SMS_BP

Generated by Doxygen 1.11.0

Chapter 1

Documentation for the simulation configuration file of the same name

- Simulation Configuration File
- Latest version supported: v.0.1

1.1 Simulation Configuration File

- version: string
  - version of the simulation configuration file
- length_unit: string
  - length unit of the simulation (e.g. nm, um, mm)
- space_unit: string
  - space unit of the simulation (this is just pixel, should not change)
- time_unit: string
  - time unit of the simulation (e.g. s, ms, us)
- intensity_unit: string
  - intensity unit of the simulation (AUD only supported)
- diffusion_unit: string
  - diffusion unit of the simulation (e.g. um^2/s, mm^2/s)
- Cell_Parameters: dict
  - cell_space: 2D array (units of space_unit)
    1. cell_space[0]: x coordinates of the cell space (min, max)
    2. cell_space[1]: y coordinates of the cell space (min, max)
  - cell_axial_radius: float (units of space_unit)
    1. The distance from z=0 in either direction that the cell extends
  - number_of_cells: int

    1. number of cells to simulate (if more than 1 hen all are simulated in one folder defined by the output_path)

- Track_Parameters: dict

  - num_tracks: int
    1. number of tracks to simulate
  - track_type: string
    1. type of track to simulate ("fbm")
  - track_length_mean: int (frames)
    1. mean length of the track
  - track_distribution: string
    1. distribution of the track lengths ("exponential","constant")
  - diffusion_coefficient: list of floats (units of diffusion_unit)
    1. diffusion coefficient of the track, the length of the list is the unique type of diffusion coefficients
  - diffusion_track_amount: list of floats
    1. Only viable if allow_transition_probability is False
    2. length is the total number of diffusion coefficients
    3. each element is the probability of the track having the diffusion coefficient at the same index in the diffusion_coefficient list (add up to 1.0)
  - hurst_expontent: list of floats
    1. hurst exponent of the track, the length of the list is the unique type of hurst exponents
  - hurst_track_amount: list of floats
    1. Only viable if allow_transition_probability is False
    2. length is the total number of hurst exponents
    3. each element is the probability of the track having the hurst exponent at the same index in the hurst_exponent list (add up to 1.0)
  - allow_transition_probability: bool
    1. whether to allow transition probabilities between different diffusion coefficients and hurst exponents within a track
    2. if false, the track will have a single diffusion coefficient and hurst exponent
  - transition_matrix_time_step: int
    1. time step at which the diffusion and hurst exponent transition matrices are supplied in the following parameters
    2. the units are in time_unit (so 100 ms would be 100)
  - diffusion_transition_matrix: 2D array (discrete state probabilitiy at the transition_matrix_time_step = dt)
    1. transition matrix between different diffusion coefficients
    2. rows are the current diffusion coefficient
    3. columns are the next diffusion coefficient
    4. rows must sum to 1.0
  - hurst_transition_matrix: 2D array (discrete state probabilitiy at the transition_matrix_time_step = dt)
    1. transition matrix between different hurst exponents
    2. rows are the current hurst exponent
    3. columns are the next hurst exponent
    4. rows must sum to 1.0
  - state_probability_diffusion: 1D array (probability)
    1. probability of a track being in a certain diffusion coefficient state
    2. length is the number of unique diffusion coefficients

  - state_probability_hurst: 1D array (probability)
    1. probability of a track being in a certain hurst exponent state
    2. length is the number of unique hurst exponents

- Global_Parameters: dict

  - field_of_view_dim: 1D array (units of space_unit)
    1. field of view dimensions (x,y (pixels))
  - frame_count: int
    1. number of frames to simulate
  - exposure_time: float or int (units of time_unit)
    1. exposure time of the camera
  - interval_time: float or int (units of time_unit)
    1. time between frames that the camera is on
  - oversample_motion_time: float or int (units of time_unit)
    1. oversampling the motion for motion blur
    2. if oversample_motion_time == frame_time == exposure_time, then there is no motion blur
    3. cannot be greater than frame_time or exposure_time
  - pixel_size: float (units of length_unit)
    1. size of the pixel
  - axial_detection_range: float (units of length_unit)
    1. from z=0, the distance in either direction that the camera can detect a single molecule excitation
  - base_noise: float (units of intensity_unit)
    1. base noise of the camera (offset)
  - point_intensity: float (units of intensity_unit)
    1. intensity of a single molecule excitation
  - psf_sigma: float (units of length_unit)
    1. size of the psf (assumed to be gaussian)
  - axial_function: string ("exponential","ones"(no effect))
    1. function used to determine how the intensity of the single molecule changes with z

- Condenstate_Parameters: dict

  - initial_centers: 2D array (units of space_unit)
    1. initial centers of the condensates
    2. [x,y,z] coordinates per row
    3. number of rows is the number of condensates
  - initial_scale: 1D (units of space_unit)
    1. initial radius of the condensates
    2. number of elements is the number of condensates
    3. must be the same length as initial_centers
  - diffusion_coefficient: 1D array (units of diffusion_unit)
    1. diffusion coefficient of the condensates
    2. number of elements is the number of condensates
    3. must be the same length as initial_centers
  - hurst_exponent: 1D array
    1. hurst exponent of the condensates

    2. number of elements is the number of condensates
    3. must be the same length as initial_centers
  - density_dif: float
    1. density difference between the condensates and the rest of the cell

- Output_Parameters: dict

  - output_path: string
    1. path to save the output, directory
  - output_name: string
    1. name of the output file ∗subsegment_type: string
    1. function used to do projections ("mean","max","sum")
  - subsegment_number: int
    1. number of subsegments to divide the cell frames into
    2. if total movie is 500 frames and this is 5 then there will be 100 frames per subsegment and 5 subsegments in total
    3. Make sure that the total number of frames is divisible by the number of subsegments (modulus is 0)

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 SMS_BP Namespace Reference

Namespaces

- namespace boundary_conditions
- namespace condensate_movement
- namespace decorators
- namespace errors
- namespace fbm_BP
- namespace fbm_utility
- namespace probability_functions
- namespace run_cell_simulation
- namespace simulate_cell
- namespace simulate_foci

6.2 SMS_BP.boundary_conditions Namespace Reference

Functions

- _refecting_boundary (float fbm_store_last, float fbm_candidate, np.ndarray space_lim)
- _absorbing_boundary (float fbm_store_last, float fbm_candidate, np.ndarray space_lim)

6.2.1 Function Documentation

6.2.1.1 _absorbing_boundary()

```
SMS_BP.boundary_conditions._absorbing_boundary (
            float fbm_store_last,
            float fbm_candidate,
            np.ndarray space_lim)  [protected]
```

```
Absorbing boundary condition for the FBM 1D

Parameters:
-----------
fbm_store_last : float
    Last value of the FBM
fbm_candidate : float
    Candidate value of the FBM
space_lim : np.ndarray
    Space limit (min, max) for the FBM

Returns:
--------
float
    New value of the FBM
```

6.2.1.2 _refecting_boundary()

```
SMS_BP.boundary_conditions._refecting_boundary (
            float fbm_store_last,
            float fbm_candidate,
            np.ndarray space_lim)  [protected]

Reflecting boundary condition for the FBM 1D

Parameters:
-----------
fbm_store_last : float
    Last value of the FBM
fbm_candidate : float
    Candidate value of the FBM
space_lim : np.ndarray
    Space limit (min, max) for the FBM

Returns:
--------
float
    New value of the FBM
```

6.3 SMS_BP.condensate_movement Namespace Reference

Classes

- class Condensate

6.3.1 Detailed Description

```
Contains class for storing condensate data. Condensates are defined as spherical always; defined by a
center (x,y,z), radius (r), and time (t). The complete description of the condensate at any time (t) is:
(x,y,z,r,t).

Usage:
------
    Initialize the class as follows:
        condensate = Condensate(**{
            "inital_position":np.array([0, 0, 0]),
            "initial_time":0,
            "diffusion_coefficient":0,
            "hurst_exponent":0,
            "units_time":'ms',
            "units_position":'um',
            "condensate_id":0,
            "initial_scale":0,
        })
    Call the class object as follows to get the position and scale of the condensate at a given time:
        condensate(times, time_unit) -> dict{"Position":np.ndarray, "Scale":float}
```

6.4 SMS_BP.decorators Namespace Reference

Classes

- class CountCalls

Functions

- deprecated (reason)
- timer (func)
- debug (func)
- slow_down (_func=None, *, rate=1)
- repeat (_func=None, *, num_times=2)
- singleton (cls)
- cache (func)
- set_unit (unit)
- _catch_recursion_error (func)

Variables

- tuple string_types = (type(b''), type(u''))

6.4.1 Function Documentation

6.4.1.1 _catch_recursion_error()

```
SMS_BP.decorators._catch_recursion_error (
               func)  [protected]
```

6.4.1.2 cache()

```
SMS_BP.decorators.cache (
               func)
```

Keep a cache of previous function calls

6.4.1.3 debug()

```
SMS_BP.decorators.debug (
               func)
```

Print the function signature and return value

6.4.1.4 deprecated()

```
SMS_BP.decorators.deprecated (
              reason)
```

```
This is a decorator which can be used to mark functions
as deprecated. It will result in a warning being emitted
when the function is used.
```

6.4.1.5 repeat()

```
SMS_BP.decorators.repeat (
              _func = None,
             * ,
              num_times = 2)
```

```
Repeat the function a number of times

Parameters:
-----------
num_times : int
    number of times to repeat the function

Returns:
--------
decorator_repeat : function
    decorator function
```

6.4.1.6 set_unit()

```
SMS_BP.decorators.set_unit (
              unit)
```

```
Register a unit on a function
```

