (*FeatureMapper subclass*) : Feature map type

Logger

- **Num-Functions-Logged** (*int*) : How many times per experiment save logged functions
- **Log-Freq** (*double*) : Log frequency. Simulation time in seconds.
- **Log-Eval-Episodes** (*bool*) : Log evaluation episodes?
- **Log-Training-Episodes** (*bool*) : Log training episodes?
- **Log-Functions** (*bool*) : Log functions learned?

GaussianNoise

- **Sigma** (*double*) : Width of the gaussian bell
- **Alpha** (*double*) : Low-pass first-order filter's gain [0...1]. 1=no filter
- **Scale** (*NumericValue subclass*) : Scale factor applied to the noise signal before adding it to the policy's output

SinusoidalNoise

- **Time-Frequency** (*double*) : Frequency of the signal in 1/simulation seconds
- **Amplitude-Scale** (*NumericValue subclass*) : Scaling factor applied to the sinusoidal

OrnsteinUhlenbeckNoise

- **Mu** (*double*) : Mean value of the generated noise
- **Sigma** (*double*) : Degree of volatility around it caused by shocks
- **Theta** (*double*) : Rate by which noise shocks dissipate and the variable reverts towards the mean
- **Scale** (*NumericValue subclass*) : Scale factor applied to the noise signal before adding it to the policy's output

Noise-Factory

- **Noise** (*Noise subclass*) : Noise type

ConstantValue

- **Value** (*double*) : Constant value

SimpleEpisodeLinearSchedule

- **Initial-Value** (*double*) : Value at the beginning of the experiment
- **End-Value** (*double*) : Value at the end of the experiment

InterpolatedValue

- **Start-Offset** (*double*) : Normalized time from which the schedule will begin [0...1]
- **End-Offset** (*double*) : Normalized time at which the schedule will end and only return the End-Value [0...1]
- **Pre-Offset-Value** (*double*) : Output value before the schedule begins
- **Initial-Value** (*double*) : Output value at the beginning of the schedule
- **End-Value** (*double*) : Output value at the end of the schedule
- **Evaluation-Value** (*double*) : Output value during evaluation episodes
- **Interpolation** (*Interpolation*) : Interpolation type
- **Time-reference** (*TimeReference*) : The time-reference type

BhatnagarSchedule

- **Alpha-0** (*double*) : Alpha-0 parameter in Bhatnagar's schedule
- **Alpha-c** (*double*) : Alpha-c parameter in Bhatnagar's schedule
- **Time-Exponent** (*double*) : Time exponent in Bhatnagar's schedule
- **Evaluation-Value** (*double*) : Output value during evaluation episodes
- **Time-reference** (*TimeReference*) : The time reference

WireConnection

- **Wire** (*Wire*) : Wire connection from which the value comes

NumericValue-Factory

- **Schedule** (*NumericValue subclass*) : Schedule-type

PolicyLearner

- **Policy** (*Policy subclass*) : The policy to be learned

PolicyLearner-Factory

- **Policy-Learner** (*PolicyLearner subclass*) : The algorithm used to learn the policy

QEGreedyPolicy

- **Epsilon** (*NumericValue subclass*) : The epsilon parameter that balances exploitation and exploration

QSoftMaxPolicy

- **Tau** (*NumericValue subclass*) : Temperature parameter

GreedyQPlusNoisePolicy

- **Noise** (*Multiple Noise subclass*) : Noise signal added to the typical greedy action selection. The number of noise signals should match the number of actions in the action feature amp

QLearningCritic

- **Q-Function** (*LinearStateActionVFA*) : The parameterization of the Q-Function
- **E-Traces** (*ETraces*) : E-Traces
- **Alpha** (*NumericValue subclass*) : The learning gain [0-1]

QLearning

- **Policy** (*QPolicy subclass*) : The policy to be followed
- **Q-Function** (*LinearStateActionVFA*) : The parameterization of the Q-Function
- **E-Traces** (*ETraces*) : E-Traces
- **Alpha** (*NumericValue subclass*) : The learning gain [0-1]

DoubleQLearning

- **Policy** (*QPolicy subclass*) : The policy to be followed
- **Q-Function** (*LinearStateActionVFA*) : The parameterization of the Q-Function
- **E-Traces** (*ETraces*) : E-Traces
- **Alpha** (*NumericValue subclass*) : The learning gain [0-1]

SARSA

- **Policy** (*QPolicy subclass*) : The policy to be followed
- **Q-Function** (*LinearStateActionVFA*) : The parameterization of the Q-Function
- **E-Traces** (*ETraces*) : E-Traces
- **Alpha** (*NumericValue subclass*) : The learning gain [0-1]