6.4.1.7 singleton()

```
SMS_BP.decorators.singleton (
              cls)
```

```
Make a class a Singleton class (only one instance)
```

6.4.1.8 slow_down()

```
SMS_BP.decorators.slow_down (
              _func = None,
             * ,
              rate = 1)
```

```
Sleep given amount of seconds before calling the function
```

6.4.1.9 timer()

```
SMS_BP.decorators.timer (
              func)
```

Print the runtime of the decorated function

6.4.2 Variable Documentation

6.4.2.1 string_types

```
tuple SMS_BP.decorators.string_types = (type(b''), type(u''))
```

6.5 SMS_BP.errors Namespace Reference

Classes

- class DiffusionHighError
- class HurstHighError
- class HurstValueError
- class SpaceLimitError

6.6 SMS_BP.fbm_BP Namespace Reference

Classes

- class FBM_BP

Functions

- MCMC_state_selection (int initial_state_index, np.ndarray transition_matrix, np.ndarray possible_states, int n)
- _boundary_conditions (float fbm_store_last, float fbm_candidate, np.ndarray space_lim, str condition_type)

Variables

- dict BOUNDARY_CONDITIONS
- transition_matrix = np.array([[0.4, 0.6], [0.2, 0.8]])
- possible_states = np.array([1, 2])
- int n = 50000
- int initial_state_index = 1
- state_select
- state_probability = np.zeros(len(possible_states))
- total_rate = np.sum(transition_matrix)
- true_state_probability = np.sum(transition_matrix, axis=0)/total_rate
- label
- alpha
- int state_1_to_2 = np.zeros(n) - 1
- int state_2_to_1 = np.zeros(n) - 1

6.6.1 Function Documentation

6.6.1.1 _boundary_conditions()

```
SMS_BP.fbm_BP._boundary_conditions (
            float fbm_store_last,
            float fbm_candidate,
            np.ndarray space_lim,
            str condition_type)  [protected]
```

```
Boundary conditions for the FBM

Parameters:
-----------
fbm_store_last : float
    Last value of the FBM
fbm_candidate : float
    Candidate value of the FBM
space_lim : np.ndarray
    Space limit (min, max) for the FBM\
condition_type : str
    Type of boundary condition takes values in REFLECTING_CONDITIONS
Returns:
--------
float
    New value of the FBM
```

6.6.1.2 MCMC_state_selection()

```
SMS_BP.fbm_BP.MCMC_state_selection (
            int initial_state_index,
            np.ndarray transition_matrix,
            np.ndarray possible_states,
            int n)
```

```
Markov Chain Monte Carlo state selection

Parameters:
-----------
initial_state_index : int
    Initial state index, this is the index of the initial state in the possible states
transition_matrix : np.ndarray
    Transition matrix, this is the prbability at a time step. (time step is 1)
possible_states : np.ndarray
    possible states
n : int
    Number of iterations

Returns:
--------
np.ndarray
    State selection at each iteration
```

6.6.2 Variable Documentation

6.6.2.1 alpha

```
SMS_BP.fbm_BP.alpha
```

6.6.2.2 BOUNDARY_CONDITIONS

```
dict SMS_BP.fbm_BP.BOUNDARY_CONDITIONS
```

Initial value:

```
00001 =  {
00002     'reflecting': _refecting_boundary,
00003     'absorbing': _absorbing_boundary
00004 }
```

6.6.2.3 initial_state_index

```
int SMS_BP.fbm_BP.initial_state_index = 1
```

6.6.2.4 label

```
SMS_BP.fbm_BP.label
```

6.6.2.5 n

```
int SMS_BP.fbm_BP.n = 50000
```

6.6.2.6 possible_states

```
SMS_BP.fbm_BP.possible_states = np.array([1, 2])
```

6.6.2.7 state_1_to_2

```
int SMS_BP.fbm_BP.state_1_to_2 = np.zeros(n) - 1
```

6.6.2.8 state_2_to_1

```
int SMS_BP.fbm_BP.state_2_to_1 = np.zeros(n) - 1
```

6.6.2.9 state_probability

```
SMS_BP.fbm_BP.state_probability = np.zeros(len(possible_states))
```

6.6.2.10 state_select

```
SMS_BP.fbm_BP.state_select
```

Initial value:

```
00001 =  MCMC_state_selection(
00002         initial_state_index, transition_matrix, possible_states, n)
```

6.6.2.11 total_rate

```
SMS_BP.fbm_BP.total_rate = np.sum(transition_matrix)
```

6.6.2.12 transition_matrix

```
SMS_BP.fbm_BP.transition_matrix = np.array([[0.4, 0.6], [0.2, 0.8]])
```

6.6.2.13 true_state_probability

```
SMS_BP.fbm_BP.true_state_probability = np.sum(transition_matrix, axis=0)/total_rate
```

6.7 SMS_BP.fbm_utility Namespace Reference

Functions

- get_fbm_sample (l=1, h=0.5, d=1, n=1)

6.7.1 Detailed Description

```
This module contains functions for generating fractional brownian motion samples and computing the mean square

Functions:
----------
get_fbm_sample(l=1,h=0.5,d=1,n=1)
    Generates a sample of fractional brownian motion
compute_msd_np(xy, t, t_step)
    Computes the mean squared displacement for a given sample

Author: Baljyot Singh Parmar
```

6.7.2 Function Documentation

6.7.2.1 get_fbm_sample()

```
SMS_BP.fbm_utility.get_fbm_sample (
              l = 1,
              h = 0.5,
              d = 1,
              n = 1)
```

```
Generates a sample of fractional brownian motion
Theory: https://en.wikipedia.org/wiki/Fractional_Brownian_motion
Implementation is using the fbm package: https://pypi.org/project/fbm/
Default values are for testing purposes only


Parameters:
-----------
l : float,int
    end time (from 0)
h : float,int  (0 < h < 1)
    hurst parameter, must be between 0 and 1
d : int
    dimensions (x,y,z .... ) for one realization, must be greater than 0
n : int
    even intervals from 0,l, must be greater than 0

Returns:
--------
list of lists of numpy arrays, where the first list is the time values for each sample, and the second list is

Raises:
-------
TypeError
    If any of the parameters are not of the correct type
ValueError
    If any of the parameters are not of the correct value

Notes:
------
1. The number of samples is equal to the number of dimensions
```

6.8 SMS_BP.probability_functions Namespace Reference

Classes

- class multiple_top_hat_probability

Functions

- test_multiple_top_hat_probability ()

6.8.1 Function Documentation

6.8.1.1 test_multiple_top_hat_probability()

```
SMS_BP.probability_functions.test_multiple_top_hat_probability ()
```

6.9 SMS_BP.run_cell_simulation Namespace Reference

Functions

- main_CLI ()
- main_noCLI (file)

Variables

- project_directory
- config_file

6.9.1 Function Documentation

6.9.1.1 main_CLI()

```
SMS_BP.run_cell_simulation.main_CLI ()
```

```
CLI tool to run cell simulation.

Usage:
    python run_cell_simulation.py <config_file> [--output_path <output_path>]

Arguments:
    config_file     Path to the configuration file

Options:
    --output_path   Path to the output directory
```

6.9.1.2 main_noCLI()

```
SMS_BP.run_cell_simulation.main_noCLI (
              file)
```

```
Run cell simulation without using CLI arguments
```

6.9.2 Variable Documentation

6.9.2.1 config_file

```
SMS_BP.run_cell_simulation.config_file
```

Initial value:

```
00001 =  os.path.join(
00002             project_directory, "SMS_BP", "sim_config.json")
```

6.9.2.2 project_directory

```
SMS_BP.run_cell_simulation.project_directory
```

Initial value:

```
00001 =  os.path.dirname(
00002             os.path.dirname(os.path.abspath(__file__)))
```

6.10 SMS_BP.simulate_cell Namespace Reference

Classes

- class Simulate_cells

Functions

- save_tiff (image, path, img_name=None)
- sub_segment (img, sub_frame_num, img_name=None, subsegment_type="mean")
- make_directory_structure (cd, img_name, img, subsegment_type, sub_frame_num, ∗∗kwargs)
- convert_lists_to_arrays (obj)
- convert_arrays_to_lists (obj)

Variables

- sim_new
- cd
- img_name
- subsegment_type
- sub_frame_num

6.10.1 Function Documentation

6.10.1.1 convert_arrays_to_lists()

```
SMS_BP.simulate_cell.convert_arrays_to_lists (
              obj)
```

6.10.1.2 convert_lists_to_arrays()

```
SMS_BP.simulate_cell.convert_lists_to_arrays (
              obj)
```

6.10.1.3 make_directory_structure()

```
SMS_BP.simulate_cell.make_directory_structure (
              cd,
              img_name,
              img,
              subsegment_type,
              sub_frame_num,
             ** kwargs)
```

```
 Docstring for make_directory_structure: make the directory structure for the simulation and save the image +
Also perform the subsegmentation and save the subsegments in the appropriate directory
Parameters:
-----------
cd : str
    directory to save the simulation
img_name : str
    name of the image
img : array-like
    image to be subsegmented
subsegment_type : str
    type of subsegmentation to be performed, currently only "mean" is supported
sub_frame_num : int
    number of subsegments to be created
**kwargs : dict
    dictionary of keyword arguments
KWARGS:
-------
data : dict, Default = None
    dictionary of data to be saved, Keys = "map","tracks","points_per_frame" Values = array-like.
    See the return of the function simulate_cell_tracks for more details
parameters : dict, Default = self.init_dict
Returns:
--------
array-like
    list of subsegment images
```

6.10.1.4 save_tiff()

```
SMS_BP.simulate_cell.save_tiff (
              image,
              path,
              img_name = None)
```

```
 Docstring for save_tiff: save the image as a tiff file
Parameters:
-----------
image : array-like
    image to be saved
path : str
    path to save the image
img_name : str, Default = None
    name of the image
Returns:
--------
None
```

6.10.1.5 sub_segment()

```
SMS_BP.simulate_cell.sub_segment (
              img,
              sub_frame_num,
              img_name = None,
              subsegment_type = "mean")
```

```
 Docstring for sub_segment: perform subsegmentation on the image
Parameters:
-----------
img : array-like
    image to be subsegmented
sub_frame_num : int
```

```
    number of subsegments to be created
img_name : str, Default = None
    name of the image
subsegment_type : str, Default = "mean"
    type of subsegmentation to be performed, currently only "mean" is supported
Returns:
--------
hold_img : list
    list of subsegments
hold_name : list
    list of names of the subsegments
```

6.10.2 Variable Documentation

6.10.2.1 cd

```
SMS_BP.simulate_cell.cd
```

6.10.2.2 img_name

```
SMS_BP.simulate_cell.img_name
```

6.10.2.3 sim_new

```
SMS_BP.simulate_cell.sim_new
```

Initial value:

```
00001 =  Simulate_cells(
00002
      init_dict_json="/Users/baljyot/Documents/CODE/GitHub_t2/PHD/SingleMoleculeSimulations_BP/SMS_BP/sim_config_testing.json"
```

6.10.2.4 sub_frame_num

```
SMS_BP.simulate_cell.sub_frame_num
```

6.10.2.5 subsegment_type

```
SMS_BP.simulate_cell.subsegment_type
```

6.11 SMS_BP.simulate_foci Namespace Reference

Classes

- class Track_generator

Functions

- get_lengths (str track_distribution, int track_length_mean, int total_tracks)
- dict create_condensate_dict (np.ndarray initial_centers, np.ndarray initial_scale, np.ndarray diffusion_↩ coefficient, np.ndarray hurst_exponent, np.ndarray cell_space, float cell_axial_range, ∗∗kwargs)
- tophat_function_2d (var, center, radius, bias_subspace, space_prob, ∗∗kwargs)
- generate_points (pdf, total_points, min_x, max_x, center, radius, bias_subspace_x, space_prob, density_dif)
- generate_points_from_cls (pdf, total_points, min_x, max_x, min_y, max_y, min_z, max_z, density_dif)
- generate_radial_points (total_points, center, radius)
- generate_sphere_points (total_points, center, radius)
- radius_spherical_cap (R, center, z_slice)
- get_gaussian (mu, sigma, domain=[list(range(10)), list(range(10))])
- float|np.ndarray axial_intensity_factor (float|np.ndarray abs_axial_pos, float detection_range, ∗∗kwargs)
- np.ndarray generate_map_from_points (np.ndarray points, float|np.ndarray point_intensity, np.ndarray|None map, bool movie, float base_noise, float psf_sigma)

6.11.1 Detailed Description

```
Documentation for the simulate_foci.py file.
This file contains the class for simulating foci in space.

Author: Baljyot Singh Parmar
```

6.11.2 Function Documentation

6.11.2.1 axial_intensity_factor()

```
float | np.ndarray SMS_BP.simulate_foci.axial_intensity_factor (
            float | np.ndarray abs_axial_pos,
            float detection_range,
            ** kwargs)
```

```
Docstring
Calculate the factor for the axial intensity of the PSF given the absolute axial position from the 0 position
the focal plane. This is the factor that is multiplied by the intensity of the PSF

For now this is a negative exponential decay i.e:
    I = I_0*e^(-|z-z_0|)
This function returns the factor e^(-|z-z_0|**2 / (2*2.2**2)) only.

Parameters:
-----------
abs_axial_pos : float|np.ndarray
    absolute axial position from the 0 position of the focal plane
detection_range : float
    detection range of the function. This is the standard deviation of the gaussian function describing the ax
kwargs : dict

Returns:
--------
float|np.ndarray
    factor for the axial intensity of the PSF
```

6.11.2.2 create_condensate_dict()

```
dict SMS_BP.simulate_foci.create_condensate_dict (
            np.ndarray initial_centers,
            np.ndarray initial_scale,
            np.ndarray diffusion_coefficient,
            np.ndarray hurst_exponent,
            np.ndarray cell_space,
            float cell_axial_range,
            ** kwargs)

Docstring for create_condensate_dict:

Parameters:
-----------
inital_centers: numpy array of shape (num_condensates,2) with the initial centers of the condensates
initial_scale: numpy array of shape (num_condensates,2) with the initial scales of the condensates
diffusion_coefficient: numpy array of shape (num_condensates,2) with the diffusion coefficients of the condens
hurst_exponent: numpy array of shape (num_condensates,2) with the hurst exponents of the condensates
cell_space: numpy array of shape (2,2) with the cell space
cell_axial_range: float
**kwargs: additional arguments to be passed to the condensate_movement.Condensate class
```

6.11.2.3 generate_map_from_points()

```
np.ndarray SMS_BP.simulate_foci.generate_map_from_points (
            np.ndarray points,
            float | np.ndarray point_intensity,
            np.ndarray | None map,
            bool movie,
            float base_noise,
            float psf_sigma)

Docstring for generate_map_from_points:
---------------------------
Generates the space map from the points. 2D

Parameters:
-----------
points: array-like
    points numpy array of shape (total_points,2)
point_intensity: array-like
    intensity of the points, if None, then self.point_intensity is used.
map: array-like
    space map, if None, then a new space map is generated.
movie: bool
    if True, then don't add the gaussian+noise for each point. Rather add the gaussians and then to the whole
base_noise: float
    base noise to add to the space map
psf_sigma: float
    sigma of the psf (pix units)


Returns:
--------
1. space map as a numpy array of shape (max_x,max_x)
2. points as a numpy array of shape (total_points,2)


Notes:
------
1. The space map is generated using get_gaussian function.
2. For movie: In the segmented experimental images you are adding the noise of each frame to the whole subfram
    so for this (movie=False) add each gaussian point to the image with the noise per point.
    (movie=True) add the gaussians together and then add the noise to the final image.
```

6.11.2.4 generate_points()

```
SMS_BP.simulate_foci.generate_points (
              pdf,
              total_points,
              min_x,
              max_x,
              center,
              radius,
              bias_subspace_x,
              space_prob,
              density_dif)
```

```
genereates random array of (x,y) points given a distribution using accept/reject method

Parameters
----------
pdf : function
    function which defines the distribution to sample from
total_points : int
    total points to sample
min_x : float
    lower bound to the support of the distribution
max_x : float
    upper bound to the support of the distribution
center : array-like of float
    coordinates of the center of the top hat
redius : float
    raidus of the top hat
bias_subspace : float
    probability at the top hat
space_prob : float
    probaility everywhere not at the top hat

Returns
-------
array-like
    [x,y] coordinates of the points sampled from the distribution defined in pdf
```

6.11.2.5 generate_points_from_cls()

```
SMS_BP.simulate_foci.generate_points_from_cls (
              pdf,
              total_points,
              min_x,
              max_x,
              min_y,
              max_y,
              min_z,
              max_z,
              density_dif)
```

6.11.2.6 generate_radial_points()

```
SMS_BP.simulate_foci.generate_radial_points (
              total_points,
              center,
              radius)
```

```
Genereate uniformly distributed points in a circle of radius.

Parameters
----------
total_points : int
    total points from this distribution
center : array-like or tuple
    coordinate of the center of the radius. [x,y,...]
radius : float-like
    radius of the region on which to

Returns
-------
(n,2) size array
    array of coordinates of points genereated (N,3) N = # of points, 2 = dimentions
```

6.11.2.7 generate_sphere_points()

```
SMS_BP.simulate_foci.generate_sphere_points (
              total_points,
              center,
              radius)

Genereate uniformly distributed points in a sphere of radius.

Parameters
----------
total_points : int
    total points from this distribution
center : array-like or tuple
    coordinate of the center of the radius. [x,y,...]
radius : float-like
    radius of the region on which to

Returns
-------
(n,2) size array
    array of coordinates of points genereated (N,3) N = # of points, 2 = dimentions
```

6.11.2.8 get_gaussian()

```
SMS_BP.simulate_foci.get_gaussian (
              mu,
              sigma,
              domain = [list(range(10)), list(range(10))])

Parameters
----------
mu : array-like or float of floats
    center position of gaussian (x,y) or collection of (x,y)
sigma : float or array-like of floats of shape mu
    sigma of the gaussian
domain : array-like, Defaults to 0->9 for x,y
    x,y domain over which this gassuain is over


Returns
-------
array-like 2D
    values of the gaussian centered at mu with sigma across the (x,y) points defined in domain

Notes:
------
THIS IS IMPORTANT: MAKE SURE THE TYPES IN EACH PARAMETER ARE THE SAME!!!!
```

6.11.2.9 get_lengths()

```
SMS_BP.simulate_foci.get_lengths (
             str track_distribution,
             int track_length_mean,
             int total_tracks)
```

```
Returns the track lengths from the distribution track_distribution. The lengths are returned as the closest in

Parameters:
-----------
track_distribution: distribution of track lengths. Options are "exponential" and "uniform"
track_length_mean: mean track length
total_tracks: total number of tracks to be generated

Returns:
--------
track lengths as a numpy array of shape (total_tracks,1)

Notes:
------
1. If the distribution is exponential, then the track lengths are generated using exponential distribution.
2. If the distribution is uniform, then the track lengths are generated using uniform distribution between 0 a
3. If the distribution is constant, then all the track lengths are set to the mean track length. (track_length

Exceptions:
-----------
ValueError: if the distribution is not recognized.
```