QPolicy-Factory

- **Policy** (*QPolicy subclass*) : The exploration policy used to learn

SimGod

- **Target-Function-Update-Freq** (*int*) : Update frequency at which target functions will be updated. Only used if Freeze-Target-Function=true
- **Gamma** (*double*) : Gamma parameter
- **Freeze-Target-Function** (*bool*) : Defers updates on the V-functions to improve stability
- **Use-Importance-Weights** (*bool*) : Use sample importance weights to allow off-policy learning -

experimental-

- **State-Feature-Map** (*StateFeatureMap*) : The state feature map
- **Action-Feature-Map** (*ActionFeatureMap*) : The state feature map
- **Experience-Replay** (*ExperienceReplay*) : The experience replay parameters
- **Simion** (*Multiple Simion subclass*) : Simions: learning agents and controllers

Simion-Factory

- **Type** (*Simion subclass*) : The Simion class

Policy

- **Output-Action** (*Action*) : The output action variable

DeterministicPolicy

- **Output-Action** (*Action*) : The output action variable

StochasticPolicy

- **Output-Action** (*Action*) : The output action variable

DeterministicPolicyGaussianNoise

- **Deterministic-Policy-VFA** (*LinearStateVFA*) : The parameterized VFA that approximates the function
- **Exploration-Noise** (*Noise subclass*) : Parameters of the noise used as exploration
- **Output-Action** (*Action*) : The output action variable

StochasticGaussianPolicy

- **Mean-VFA** (*LinearStateVFA*) : The parameterized VFA that approximates the function
- **Sigma-VFA** (*LinearStateVFA*) : The parameterized VFA that approximates variance(s)
- **Output-Action** (*Action*) : The output action variable

Policy-Factory

- **Policy** (*Policy subclass*) : The policy type

LinearStateVFA

- **Init-Value** (*double*) : The initial value given to the weights on initialization

LinearStateActionVFA

- **Init-Value** (*double*) : The initial value given to the weights on initialization

World

- **Num-Integration-Steps** (*int*) : The number of integration steps performed each simulation time-step

- **Delta-T** (*double*) : The delta-time between simulation steps
- **Dynamic-Model** (*DynamicModel subclass*) : The dynamic model

DynamicModel-Factory

- **Model** (*DynamicModel subclass*) : The world

Worlds

BalancingPole

State variables

- x
- x_dot
- theta
- theta_dot

Action variables

- force

DoublePendulum

State variables

- theta_1
- theta_1-dot
- theta_2
- theta_2-dot

Action variables

- torque_1
- torque_2

FASTWindTurbine

State variables

- T_a
- P_a
- P_s
- P_e
- E_p
- v

- omega_r
- d_omega_r
- E_omega_r
- omega_g
- d_omega_g
- E_omega_g
- beta
- d_beta
- T_g
- d_T_g
- E_int_omega_r
- E_int_omega_g
- theta

Action variables

- beta
- T_g

MountainCar

State variables

- position
- velocity
- height
- angle

Action variables

- pedal

PitchControl

State variables

- setpoint-pitch
- attack-angle
- pitch
- pitch-rate
- control-deviation

Action variables

- pitch

PullBox1

State variables

- target-x
- target-y
- robot1-x
- robot1-y
- box-x
- box-y
- robot1-theta
- box-theta
- box-to-target-x
- box-to-target-y
- robot1-to-box-x
- robot1-to-box-y

Action variables

- robot1-v
- robot1-omega

PullBox2

State variables

- target-x
- target-y
- robot1-x
- robot1-y
- robot2-x
- robot2-y
- box-x
- box-y
- robot1-theta
- robot2-theta

- box-theta
- box-to-target-x
- box-to-target-y
- robot1-to-box-x
- robot1-to-box-y
- robot2-to-box-x
- robot2-to-box-y

Action variables

- robot1-v
- robot1-omega
- robot2-v
- robot2-omega

PushBox1

State variables

- target-x
- target-y
- robot1-x
- robot1-y
- box-x
- box-y
- robot1-to-box-x
- robot1-to-box-y
- box-to-target-x
- box-to-target-y
- robot1-theta
- box-theta

Action variables

- robot1-v
- robot1-omega

PushBox2

State variables

- target-x
- target-y
- robot1-x
- robot1-y
- robot2-x
- robot2-y
- box-x
- box-y
- robot1-to-box-x
- robot1-to-box-y
- robot2-to-box-x
- robot2-to-box-y
- box-to-target-x
- box-to-target-y
- robot1-theta
- robot2-theta
- box-theta

Action variables

- robot1-v
- robot1-omega
- robot2-v
- robot2-omega

RainCar

State variables

- position
- velocity
- position-deviation

Action variables

- acceleration

RobotControl

State variables

- target-x
- target-y
- robot1-x
- robot1-y
- robot1-theta

Action variables

- robot1-v
- robot1-omega

SwingupPendulum

State variables

- angle
- angular-velocity

Action variables

- torque

UnderwaterVehicle

State variables

- v-setpoint
- v
- v-deviation

Action variables

- u-thrust

WindTurbine

State variables

- T_a
- P_a
- P_s
- P_e
- E_p
- v
- omega_r

- d_omega_r
- E_omega_r
- omega_g
- d_omega_g
- E_omega_g
- beta
- d_beta
- T_g
- d_T_g
- E_int_omega_r
- E_int_omega_g
- theta

Action variables

- beta
- T_g