6.11.2.10 radius_spherical_cap()

```
SMS_BP.simulate_foci.radius_spherical_cap (
              R,
              center,
              z_slice)
```

```
 Find the radius of a spherical cap given the radius of the sphere and the z coordinate of the slice
Theory: https://en.wikipedia.org/wiki/Spherical_cap, https://mathworld.wolfram.com/SphericalCap.html

Parameters:
-----------
R : float,int
    radius of the sphere
center : array-like
    [x,y,z] coordinates of the center of the sphere
z_slice : float,int
    z coordinate of the slice relative to the center of the sphere, z_slice = 0 is the center of the sphere

Returns:
--------
float
    radius of the spherical cap at the z_slice

Notes:
------
1. This is a special case of the spherical cap equation where the center of the sphere is at the origin
```

6.11.2.11 tophat_function_2d()

```
SMS_BP.simulate_foci.tophat_function_2d (
              var,
              center,
              radius,
              bias_subspace,
              space_prob,
             ** kwargs)
```

```
Defines a circular top hat probability distribution with a single biased region defining the hat.
The rest of the space is uniformly distrubuted in 2D

Parameters
----------
var : array-like, float
    [x,y] defining sampling on the x,y span of this distribution
center : array-like, float
    [c1,c2] defining the center coordinates of the top hat region
radius : float
    defines the radius of the circular tophat from the center
bias_subspace : float
    probability at the top position of the top hat
space_prob : float
    probability everywhere not in the bias_subspace

Returns
-------
float, can be array-like if var[0],var[1] is array-like
    returns the value of bias_subspace or space_prob depending on where the [x,y] data lies
```

Chapter 7

Class Documentation

7.1 SMS_BP.condensate_movement.Condensate Class Reference

Public Member Functions

- __init__ (self, np.ndarray inital_position=np.array([0, 0, 0]), int initial_time=0, float diffusion_coefficient=0, float hurst_exponent=0, str units_time='ms', str units_position='um', int condensate_id=0, float initial_scale=0, np.ndarray cell_space=np.array([[0, 0], [0, 0], [0, 0]]), float|int cell_axial_range=0)
- np.ndarray times (self)
- None times (self, value)
- np.ndarray condensate_positions (self)
- None condensate_positions (self, value)
- np.ndarray scale (self)
- None scale (self, value)
- None add_positions (self, np.ndarray time, np.ndarray position, np.ndarray scale)
- list __call__ (self, int time, str time_unit)
- None generate_condensate_positions (self, int time)
- np.ndarray calculate_scale (self, np.ndarray time, np.ndarray position)
- plt.Axes plot_condensate (self, plt.Axes ax, **kwargs)

Public Attributes

- initial_position
- initial_time
- diffusion_coefficient
- hurst_exponent
- units_time
- units_position
- condensate_id
- initial_scale
- dim
- cell_space
- cell_axial_range
- times
- condensate_positions
- scale

Protected Member Functions

- None _generate_condensate_positions (self, int time)

Protected Attributes

- _times
- _condensate_positions
- _scale

7.1.1 Detailed Description

```
 Condensate class for storing condensate data.

Parameters:
-----------
inital_position: np.ndarray = np.array([0, 0, 0])
    Initial position of the condensate.
initial_time: float = 0
    Initial time of the condensates.
diffusion_coefficient: float = 0
    Diffusion coefficient of the condensate.
hurst_exponent: float = 0
    Hurst exponent of the condensate.
units_time: str = 's'
    Units of time. Units work as follows: in the class reference frame, start from 0 and iterate by 1 each tim
    For a units_time of "ms", 1 represents 1ms.
    For a units_time of "s", 1 represents 1s.
    For a units_time of "20ms", 1 represents 20ms.
units_position: str = 'um'
    Units of position.
condensate_id: int = 0
    ID of the condensate.
initial_scale: float = 0
    Initial scale of the condensate.
cell_space: np.ndarray = np.array([[0,0],[0,0],[0,0]])
    Space of the cell.
cell_axial_range: float|int = 0
    Axial range of the cell.
```

7.1.2 Constructor & Destructor Documentation

7.1.2.1 __init__()

```
SMS_BP.condensate_movement.Condensate.__init__ (
             self,
            np.ndarray  inital_position = np.array([0, 0, 0]),
            int  initial_time = 0,
            float  diffusion_coefficient = 0,
            float  hurst_exponent = 0,
            str  units_time = 'ms',
            str  units_position = 'um',
            int  condensate_id = 0,
            float  initial_scale = 0,
            np.ndarray  cell_space = np.array([[0, 0], [0, 0], [0, 0]]),
            float | int  cell_axial_range = 0)
```

7.1.3 Member Function Documentation

7.1.3.1 __call__()

```
list SMS_BP.condensate_movement.Condensate.__call__ (
             self,
            int time,
            str time_unit)
```

Returns the position and scale of the condensate at a given time.

```
Parameters:
-----------
time: float
    Time at which to return the position of the condensate. User needs to convert to the reference frame of th
time_unit: str
    Units of time.
    Just to make sure the user is aware of the conversion they need to do to get into the reference frame of t

Returns:
--------
Dict of the position and scale of the condensate at the given time.
    Keys:
        Position: np.ndarray
            Position of the condensate at the given time.
        Scale: float
            Scale of the condensate at the given time.
```

7.1.3.2 _generate_condensate_positions()

```
None SMS_BP.condensate_movement.Condensate._generate_condensate_positions (
             self,
            int time)  [protected]
```

Generates the condensate positions up to a given time.

```
Parameters:
-----------
time: int
    Time up to which to generate the condensate positions.
```

7.1.3.3 add_positions()

```
None SMS_BP.condensate_movement.Condensate.add_positions (
             self,
            np.ndarray time,
            np.ndarray position,
            np.ndarray scale)
```

Adds positions to the condensate.

```
Parameters:
-----------
time: np.ndarray
    Times at which to add positions.
position: np.ndarray
    Positions to add to the condensate.
scale: np.ndarray
    Scale to add to the condensate.
```

7.1.3.4 calculate_scale()

```
np.ndarray SMS_BP.condensate_movement.Condensate.calculate_scale (
              self,
             np.ndarray time,
             np.ndarray position)

Calculates the scale of the condensate at a given time.

Parameters:
-----------
time: np.ndarray
    Times at which to calculate the scale.
position: np.ndarray
    Positions at which to calculate the scale.
```

7.1.3.5 condensate_positions() [1/2]

```
np.ndarray SMS_BP.condensate_movement.Condensate.condensate_positions (
              self)

 Returns the positions of the condensate.
```

7.1.3.6 condensate_positions() [2/2]

```
None SMS_BP.condensate_movement.Condensate.condensate_positions (
              self,
              value)
```

7.1.3.7 generate_condensate_positions()

```
None SMS_BP.condensate_movement.Condensate.generate_condensate_positions (
              self,
             int time)

 Generates the condensate positions up to a given time.

Parameters:
-----------
time: int
    Time up to which to generate the condensate positions.
```

7.1.3.8 plot_condensate()

```
plt.Axes SMS_BP.condensate_movement.Condensate.plot_condensate (
              self,
             plt.Axes ax,
             ** kwargs)

Plots the condensate

Parameters:
-----------
ax: plt.Axes
    Axes to plot the condensate on.
**kwargs:
    Keyword arguments to pass to the plot function.
```

7.1.3.9 scale() [1/2]

```
np.ndarray SMS_BP.condensate_movement.Condensate.scale (
              self)
```

Returns the scale of the condensate.

7.1.3.10 scale() [2/2]

```
None SMS_BP.condensate_movement.Condensate.scale (
              self,
              value)
```

7.1.3.11 times() [1/2]

```
np.ndarray SMS_BP.condensate_movement.Condensate.times (
              self)
```

Returns the times of the condensate.

7.1.3.12 times() [2/2]

```
None SMS_BP.condensate_movement.Condensate.times (
              self,
              value)
```

7.1.4 Member Data Documentation

7.1.4.1 _condensate_positions

```
SMS_BP.condensate_movement.Condensate._condensate_positions  [protected]
```

7.1.4.2 _scale

```
SMS_BP.condensate_movement.Condensate._scale  [protected]
```

7.1.4.3 _times

```
SMS_BP.condensate_movement.Condensate._times  [protected]
```

7.1.4.4 cell_axial_range

```
SMS_BP.condensate_movement.Condensate.cell_axial_range
```

7.1.4.5 cell_space

SMS_BP.condensate_movement.Condensate.cell_space

7.1.4.6 condensate_id

SMS_BP.condensate_movement.Condensate.condensate_id

7.1.4.7 condensate_positions

SMS_BP.condensate_movement.Condensate.condensate_positions

7.1.4.8 diffusion_coefficient

SMS_BP.condensate_movement.Condensate.diffusion_coefficient

7.1.4.9 dim

SMS_BP.condensate_movement.Condensate.dim

7.1.4.10 hurst_exponent

SMS_BP.condensate_movement.Condensate.hurst_exponent

7.1.4.11 initial_position

SMS_BP.condensate_movement.Condensate.initial_position

7.1.4.12 initial_scale

SMS_BP.condensate_movement.Condensate.initial_scale

7.1.4.13 initial_time

SMS_BP.condensate_movement.Condensate.initial_time

7.1.4.14 scale

SMS_BP.condensate_movement.Condensate.scale

7.1.4.15 times

```
SMS_BP.condensate_movement.Condensate.times
```

7.1.4.16 units_position

```
SMS_BP.condensate_movement.Condensate.units_position
```

7.1.4.17 units_time

```
SMS_BP.condensate_movement.Condensate.units_time
```

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

- condensate_movement.py

7.2 SMS_BP.decorators.CountCalls Class Reference

Public Member Functions

- __init__ (self, func)
- __call__ (self, *args, **kwargs)

Public Attributes

- func
- num_calls

7.2.1 Detailed Description

```
Count how many times a function is called
```

7.2.2 Constructor & Destructor Documentation

7.2.2.1 __init__()

```
SMS_BP.decorators.CountCalls.__init__ (
              self,
              func)
```

7.2.3 Member Function Documentation

7.2.3.1 __call__()

```
SMS_BP.decorators.CountCalls.__call__ (
              self,
             * args,
             ** kwargs)
```

7.2.4 Member Data Documentation

7.2.4.1 func

```
SMS_BP.decorators.CountCalls.func
```

7.2.4.2 num_calls

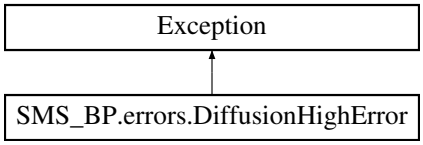
```
SMS_BP.decorators.CountCalls.num_calls
```

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

- decorators.py

7.3 SMS_BP.errors.DiffusionHighError Class Reference

Inheritance diagram for SMS_BP.errors.DiffusionHighError:



7.3.1 Detailed Description

```
Raised when the diffusion value is too high for the space limit
```

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

- errors.py

7.4 SMS_BP.fbm_BP.FBM_BP Class Reference

Public Member Functions

- __init__ (self, int n, float dt, np.ndarray diffusion_parameters, np.ndarray hurst_parameters, np.↩ ndarray diffusion_parameter_transition_matrix, np.ndarray hurst_parameter_transition_matrix, np.ndarray state_probability_diffusion, np.ndarray state_probability_hurst, np.ndarray space_lim)
- fbm (self)

Public Attributes

- n
- dt
- diffusion_parameter
- hurst_parameter
- diffusion_parameter_transition_matrix
- hurst_parameter_transition_matrix
- state_probability_diffusion
- state_probability_hurst
- space_lim

Protected Member Functions

- _autocovariance (self, k, hurst)
- None _setup (self)

Protected Attributes

- _cov
- _diff_a_n
- _hurst_n

7.4.1 Constructor & Destructor Documentation

7.4.1.1 __init__()

```
SMS_BP.fbm_BP.FBM_BP.__init__ (
              self,
             int n,
             float dt,
             np.ndarray diffusion_parameters,
             np.ndarray hurst_parameters,
             np.ndarray diffusion_parameter_transition_matrix,
             np.ndarray hurst_parameter_transition_matrix,
             np.ndarray state_probability_diffusion,
             np.ndarray state_probability_hurst,
             np.ndarray space_lim)
```

7.4.2 Member Function Documentation

7.4.2.1 _autocovariance()

```
SMS_BP.fbm_BP.FBM_BP._autocovariance (
              self,
              k,
              hurst)  [protected]
```

```
Autocovariance function for fGn

Parameters:
-----------
k : int
    Lag
dt : float
    Time step
hurst : float
    Hurst parameter
diff_a : float
    Diffusion coefficient related to the Hurst parameter

Returns:
--------
float
    Autocovariance function
```

7.4.2.2 _setup()

```
None SMS_BP.fbm_BP.FBM_BP._setup (
              self)  [protected]
```

7.4.2.3 fbm()

```
SMS_BP.fbm_BP.FBM_BP.fbm (
              self)
```

7.4.3 Member Data Documentation

7.4.3.1 _cov

```
SMS_BP.fbm_BP.FBM_BP._cov  [protected]
```

7.4.3.2 _diff_a_n

```
SMS_BP.fbm_BP.FBM_BP._diff_a_n  [protected]
```

7.4.3.3 _hurst_n

```
SMS_BP.fbm_BP.FBM_BP._hurst_n  [protected]
```

7.4.3.4 diffusion_parameter

SMS_BP.fbm_BP.FBM_BP.diffusion_parameter

7.4.3.5 diffusion_parameter_transition_matrix

SMS_BP.fbm_BP.FBM_BP.diffusion_parameter_transition_matrix

7.4.3.6 dt

SMS_BP.fbm_BP.FBM_BP.dt

7.4.3.7 hurst_parameter

SMS_BP.fbm_BP.FBM_BP.hurst_parameter

7.4.3.8 hurst_parameter_transition_matrix

SMS_BP.fbm_BP.FBM_BP.hurst_parameter_transition_matrix

7.4.3.9 n

SMS_BP.fbm_BP.FBM_BP.n

7.4.3.10 space_lim

SMS_BP.fbm_BP.FBM_BP.space_lim

7.4.3.11 state_probability_diffusion

SMS_BP.fbm_BP.FBM_BP.state_probability_diffusion

7.4.3.12 state_probability_hurst

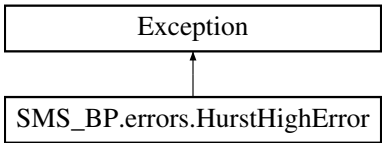
SMS_BP.fbm_BP.FBM_BP.state_probability_hurst

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

- fbm_BP.py

7.5 SMS_BP.errors.HurstHighError Class Reference

Inheritance diagram for SMS_BP.errors.HurstHighError:



7.5.1 Detailed Description

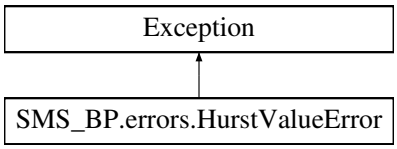
```
Raised when the Hurst value is too high for the space limit
```

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

- errors.py

7.6 SMS_BP.errors.HurstValueError Class Reference

Inheritance diagram for SMS_BP.errors.HurstValueError:



7.6.1 Detailed Description

```
Raised when the Hurst value is not within the range (0, 1)
```

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

- errors.py

7.7 SMS_BP.probability_functions.multiple_top_hat_probability Class Reference

Public Member Functions

- None __init__ (self, int num_subspace, np.ndarray subspace_centers, np.ndarray subspace_radius, float density_dif, np.ndarray space_size)
- float __call__ (self, np.ndarray position, **kwargs)
- None update_parameters (self, int num_subspace=None, np.ndarray subspace_centers=None, np.ndarray subspace_radius=None, float density_dif=None, np.ndarray space_size=None)
- int num_subspace (self)
- None num_subspace (self, int value)
- np.ndarray subspace_centers (self)
- None subspace_centers (self, np.ndarray value)
- np.ndarray subspace_radius (self)
- None subspace_radius (self, np.ndarray value)
- float density_dif (self)
- None density_dif (self, float value)
- np.ndarray space_size (self)
- None space_size (self, np.ndarray value)
- float subspace_probability (self)
- None subspace_probability (self, float value)
- float non_subspace_probability (self)
- None non_subspace_probability (self, float value)

Public Attributes

- num_subspace
- subspace_centers
- subspace_radius
- density_dif
- space_size
- subspace_probability
- non_subspace_probability

Protected Member Functions

- _calculate_subspace_probability (self, np.ndarray space_size, float density_dif)
- _calculate_non_subspace_probability (self, np.ndarray space_size, float density_dif, int num_subspace, np.ndarray subspace_radius)

Protected Attributes

- _num_subspace
- _subspace_centers
- _subspace_radius
- _density_dif
- _space_size
- _subspace_probability
- _non_subspace_probability

7.7.1 Detailed Description

```
 Class for the probability function of multiple top hats.
Once initalized an object of this class can be called to return the probability at a given position.

!!!--DO NOT CHANGE THE PARAMETERS AFTER INITALIZATION DIRECTLY. USE THE UPDATE_PARAMETERS METHOD--!!!
```

7.7.2 Constructor & Destructor Documentation

7.7.2.1 __init__()

```
None SMS_BP.probability_functions.multiple_top_hat_probability.__init__ (
              self,
             int num_subspace,
             np.ndarray subspace_centers,
             np.ndarray subspace_radius,
             float density_dif,
             np.ndarray space_size)
```

7.7.3 Member Function Documentation

7.7.3.1 __call__()

```
float SMS_BP.probability_functions.multiple_top_hat_probability.__call__ (
              self,
             np.ndarray position,
             ** kwargs)

 Returns the probability given a coordinate
```

7.7.3.2 _calculate_non_subspace_probability()

```
SMS_BP.probability_functions.multiple_top_hat_probability._calculate_non_subspace_probability
(
              self,
             np.ndarray space_size,
             float density_dif,
             int num_subspace,
             np.ndarray subspace_radius)  [protected]
```

7.7.3.3 _calculate_subspace_probability()

```
SMS_BP.probability_functions.multiple_top_hat_probability._calculate_subspace_probability (
              self,
             np.ndarray space_size,
             float density_dif)  [protected]
```

7.7.3.4 density_dif() [1/2]

```
float SMS_BP.probability_functions.multiple_top_hat_probability.density_dif (
              self)
```

Returns the difference in density between the subspaces and the rest of the space.

7.7.3.5 density_dif() [2/2]

```
None SMS_BP.probability_functions.multiple_top_hat_probability.density_dif (
              self,
             float value)
```

7.7.3.6 non_subspace_probability() [1/2]

```
float SMS_BP.probability_functions.multiple_top_hat_probability.non_subspace_probability (
              self)
```

Returns the probability of the non-subspaces.

7.7.3.7 non_subspace_probability() [2/2]

```
None SMS_BP.probability_functions.multiple_top_hat_probability.non_subspace_probability (
              self,
             float value)
```

7.7.3.8 num_subspace() [1/2]

```
int SMS_BP.probability_functions.multiple_top_hat_probability.num_subspace (
              self)
```

Returns the number of subspaces.

7.7.3.9 num_subspace() [2/2]

```
None SMS_BP.probability_functions.multiple_top_hat_probability.num_subspace (
              self,
             int value)
```

7.7.3.10 space_size() [1/2]

```
np.ndarray SMS_BP.probability_functions.multiple_top_hat_probability.space_size (
              self)
```

Returns the size of the space.

7.7.3.11 space_size() [2/2]

```
None SMS_BP.probability_functions.multiple_top_hat_probability.space_size (
              self,
             np.ndarray value)
```

7.7.3.12 subspace_centers() [1/2]

```
np.ndarray SMS_BP.probability_functions.multiple_top_hat_probability.subspace_centers (
              self)
```

Returns the centers of the subspaces.

7.7.3.13 subspace_centers() [2/2]

```
None SMS_BP.probability_functions.multiple_top_hat_probability.subspace_centers (
              self,
             np.ndarray value)
```

7.7.3.14 subspace_probability() [1/2]

```
float SMS_BP.probability_functions.multiple_top_hat_probability.subspace_probability (
              self)
```

7.7.3.15 subspace_probability() [2/2]

```
None SMS_BP.probability_functions.multiple_top_hat_probability.subspace_probability (
              self,
             float value)
```

7.7.3.16 subspace_radius() [1/2]

```
np.ndarray SMS_BP.probability_functions.multiple_top_hat_probability.subspace_radius (
              self)
```

Returns the radius of the subspaces.

7.7.3.17 subspace_radius() [2/2]

```
None SMS_BP.probability_functions.multiple_top_hat_probability.subspace_radius (
              self,
             np.ndarray value)
```

7.7.3.18 update_parameters()

```
None SMS_BP.probability_functions.multiple_top_hat_probability.update_parameters (
              self,
             int  num_subspace = None,
             np.ndarray  subspace_centers = None,
             np.ndarray  subspace_radius = None,
             float  density_dif = None,
             np.ndarray  space_size = None)
```

Updates the parameters of the probability function.

7.7.4 Member Data Documentation

7.7.4.1 _density_dif

```
SMS_BP.probability_functions.multiple_top_hat_probability._density_dif  [protected]
```

7.7.4.2 _non_subspace_probability

```
SMS_BP.probability_functions.multiple_top_hat_probability._non_subspace_probability  [protected]
```

7.7.4.3 _num_subspace

```
SMS_BP.probability_functions.multiple_top_hat_probability._num_subspace  [protected]
```

7.7.4.4 _space_size

```
SMS_BP.probability_functions.multiple_top_hat_probability._space_size  [protected]
```

7.7.4.5 _subspace_centers

```
SMS_BP.probability_functions.multiple_top_hat_probability._subspace_centers  [protected]
```

7.7.4.6 _subspace_probability

```
SMS_BP.probability_functions.multiple_top_hat_probability._subspace_probability  [protected]
```

7.7.4.7 _subspace_radius

```
SMS_BP.probability_functions.multiple_top_hat_probability._subspace_radius  [protected]
```

7.7.4.8 density_dif

SMS_BP.probability_functions.multiple_top_hat_probability.density_dif

7.7.4.9 non_subspace_probability

SMS_BP.probability_functions.multiple_top_hat_probability.non_subspace_probability

7.7.4.10 num_subspace

SMS_BP.probability_functions.multiple_top_hat_probability.num_subspace

7.7.4.11 space_size

SMS_BP.probability_functions.multiple_top_hat_probability.space_size

7.7.4.12 subspace_centers

SMS_BP.probability_functions.multiple_top_hat_probability.subspace_centers

7.7.4.13 subspace_probability

SMS_BP.probability_functions.multiple_top_hat_probability.subspace_probability

7.7.4.14 subspace_radius

SMS_BP.probability_functions.multiple_top_hat_probability.subspace_radius

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

- probability_functions.py

7.8 SMS_BP.simulate_cell.Simulate_cells Class Reference

Public Member Functions

- __init__ (self, dict|str init_dict_json)
- dict get_cell (self)
- None get_and_save_sim (self, str cd, str img_name, str subsegment_type, int sub_frame_num, **kwargs)
- dict condensates (self)
- condensates (self, dict condensates)

Public Attributes

- init_dict
- frame_count
- interval_time
- oversample_motion_time
- exposure_time
- total_time
- track_length_mean
- track_diffusion_updated
- condensate_diffusion_updated
- pixel_size_pix
- axial_detection_range_pix
- psf_sigma_pix
- transition_matrix_time_step
- diffusion_transition_matrix
- hurst_transition_matrix
- condensates

Protected Member Functions

- int _convert_frame_to_time (self, int frame, int exposure_time, int interval_time, int oversample_motion_time)
- _update_units (self, unit, orig_type, update_type)
- bool _check_init_dict (self)
- dict _read_json (self, str json_file)
- _define_space (self, dims=(100, 100), movie_frames=500)
- _convert_track_dict_points_per_frame (self, tracks, movie_frames)
- _convert_track_dict_msd (self, tracks)
- _create_track_pop_dict (self, np.ndarray simulation_cube)
- _create_map (self, np.ndarray initial_map, dict points_per_frame, str axial_function)
- dict _point_per_time_selection (self, dict points_per_time)
- _format_points_per_frame (self, points_per_frame)

Protected Attributes

- _condensates

7.8.1 Constructor & Destructor Documentation

7.8.1.1 __init__()

```
SMS_BP.simulate_cell.Simulate_cells.__init__ (
              self,
             dict | str init_dict_json)
```

```
 Docstring for Simulate_cells: Class for simulating cells in space.
Parameters:
-----------
init_dict_json : dict|str
    dictionary of parameters or path to the json file containing the parameters
    see sim_config.md for more details
Returns:
--------
None
```

7.8.2 Member Function Documentation

7.8.2.1 _check_init_dict()

```
bool SMS_BP.simulate_cell.Simulate_cells._check_init_dict (
             self)  [protected]

 Docstring for _check_init_dict: check the init_dict for the required keys, and if they are consistent with ot

Parameters:
-----------
None

Returns:
--------
bool: True if the init_dict is correct

Raises:
-------
InitializationKeys: if the init_dict does not have the required keys
InitializationValues: if the init_dict values are not consistent with each other
```

7.8.2.2 _convert_frame_to_time()

```
int SMS_BP.simulate_cell.Simulate_cells._convert_frame_to_time (
              self,
             int frame,
             int exposure_time,
             int interval_time,
             int oversample_motion_time)  [protected]

 Docstring for _convert_frame_to_time: convert the frame number to time
Parameters:
-----------
frame : int
    frame number
exposure_time : int
    exposure time
interval_time : int
    interval time
oversample_motion_time : int
    oversample motion time
Returns:
--------
int
    time in the appropriate units
```

7.8.2.3 _convert_track_dict_msd()

```
SMS_BP.simulate_cell.Simulate_cells._convert_track_dict_msd (
             self,
             tracks)  [protected]

Docstring for _convert_track_dict_msd: convert the track dictionary to a dictionary of tracks with the format
required for the msd function

Parameters:
-----------
tracks : dict
    dictionary of tracks, keys = track number, values = dictionary with keys = 'xy','frames','diffusion_coeffi
Returns:
--------
track_msd : dict
    dictionary of tracks with the format required for the msd function, keys = track number, values = list of
```

7.8.2.4 _convert_track_dict_points_per_frame()

```
SMS_BP.simulate_cell.Simulate_cells._convert_track_dict_points_per_frame (
              self,
              tracks,
              movie_frames)  [protected]
```

```
Docstring for _convert_track_dict_points_per_frame: convert the track dictionary to a dictionary of points per

Parameters:
-----------
tracks : dict
    dictionary of tracks, keys = track number, values = dictionary with keys = 'xy','frames','diffusion_coeffi
movie_frames : int
    number of frames in the movie
Returns:
--------
points_per_frame : dict
    dictionary of points per frame, keys = frame number, values = list of (x,y,z) tuples
```

7.8.2.5 _create_map()

```
SMS_BP.simulate_cell.Simulate_cells._create_map (
              self,
             np.ndarray initial_map,
             dict points_per_frame,
             str axial_function)  [protected]
```

```
 Docstring for __create_map: create the map for the simulation using the points_per_frame dictionary

Parameters:
-----------
initial_map : array-like
    empty space for simulation
points_per_frame : dict
    dictionary of points per frame (this is different from the points_per_time dictionary, and is sampled at t
axial_function : str
    function to be used to create the axial map
Returns:
--------
map : array-like
    map for the simulation
```

7.8.2.6 _create_track_pop_dict()

```
SMS_BP.simulate_cell.Simulate_cells._create_track_pop_dict (
              self,
             np.ndarray simulation_cube)  [protected]
```

```
 Docstring for _create_cell_tracks: create the tracks for the cell

Parameters:
-----------
simulation_cube : array-like
    empty space for simulation
Returns:
--------
tracks : list
    list of tracks for each cell
points_per_time : list
    list of number of points in each time
```

7.8.2.7 _define_space()

```
SMS_BP.simulate_cell.Simulate_cells._define_space (
              self,
              dims = (100, 100),
              movie_frames = 500)  [protected]
```

```
 Docstring for _define_space: make the empty space for simulation
Parameters:
-----------
dims : tuple, Default = (100,100)
    dimensions of the space to be simulated
movie_frames : int, Default = 500
    number of frames to be simulated
Returns:
--------
space : array-like, shape = (movie_frames,dims[0],dims[1])
    empty space for simulation
```

7.8.2.8 _format_points_per_frame()

```
SMS_BP.simulate_cell.Simulate_cells._format_points_per_frame (
              self,
              points_per_frame)  [protected]
```

```
Docstring for _format_points_per_frame: format the points per frame dictionary so that for each key the set of
converted to a numpy array of N x 2 where N is the total amount of points in that frame. You don't need this f

Parameters:
-----------
points_per_frame : dict
    keys = str(i) for i in range(self.total_time), values = list of tracks, which are collections of [x,y] coo

Returns:
--------
points_per_frame : dict
    keys = str(i) for i in range(movie_frames), values = numpy array of N x 2 where N is the total amount of p
```

7.8.2.9 _point_per_time_selection()

```
dict SMS_BP.simulate_cell.Simulate_cells._point_per_time_selection (
              self,
             dict points_per_time)  [protected]
```

```
 Docstring for _track_and_point_per_time_selection: select the tracks and points per time for the simulation
Parameters:
-----------
points_per_time : dict
    dictionary of points per time
Returns:
--------
points_per_frame: dict
    dictionary of points per frame
```

7.8.2.10 _read_json()

```
dict SMS_BP.simulate_cell.Simulate_cells._read_json (
              self,
             str json_file)  [protected]
```

```
 Docstring for _read_json: read the json file and return the dictionary
Parameters:
-----------
json_file : str
    path to the json file

Returns:
--------
dict
    dictionary of parameters
```

7.8.2.11 _update_units()

```
SMS_BP.simulate_cell.Simulate_cells._update_units (
              self,
              unit,
              orig_type,
              update_type)  [protected]
```

```
 Docstring for _update_units: update the unit from one type to another
Parameters:
-----------
unit : int
    unit to be updated
orig_type : str
    original type of unit
update_type : str
    type to update unit to
```

7.8.2.12 condensates() [1/2]

```
dict SMS_BP.simulate_cell.Simulate_cells.condensates (
              self)
```

7.8.2.13 condensates() [2/2]

```
SMS_BP.simulate_cell.Simulate_cells.condensates (
              self,
             dict condensates)
```

7.8.2.14 get_and_save_sim()

```
None SMS_BP.simulate_cell.Simulate_cells.get_and_save_sim (
              self,
             str cd,
             str img_name,
             str subsegment_type,
             int sub_frame_num,
             ** kwargs)
```

```
 Docstring for make_directory_structure: make the directory structure for the simulation and save the image +
Also perform the subsegmentation and save the subsegments in the appropriate directory
Parameters:
-----------
cd : str
    directory to save the simulation
img_name : str
    name of the image
img : array-like
    image to be subsegmented
subsegment_type : str
    type of subsegmentation to be performed, currently only "mean" is supported
sub_frame_num : int
    number of subsegments to be created
**kwargs : dict
    dictionary of keyword arguments
KWARGS:
-------
data : dict, Default = None
    dictionary of data to be saved, Keys = "map","tracks","points_per_frame" Values = array-like.
    See the return of the function simulate_cell_tracks for more details
parameters : dict, Default = self.init_dict
Returns:
--------
none
```

7.8.2.15 get_cell()

```
dict SMS_BP.simulate_cell.Simulate_cells.get_cell (
              self)
```

```
 Docstring for get_cell: get the cell simulation
Parameters:
-----------
None
Returns:
--------
cell : dict
    dictionary of the cell simulation, keys = "map","tracks","points_per_frame"
```

7.8.3 Member Data Documentation

7.8.3.1 _condensates

```
SMS_BP.simulate_cell.Simulate_cells._condensates  [protected]
```

7.8.3.2 axial_detection_range_pix

SMS_BP.simulate_cell.Simulate_cells.axial_detection_range_pix

7.8.3.3 condensate_diffusion_updated

SMS_BP.simulate_cell.Simulate_cells.condensate_diffusion_updated

7.8.3.4 condensates

SMS_BP.simulate_cell.Simulate_cells.condensates

7.8.3.5 diffusion_transition_matrix

SMS_BP.simulate_cell.Simulate_cells.diffusion_transition_matrix

7.8.3.6 exposure_time

SMS_BP.simulate_cell.Simulate_cells.exposure_time

7.8.3.7 frame_count

SMS_BP.simulate_cell.Simulate_cells.frame_count

7.8.3.8 hurst_transition_matrix

SMS_BP.simulate_cell.Simulate_cells.hurst_transition_matrix

7.8.3.9 init_dict

SMS_BP.simulate_cell.Simulate_cells.init_dict

7.8.3.10 interval_time

SMS_BP.simulate_cell.Simulate_cells.interval_time

7.8.3.11 oversample_motion_time

SMS_BP.simulate_cell.Simulate_cells.oversample_motion_time

7.8.3.12 pixel_size_pix

SMS_BP.simulate_cell.Simulate_cells.pixel_size_pix

7.8.3.13 psf_sigma_pix

SMS_BP.simulate_cell.Simulate_cells.psf_sigma_pix

7.8.3.14 total_time

SMS_BP.simulate_cell.Simulate_cells.total_time

7.8.3.15 track_diffusion_updated

SMS_BP.simulate_cell.Simulate_cells.track_diffusion_updated

7.8.3.16 track_length_mean

SMS_BP.simulate_cell.Simulate_cells.track_length_mean

7.8.3.17 transition_matrix_time_step

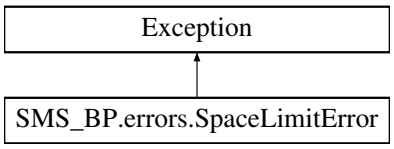
SMS_BP.simulate_cell.Simulate_cells.transition_matrix_time_step

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

- simulate_cell.py

7.9 SMS_BP.errors.SpaceLimitError Class Reference

Inheritance diagram for SMS_BP.errors.SpaceLimitError:



7.9.1 Detailed Description

Raised when the space limit is not within the range (-inf, inf)

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

- errors.py

7.10 SMS_BP.simulate_foci.Track_generator Class Reference

Public Member Functions

- None __init__ (self, np.ndarray|list cell_space, int|float cell_axial_range, int frame_count, int|float exposure_time, int|float interval_time, int|float oversample_motion_time)
- dict track_generation_no_transition (self, float diffusion_coefficient, float hurst_exponent, int track_length, np.ndarray initials, int|float start_time)
- dict track_generation_with_transition (self, np.ndarray|list diffusion_transition_matrix, np.ndarray|list hurst↩ _transition_matrix, np.ndarray|list diffusion_parameters, np.ndarray|list hurst_parameters, np.ndarray|list diffusion_state_probability, np.ndarray|list hurst_state_probability, int track_length, np.ndarray initials, int|float start_time)
- dict track_generation_constant (self, int track_length, np.ndarray initials, int starting_time)

Public Attributes

- cell_space
- min_x
- max_x
- min_y
- max_y
- cell_axial_range
- space_lim
- frame_count
- exposure_time
- interval_time
- oversample_motion_time
- total_time

Protected Member Functions

- int _convert_time_to_frame (self, int time)
- int _convert_frame_to_time (self, int frame)

7.10.1 Constructor & Destructor Documentation

7.10.1.1 __init__()

```
None SMS_BP.simulate_foci.Track_generator.__init__ (
              self,
             np.ndarray | list cell_space,
             int | float cell_axial_range,
             int frame_count,
             int | float exposure_time,
             int | float interval_time,
             int | float oversample_motion_time)
```

7.10.2 Member Function Documentation

7.10.2.1 _convert_frame_to_time()

```
int SMS_BP.simulate_foci.Track_generator._convert_frame_to_time (
              self,
             int frame)  [protected]
```

```
Parameters:
-----------
frame : int
    frame number

Returns:
--------
int: time in ms
```

7.10.2.2 _convert_time_to_frame()

```
int SMS_BP.simulate_foci.Track_generator._convert_time_to_frame (
              self,
             int time)  [protected]
```

```
Parameters:
-----------
time : int
    time in ms

Returns:
--------
int: frame number
```

7.10.2.3 track_generation_constant()

```
dict SMS_BP.simulate_foci.Track_generator.track_generation_constant (
              self,
             int track_length,
             np.ndarray initials,
             int starting_time)
```

```
Parameters:
-----------
track_length : int
    mean track length, in this case the track length is constant with this mean
initials : array-like
    [[x,y,z]] coordinates of the initial positions of the track
starting_time : int
    time at which the track start (this is not the frame, and needs to be converted to the frame using the exp

Returns:
--------
np.ndarray
    track data for the constant track, {"xy":xyz,"frames":frames,"diffusion_coefficient":diffusion_coefficient
```

7.10.2.4 track_generation_no_transition()

```
dict SMS_BP.simulate_foci.Track_generator.track_generation_no_transition (
              self,
             float diffusion_coefficient,
             float hurst_exponent,
             int track_length,
             np.ndarray initials,
             int | float start_time)
```

```
Simulates the track generation with no transition between the diffusion coefficients and the hurst exponents
namely, this means each track has a unique diffusion coefficient and hurst exponent
This simulation is confined to the cell space and the axial range of the cell

Parameters:
-----------
diffusion_coefficient : float
    diffusion coefficient for the track
hurst_exponent : float
    hurst exponent for the track
track_length : int
    track_length for the track
initials : array-like
    [[x,y,z]] coordinates of the initial positions of the track
start_time : int
    time at which the track start (this is not the frame, and needs to be converted to the frame using the exp
Returns:
--------
dict-like with format: {"xy":xyz,"frames":frames,"diffusion_coefficient":diffusion_coefficient,"hurst":hurst_e
```

7.10.2.5 track_generation_with_transition()

```
dict SMS_BP.simulate_foci.Track_generator.track_generation_with_transition (
              self,
             np.ndarray | list diffusion_transition_matrix,
             np.ndarray | list hurst_transition_matrix,
             np.ndarray | list diffusion_parameters,
             np.ndarray | list hurst_parameters,
             np.ndarray | list diffusion_state_probability,
             np.ndarray | list hurst_state_probability,
             int track_length,
             np.ndarray initials,
             int | float start_time)
```

```
Genereates the track data with transition between the diffusion coefficients and the hurst exponents

Parameters:
-----------
diffusion_transition_matrix : array-like
    transition matrix for the diffusion coefficients
hurst_transition_matrix : array-like
    transition matrix for the hurst exponents
diffusion_parameters : array-like
    diffusion coefficients for the tracks
hurst_parameters : array-like
    hurst exponents for the tracks
diffusion_state_probability : array-like
    probabilities for the diffusion coefficients
hurst_state_probability : array-like
    probabilities for the hurst exponents
track_length : int
```

```
    track_length for the track
initials : array-like
    [[x,y,z]] coordinates of the initial positions of the track
start_time : int
    time at which the track start (this is not the frame, and needs to be converted to the frame using the exp

Returns:
--------
dict-like with format: {"xy":xyz,"frames":frames,"diffusion_coefficient":diffusion_coefficient,"hurst":hurst_e
```

7.10.3 Member Data Documentation

7.10.3.1 cell_axial_range

SMS_BP.simulate_foci.Track_generator.cell_axial_range

7.10.3.2 cell_space

SMS_BP.simulate_foci.Track_generator.cell_space

7.10.3.3 exposure_time

SMS_BP.simulate_foci.Track_generator.exposure_time

7.10.3.4 frame_count

SMS_BP.simulate_foci.Track_generator.frame_count

7.10.3.5 interval_time

SMS_BP.simulate_foci.Track_generator.interval_time

7.10.3.6 max_x

SMS_BP.simulate_foci.Track_generator.max_x

7.10.3.7 max_y

SMS_BP.simulate_foci.Track_generator.max_y

7.10.3.8 min_x

SMS_BP.simulate_foci.Track_generator.min_x

7.10.3.9 min_y

SMS_BP.simulate_foci.Track_generator.min_y

7.10.3.10 oversample_motion_time

SMS_BP.simulate_foci.Track_generator.oversample_motion_time

7.10.3.11 space_lim

SMS_BP.simulate_foci.Track_generator.space_lim

7.10.3.12 total_time

SMS_BP.simulate_foci.Track_generator.total_time

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

- simulate_foci.py

Chapter 8

File Documentation

8.1 __init__.py File Reference

Namespaces

- namespace SMS_BP

8.2 boundary_conditions.py File Reference

Namespaces

- namespace SMS_BP
- namespace SMS_BP.boundary_conditions

Functions

- SMS_BP.boundary_conditions._refecting_boundary (float fbm_store_last, float fbm_candidate, np.ndarray space_lim)
- SMS_BP.boundary_conditions._absorbing_boundary (float fbm_store_last, float fbm_candidate, np.ndarray space_lim)

8.3 condensate_movement.py File Reference

Classes

- class SMS_BP.condensate_movement.Condensate

Namespaces

- namespace SMS_BP
- namespace SMS_BP.condensate_movement

8.4 decorators.py File Reference

Classes

- class SMS_BP.decorators.CountCalls

Namespaces

- namespace SMS_BP
- namespace SMS_BP.decorators

Functions

- SMS_BP.decorators.deprecated (reason)
- SMS_BP.decorators.timer (func)
- SMS_BP.decorators.debug (func)
- SMS_BP.decorators.slow_down (_func=None, *, rate=1)
- SMS_BP.decorators.repeat (_func=None, *, num_times=2)
- SMS_BP.decorators.singleton (cls)
- SMS_BP.decorators.cache (func)
- SMS_BP.decorators.set_unit (unit)
- SMS_BP.decorators._catch_recursion_error (func)

Variables

- tuple SMS_BP.decorators.string_types = (type(b''), type(u''))

8.5 errors.py File Reference

Classes

- class SMS_BP.errors.HurstValueError
- class SMS_BP.errors.SpaceLimitError
- class SMS_BP.errors.DiffusionHighError
- class SMS_BP.errors.HurstHighError

Namespaces

- namespace SMS_BP
- namespace SMS_BP.errors

8.6 fbm_BP.py File Reference

Classes

- class SMS_BP.fbm_BP.FBM_BP

Namespaces

- namespace SMS_BP
- namespace SMS_BP.fbm_BP

Functions

- SMS_BP.fbm_BP.MCMC_state_selection (int initial_state_index, np.ndarray transition_matrix, np.ndarray possible_states, int n)
- SMS_BP.fbm_BP._boundary_conditions (float fbm_store_last, float fbm_candidate, np.ndarray space_lim, str condition_type)

Variables

- dict SMS_BP.fbm_BP.BOUNDARY_CONDITIONS
- SMS_BP.fbm_BP.transition_matrix = np.array([[0.4, 0.6], [0.2, 0.8]])
- SMS_BP.fbm_BP.possible_states = np.array([1, 2])
- int SMS_BP.fbm_BP.n = 50000
- int SMS_BP.fbm_BP.initial_state_index = 1
- SMS_BP.fbm_BP.state_select
- SMS_BP.fbm_BP.state_probability = np.zeros(len(possible_states))
- SMS_BP.fbm_BP.total_rate = np.sum(transition_matrix)
- SMS_BP.fbm_BP.true_state_probability = np.sum(transition_matrix, axis=0)/total_rate
- SMS_BP.fbm_BP.label
- SMS_BP.fbm_BP.alpha
- int SMS_BP.fbm_BP.state_1_to_2 = np.zeros(n) - 1
- int SMS_BP.fbm_BP.state_2_to_1 = np.zeros(n) - 1

8.7 fbm_utility.py File Reference

Namespaces

- namespace SMS_BP
- namespace SMS_BP.fbm_utility

Functions

- SMS_BP.fbm_utility.get_fbm_sample (l=1, h=0.5, d=1, n=1)

8.8 probability_functions.py File Reference

Classes

- class SMS_BP.probability_functions.multiple_top_hat_probability

Namespaces

- namespace SMS_BP
- namespace SMS_BP.probability_functions

Functions

- SMS_BP.probability_functions.test_multiple_top_hat_probability ()

8.9 run_cell_simulation.py File Reference

Namespaces

- namespace SMS_BP
- namespace SMS_BP.run_cell_simulation

Functions

- SMS_BP.run_cell_simulation.main_CLI ()
- SMS_BP.run_cell_simulation.main_noCLI (file)

Variables

- SMS_BP.run_cell_simulation.project_directory
- SMS_BP.run_cell_simulation.config_file

8.10 sim_config.md File Reference

8.11 simulate_cell.py File Reference

Classes

- class SMS_BP.simulate_cell.Simulate_cells

Namespaces

- namespace SMS_BP
- namespace SMS_BP.simulate_cell

Functions

- SMS_BP.simulate_cell.save_tiff (image, path, img_name=None)
- SMS_BP.simulate_cell.sub_segment (img, sub_frame_num, img_name=None, subsegment_type="mean")
- SMS_BP.simulate_cell.make_directory_structure (cd, img_name, img, subsegment_type, sub_frame_num, **kwargs)
- SMS_BP.simulate_cell.convert_lists_to_arrays (obj)
- SMS_BP.simulate_cell.convert_arrays_to_lists (obj)

Variables

- SMS_BP.simulate_cell.sim_new
- SMS_BP.simulate_cell.cd
- SMS_BP.simulate_cell.img_name
- SMS_BP.simulate_cell.subsegment_type
- SMS_BP.simulate_cell.sub_frame_num

8.12 simulate_foci.py File Reference

Classes

- class SMS_BP.simulate_foci.Track_generator

Namespaces

- namespace SMS_BP
- namespace SMS_BP.simulate_foci

Functions

- SMS_BP.simulate_foci.get_lengths (str track_distribution, int track_length_mean, int total_tracks)
- dict SMS_BP.simulate_foci.create_condensate_dict (np.ndarray initial_centers, np.ndarray initial_scale, np.↩ ndarray diffusion_coefficient, np.ndarray hurst_exponent, np.ndarray cell_space, float cell_axial_range, ∗∗kwargs)
- SMS_BP.simulate_foci.tophat_function_2d (var, center, radius, bias_subspace, space_prob, ∗∗kwargs)
- SMS_BP.simulate_foci.generate_points (pdf, total_points, min_x, max_x, center, radius, bias_subspace_x, space_prob, density_dif)
- SMS_BP.simulate_foci.generate_points_from_cls (pdf, total_points, min_x, max_x, min_y, max_y, min_↩ z, max_z, density_dif)
- SMS_BP.simulate_foci.generate_radial_points (total_points, center, radius)
- SMS_BP.simulate_foci.generate_sphere_points (total_points, center, radius)
- SMS_BP.simulate_foci.radius_spherical_cap (R, center, z_slice)
- SMS_BP.simulate_foci.get_gaussian (mu, sigma, domain=[list(range(10)), list(range(10))])
- float|np.ndarray SMS_BP.simulate_foci.axial_intensity_factor (float|np.ndarray abs_axial_pos, float detection_range, ∗∗kwargs)
- np.ndarray SMS_BP.simulate_foci.generate_map_from_points (np.ndarray points, float|np.ndarray point_↩ intensity, np.ndarray|None map, bool movie, float base_noise, float psf_sigma)

Index